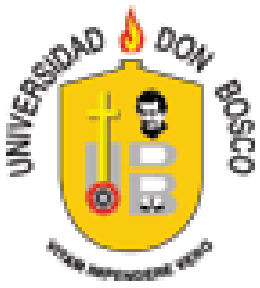


ESTADO DEL ARTE DE LA TECNOLOGÍA DE WEB SERVICES

TRABAJO DE GRADUACIÓN PRESENTADO POR:

EDGARDO ALBERTO ROMERO MASIS

PARA OPTAR AL GRADO DE:

INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

ASESOR:

ING. JULIO ADALBERTO RIVERA PINEDA

SEPTIEMBRE DEL 2004.

SAN SALVADOR, EL SALVADOR, CENTROAMÉRICA.

RECTOR

ING. FEDERICO MIGUEL HUGUET RIVERA

SECRETARIO GENERAL

LIC. MARIO OLMOS

DECANO DE LA FACULTAD DE INGENIERÍA

ING. ERNESTO GODOFREDO GIRÓN

ASESOR DEL TRABAJO DE GRADUACIÓN

ING. JULIO ADALBERTO RIVERA PINEDA

JURADO EVALUADOR

ING. CELIA MOLINA DE HERNÁNDEZ

ING. ARNOLDO INOCENCIO RIVAS MOLINA

ING. RENÉ AMÉRICO HERNÁNDEZ

FACULTAD DE INGENIERÍA

INGENIERIA EN CIENCIAS DE LA COMPUTACIÓN

JURADO EVALUADOR DEL TRABAJO DE GRADUACIÓN

ESTADO DEL ARTE DE LA TECNOLOGÍA DE WEB SERVICES

_____

ING. CELIA MOLINA DE HERNÁNDEZ

_____ _____

ING. ARNOLDO INOCENCIO RIVAS MOLINA **ING. RENÉ AMÉRICO HERNÁNDEZ**

_____

ING. JULIO ADALBERTO RIVERA PINEDA

DEDICATORIA.

A Jehová Dios Todoporeso y a nuestro señor Jesucristo por haberme dado el privilegio de coronar mi carrera, dándome fortaleza, la verdadera paz y felicidad que tanto necesitamos los seres humanos, llenándome de bendiciones todos los días de mi vida.

A mi padre Elio Romero Escamilla por apoyarme incondicionalmente en todas las áreas y etapas de mí recorrido por este mundo, educándome con disciplina y buenos principios para enfrentar en estos días el duro camino de la vida.

A mi madre Ángela Magdalena Masis por ser la persona que me trajo a este mundo, brindarme educación y apoyo a mis ideales, **a mi hermana Karen Zuleyma** por brindarme todo el cariño de siempre.

A mis hijos Jazmín Esmeralda y Edgar Alexis ya que son mis más preciados tesoros que amo con todo mi corazón y el motivo fundamental de superación y lucha por seguir adelante.

A mi amigo Julio Rivera pineda y la familia Rivera Pineda, por darme ánimos de continuar en los momentos que mas lo necesite, además de su colaboración y coordinación en todo el desarrollo del proyecto.

Mis más sinceros agradecimientos a toda mi familia, amigos y personas que han contribuido desinteresadamente directa o indirectamente y hecho posible la realización de esta obra.

ÍNDICE DE CONTENIDO

Introducción. _____ *iii*
Objetivos _____ *iv*
Alcances _____ *v*
Limitaciones _____ *v*
Justificación _____ *vi*
1. Historia de los web services _____ *1*
2. ¿Qué es un web service? _____ *5*
3. Características de los web services. _____ *10*
4. Plataformas de desarrollo _____ *12*
5. Implementación de Web Services. _____ *19*
6. Lenguajes de programación. _____ *22*
7. Estándares y tecnologías asociadas. _____ *31*
8. Los web services y la tecnologia actual _____ *57*
9. Apache Web Server. _____ *83*
10. Apache Project Axis. _____ *99*
11. SOAP _____ *126*
12. XML. _____ *137*
13. Microsoft .NET Framework _____ *157*
14. ASP.NET _____ *186*
15. Introducción a C# _____ *199*
16. Mono Project. _____ *233*
17. PHP. _____ *248*
18. JAVA. _____ *258*
Conclusiones _____ *279*
Bibliografía _____ *280*
Glosario _____ *282*
Presupuesto _____ *286*
Cronograma de actividades _____ *287*

ÍNDICE DE FIGURAS

Figura 1. Web Interactiva y Web Programable....4
Figura 2. Secuencia de Flujo de Información. 8
Figura 3. Secuencia de funcionamiento de los Web Services....9
Figura 4. Ejemplo de contratación de viaje. 11
Figura 5. Capas de estructura de Web Services....20
Figura 6. Estructura de Interacción. 21
Figura 7. Estándares y Tecnologías asociadas a Web Services....31
Figura 8. Modelo Soap 32
Figura 9. Estructura de SOAP....33
Figura 10. Funcionamiento UDDI 51
Figura 11. Representación de datos con HTML y XML....54
Figura 12. Tecnología XML 55
Figura 13. Aplicaciones XML....55
Figura 14. Servlet Engine. 100
Figura 15. Application Servers....101
Figura 16. Modelo de implementación de servicios....107
Figura 17. Modelo de Ejecución. 108
Figura 18. Proceso SOAP 111
Figura 19. TCP Monitor....112
Figura 20. Servicios Web XML....157
Figura 21. NET Framework en tres partes....166
Figura 22. .NET Framework 166
Figura 23.Arquitectura de .Net Framework. 167
Figura 24. Interfaces 168
Figura 25. Implementación de interfaces 168
Figura 26. Common Type System. 170
Figura 27. Soporte Multilenguaje de .NET 171
Figura 28. Interoperabilidad de .NET con COM. 173
Figura 29. Common Language Runtime....174
Figura 30. Lo que ve un programador: Las clases unificadas....177
Figura 31. Biblioteca de clases de .NET Framework....178
Figura 32. .NET Pet Shop vs Java Pet Store – Lines of Code....180
Figura 33. Nile Benchmark of .NET....181
Figura 34. Ventaja de creación de nuevo proyecto en Visual Studio.NET. 211
Figura 35. Plantilla para aplicaciones de consola generada por Visual Studio.NET....212
Figura 36. Hoja de propiedades del proyecto en Visual Studio .NET 213
Figura 37. Código de región expandido....222
Figura 38. Código de región comprimido 222
Figura 39. Páginas de propiedades del proyecto en Visual Studio .NET 228

INTRODUCCIÓN.

El uso de los web services en el mundo se está ampliando rápidamente, mientras que la necesidad de la comunicación y de la interoperabilidad entre aplicaciones crece rápidamente. Los web services proporcionan medios de la comunicación, estándares entre diversos usos del software implicados en la presentación de información dinámica al usuario. Para promover interoperabilidad e integración entre aplicaciones, permitir que más operaciones sean combinadas para realizar tareas más complejas, se necesita una arquitectura estándar.

El contenido presentado en el siguiente trabajo está enfocado en una investigación Bibliográfica sobre las diversas tecnologías que se usan para el diseño e implementación de web services, describir los requisitos para una arquitectura estándar de la referencia para estos, principales características, documentación estandarizada o manuales como son los RFC, white paper's, protocolos y plataformas de operación, ventajas y desventajas que presentan, estándares de lenguajes de programación etc.

OBJETIVOS

Objetivo General.

- Desarrollar una investigación bibliográfica sobre los web services así como las principales tecnologías de implementación de estos en la actualidad, con el fin de mostrar su influencia en los factores tecnológicos y económicos de la sociedad.

Objetivos Específicos.

- Definir la arquitectura de los web services.
- Mostrar las diferentes alternativas que existen para la implementación de plataformas tecnológicas de apoyo a sistemas de web services.
- Mostrar los lenguajes de programación mas importantes para el desarrollo de web services incluyendo e lenguaje de marcado utilizado para la implementación de web services.
- Establecer el funcionamiento y configuración de los principales protocolos de comunicaciones utilizados en la implementación de web services (SOAP).

ALCANCES

- La investigación contempla la información mas importante sobre los web services, si el lector necesita obtener mas detalles al respecto se proveerán citas a los manuales o libro de referencia, por lo que en ningún momento debe ser comparado con estos.

- Establecerá las principales características de las diferentes plataformas tecnológicas para la implementación de web services.

- Abarcara las diferentes tecnologías considerando sus características, ventajas y desventajas, así como también los documentos estándares sobre dicha tecnología.

LIMITACIONES

- El Proyecto no contempla el desarrollo de software dentro del mismo, sino más bien algunas demostraciones prácticas.

- La escasa documentación en español dificulta el proceso de investigación por lo que se retomaran datos en otros idiomas generalmente inglés.

- Algunas tecnologías de tipo propietarias o de patentes electrónicas pueden impedir el desarrollo de la investigación por lo que no se consideran modelos de este tipo.

JUSTIFICACIÓN

La capacidad de desarrollo de aplicaciones distribuidas que caracteriza al modelo de los web services es realmente sorprendente. Por ejemplo una empresa puede tener un servicio de pago electrónico en línea y ofrecérselo a sus socios que, a su vez, pueden conectarse a él independientemente de la plataforma que utilicen, Las empresas de alquiler de vehículos pueden integrar sus sistemas de reserva en línea con aerolíneas y hoteles, con el fin de que el cliente pueda reservar un auto, un vuelo y una habitación de hotel a la vez; es por esto que se observa que el futuro está en los web services ya que forman parte de la plataforma de negocios en Internet del mañana.

A medida que las empresas de envíos, de servicios y de pago electrónico comiencen a ofrecer sus servicios sistemas por medio de los web services se facilitará la conexión a los sitios de comercio electrónico que se estén creando, permitiendo así la interacción directa entre empresas de diferentes rubros y agilizando de esta forma los procesos de negocios.

Hasta el momento nuestro país no cuenta con ninguna investigación de este tipo, con la realización de esta se le abrirá una puerta más, a las empresas salvadoreñas de informarse y conocer detalladamente sobre las ultimas tecnologías que se están implantando a lo largo del globo terrestre. Además el estudio permitirá aproximarse al IOS (Internet OperatIng System) o el Internet del futuro, logrando definir una plataforma para la incorporar los web services al interconectarlos a los sistemas que ya posean las empresas. En el sector educativo se obtendrá plasmar los aspectos fundamentales para desarrollar otras tesis sobre el tema, a la vez servirá como fuente alterna de consulta para impartir clases sobre el área.

1. HISTORIA DE LOS WEB SERVICES

1.1 Repaso a la historia de la Informática Distribuida.

Hace años, todas las aplicaciones informáticas de importancia se llevaban a cabo mediante grandes ordenadores. Luego aparecieron las terminales para conectarse a estos grandes ordenadores, de forma que los usuarios pudieran utilizarlos por medio de comandos en texto normal. Algunos años después, surgió el ordenador personal o PC, desde el cual los usuarios podían ejecutar sus propias aplicaciones.

En los 80' s, en particular en el sector de los ordenadores personales, los protocolos de comunicación no ocupaban un lugar demasiado importante para los desarrolladores, la dificultad consistía en que varias aplicaciones se comunicaran entre si.

En los 90' s, algunas estructuras de objetos, como COM (Component Object CODEL) de Microsoft y CORBA (Common Object Request Broker Architecture), que se comercializó como una iniciativa entre proveedores de OMG (Grupo de Gestión de Objetos), cobraron popularidad.

COM y CORBA eran modelos y arquitecturas diseñadas para la escritura y encapsulamiento del código binario, componentes que eran llamados por los programadores desde cualquier aplicación.

COM y CORBA no interoperaban fácilmente. En estos tiempos lejanos, las máquinas informáticas independientes eran las que dominaban el mundo.

Informática distribuida a nivel de aplicación que se comunica con otra aplicación La comunicación a bajo nivel maquina-maquina ya estaba disponible para los 90' s.

1.2 La era de las redes locales.

La extensión de las redes locales a principios de los 90' s, la conexión entre máquinas se volvió una prioridad. Los proveedores y las organizaciones que ya contaban con sus propias estructuras de modelo-objeto las ampliaron para permitir la comunicación a través de redes.

OMG estableció el IIOP (Internet Inter-ORB Protocol) como protocolo de cable estándar de CORBA.

Microsoft introdujo el DCOM (Distributed COM) como su protocolo de cable que permitía cruzar las fronteras entre equipos. Otro poderoso aspirante surgió con posterioridad a IIOP y DCOM, el RMI (Remote Method Invocation) de Sun MicroSystems, utilizado por los usuarios de Java.

1.3 La era de Internet y la Web.

La conexión de aplicaciones mediante el uso de cualquiera de los protocolos mencionados con anterioridad se caracteriza por un buen funcionamiento, en especial cuando dichas aplicaciones se encuentran en la misma red local.

Con la aparición de Internet, y en particular de la Web, la red creció inmensamente, y se volvió extremadamente distribuida y descentralizada.

Ni personas, ni empresas eran capaces de tomar la decisión sobre que sistema operativo o entorno de programación/lenguaje, se ejecutaría en los diversos ordenadores conectados a Internet. Esto significa que las reglas que tienen vigencia en la red local, no funcionan de forma óptima en Internet y la Web.

La pregunta es ¿Cuáles son los retos actuales en la utilización de Internet y la Web de los protocolos de aplicación distribuida?

Primero surgieron las Web Aisladas. Luego surgió la interoperabilidad en aplicaciones y sitios Web por medio de los Frames (Frameset). Otra forma que se dio fue, cuando el servidor Web actual como un cliente de otra aplicación Web y rastrea el contenido pertinente de la página.

Posteriormente se empleo CGI (Common Gateway Interface), para la publicación de la información (HTTP GET y POST).

Tras la aparición de XML a mediados de los noventa, los desarrolladores se encontraron con la posibilidad de expresar una estructura de información y mensajes de forma autodescriptiva y uniforme, lo que impulso a XML, para aplicar un formato a los mensajes intercambiados entre sistemas. Esta técnica permite que los usuarios intercambien mensajes de formato correcto entre sistemas de forma autodocumentada/autodescriptiva y extensible, independientemente del sistema operativo o del entorno de lenguaje de los sistemas.

Uno de los primeros protocolos para la comunicación mediante el protocolo HTTP, que permite llamar a procedimientos remotos sin importar el lenguaje de desarrollo o el sistema operativo fue XML-RPC, sin embargo a pesar de ser el primero en este campo y de las ventajas que ofrece la mayoría de proveedores optaron por su sucesor SOAP (Simple Object Access Protocol).

El concepto de web services comenzó a tomar forma definitiva con la introducción de SOAP como protocolo de mensajería entre ordenadores. SOAP es un protocolo de cable sencillo basado en XML, se diseño para conexión entre ordenadores independientes de subsistemas operativos, lenguajes de programación o modelos de objetos utilizados (e incluso con la carencia total del modelo de objeto).

A pesar de que su nombre pueda parecer que requiere del uso de determinados objetos, SOAP especifica el formato del mensaje que accede e invoca a los objetos, en vez de especificar los objetos en si.

SOAP es un protocolo sencillo y extensible para la comunicación de ordenador, que emplea por igual los estándares actuales de Internet: XML para el formato de mensajes, http y otros protocolos de Internet para el transporte de mensajes.

En mayo del año 2000, el W3C (World Wide Web Consortium http://www.w3c.org/) reconoció la propuesta de SOAP presentada de forma conjunta por un conglomerado de empresas (Ariba Inc., CommerceOne Inc., Compaq Computer Corp., Microsoft Corp., Development Corp., IBM Corp, Hewlett-Packard Co., etc). Siendo desarrollado en base a un estándar abierto.

1.4 Evolución histórica del Web.

Con el surgimiento del Internet las aplicaciones en la web partieron usando el lenguaje de etiquetas HTML, permitían usar la web como medio para distribuir documentos estáticos, estos se alojaban en servidores Web que usaban el protocolo de comunicación HTTP, luego la información se desplegada por navegadores web en cualquier cliente en el mundo que los invocara por medio de una dirección web conocida como URL, con el paso del tiempo se observo la necesidad de generar información dinámicamente, como solución ante esta se desarrollaron aplicaciones web, usando códigos fuente escritos en diferentes lenguajes de programación como ASP, PHP, PERL, etc. Estos códigos fuentes se alojaban y se ejecutaban en servidores web, estos generaban HTML en forma dinámica. Varias tecnologías: desde CGI a Servlets.

Luego los servidores web permitieron incorporar capacidades de programación distribuida. Pero las necesidades de los usuarios cada día son mayores hasta que se necesito integrar las aplicaciones web con otras del mismo tipo, integrar aplicaciones entre diferentes plataformas, lenguajes de programación, sistemas operativos, compañías, etc. como alternativas de a tal necesidad la representación de información (como XML) y nuevos mecanismos de transporte (como SOAP) aparecieron uniéndose para formar lo que ahora se conoce como los web services.

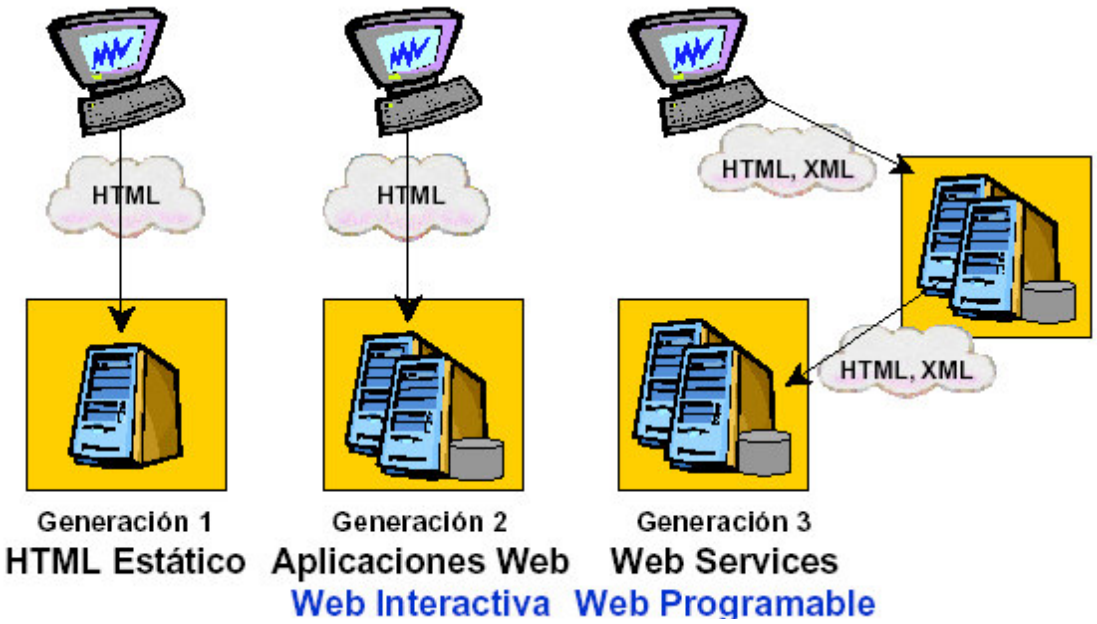


Figura 1. Web Interactiva y Web Programable.

2. ¿QUÉ ES UN WEB SERVICE?

Un web service es un programa modular y autodescriptivo que se encuentra alojado dentro de un servidor web, sus funciones pueden ser solicitadas por aplicaciones cliente que tengan la capacidad de ejecutarlas esto se hace posible mediante el envío y recepción de mensajes SOAP a través del protocolo de transferencias de hipertexto HTTP y el lenguaje XML, pueden hacerlo desde cualquier red de área local que satisfaga los estándares de Internet o desde la web, los web services se identifican en la red por una URI, cada web service es creado para cumplir una función especifica permitiendo que los programas escritos en lenguajes y plataformas diferentes puedan establecer comunicación entre sí de forma estándar, es por esta razón que los web services son independientes del sistema operativo o lenguaje de programación en que fueron creados o desarrollados, además los web services tienen la capacidad de reutilizarse o combinarse, trabajando de forma integral con aplicaciones ya existentes para realizar mas tareas, el funcionamiento de un web service se define por el lenguaje Web Service Description Language WSDL y para encontrarlos usan (UDDI), el World Wide Web Consultorium es la entidad que se encarga de establecer los estándares que deben cumplir los web services, aplicaciones mas complejas se desarrollan usando lenguaje XML.http://www.ietf.org/rfc/rfc2396.txt

Según el World Wide Web Consultorium, web service es un sistema software identificado por una URI (Uniform Resource Identifiers: sintaxis Generica, IETF RFC 2396, T. Berners-Lee, R. Fielding, L. Masinter, August 1998 vea <>.) cuyas interfaces públicos están definidos y descritos mediante XML. Esta definición puede ser accedida por otros sistemas software, los cuales pueden interactuar con el web service en la forma prescrita en su definición, utilizando mensajes XML y transportados por protocolos Internet.

2.1 ¿Cómo funcionan los web services?

Los web services se construyen sobre el acoplamiento del modelo de programación web tradicional, y lo extienden para su uso en otros tipos de aplicaciones. Existen tres diferencias principales entre los web services y las aplicaciones web tradicionales: los web services utilizan mensajes SOAP en lugar de mensajes MIME, los web services no son dependientes de HTTP y los web services proporcionan metadatos que describen los mensajes que producen y consumen.

En primer lugar, los servicios Web se comunican utilizando mensajes SOAP. SOAP formaliza el uso de XML como un modo de pasar datos de un proceso a otro. SOAP define un modelo de trama para el versionado y extensibilidad del protocolo, un modo de transportar información de errores y un modo de enviar mensajes sobre HTTP. El cuerpo de un mensaje SOAP contiene cualquier tipo de XML que una aplicación desee enviar.

El cambio de mensajes escritos con MIME a mensajes basados en XML refleja una diferencia fundamental entre el cliente de una aplicación web tradicional (un navegador) y el cliente de un web service.

Generalmente, los navegadores representan páginas HTML (u otros datos escritos con MIME, como imágenes) y dejan al usuario la interpretación de la información que visualizan. Por otra parte, los clientes de los web services normalmente necesitan interpretar los datos que reciben y hacer algo útil con ellos; incluso pueden no tener un interfaz de usuario. XML proporciona un modo estándar de representar y manipular datos, y las herramientas de procesamiento XML son una lógica elección como formato de mensajes para los web services.

Aunque SOAP autoriza el uso de XML para representar el contenido de un mensaje, no dice nada sobre cuál debe ser ese aspecto de XML. Corresponde a los diseñadores de web services, decidir qué debe transportar cada mensaje. En algunos casos, los mensajes pueden contener parámetros para invocaciones a métodos. La ventaja de esta aproximación es que resulta muy familiar.

Específicamente, la mayoría de los sistemas centrados en métodos esperan que un mensaje contenga exactamente el número adecuado de argumentos, en el orden adecuado y con los tipos adecuados. En algunos casos, los mensajes pueden contener información que no represente la llamada a un método. Esta aproximación permite un acoplamiento débil; los clientes y servidores pueden tener una mayor flexibilidad sobre el formato y contenido de los datos que producen y consumen. Este modelo de programación es parecido al modelo clásico Web, que no indica cómo se procesan los datos enviados en un mensaje.

La segunda diferencia principal entre los web services y las aplicaciones web tradicionales es que los web services no son específicos de los protocolos de transporte. Aunque la especificación SOAP únicamente define cómo enviar mensajes sobre HTTP (y eso es lo que la amplia mayoría de los web services actuales realizan) también pueden utilizarse otros protocolos de transporte. Es posible enviar mensajes SOAP utilizando SMTP, TCP, un protocolo de mensajería instantánea como Jabber o cualquier otro protocolo. Aunque la mayoría de mensajes SOAP se enviarán sobre HTTP en un futuro inmediato, la capacidad de utilizar otros protocolos es muy importante. HTTP no se diseñó para soportar peticiones ejecutándose durante mucho tiempo o para enviar notificaciones de eventos a clientes. Estos aspectos se solucionan mejor utilizando otros protocolos, y con el tiempo se dispondrá de un soporte estandarizado para ello.

Como los web services pueden comunicarse utilizando una amplia variedad de protocolos de transporte, es necesario definir servicios de más alto nivel, como la seguridad, de un modo centrado en mensajes y neutral respecto del transporte. Para soportarlo, el formato de un mensaje SOAP incluye un elemento de cabecera opcional. La cabecera transporta metadatos del mensaje; es decir, información adicional no relacionada directamente con la información sobre el dominio del problema en el cuerpo del mensaje. Este mecanismo se inspiró en HTTP, que utiliza cabeceras de mensaje como un modo de extender su comportamiento básico de petición/respuesta. Definir extensiones de protocolo al nivel de SOAP garantiza un buen entendimiento de la semántica de mensaje, independientemente del protocolo de transporte utilizado para su entrega.

Además de ser neutral respecto del transporte, SOAP no requiere el envío de un mensaje desde un cliente hasta un servidor en un solo "salto". La especificación SOAP define la noción de intermediarios, nodos por los que cruza un mensaje en su camino hacia su destino final. Utilizando intermediarios, se puede "virtualizar" la topología de red física para poder enviar mensajes a web services utilizando el camino y la combinación de protocolos que resulten más apropiados. No existe ningún soporte generalizado para utilizar intermediarios SOAP con los kits de herramientas de web services actuales, pero estará disponible en un futuro. Sin embargo, cuando esta funcionalidad sea más convencional, se podrá implantar web services en una amplia variedad de configuraciones de red sin tener que modificar código de cliente ni de servidor.

La tercera diferencia principal entre los web services y las aplicaciones Web tradicionales es que los web services son autodescriptivos; proporcionan metadatos que describen los mensajes que producen y consumen, y la información de direccionamiento requerida para invocarlos. Los formatos de mensaje de un web service, se definen utilizando XML Schema (XSD). XML Schema es suficientemente flexible para describir un amplio número de estructuras de mensajes, incluyendo modelos de contenidos abiertos con un control fino sobre la extensibilidad, lo que resulta crítico para que los servicios sean débilmente acoplados respecto a los datos que envían y reciben. Los comportamientos de un web service se describen utilizando el lenguaje Web Service Description Language (WSDL), que mapea intercambios de mensajes a operaciones agrupadas en portTypes (interfaces) y describe cómo pueden invocarse estas operaciones utilizando enlaces de protocolos de transporte particulares. Se Puede utilizar estas descripciones para crear software que se comunica con un web service, tanto directa como indirectamente mediante alguna herramienta de generación de código.

Los web services hacen referencia a la creación de una plataforma nueva y de propósito general para construir sistemas distribuidos débilmente acoplados. En el mundo nuevo de los web services, hay cuatro cosas que recordar:

- Todo se basa en un acoplamiento débil. En ello se basa el éxito de la Web y lo que hace interesantes los web servicies.

- Todo se basa en XML. Cuanto más se conozca de XML mucho mejor.

- Se puede utilizar objetos para implementar web services, pero éstos no son el núcleo del modelo de programación.

- La evolución de la plataforma continúa. Hoy se pueden construir web services básicos en una amplia variedad de plataformas. Se sigue trabajando en web services de más alto nivel, el uso de protocolos de transporte alternativos y otros temas interesantes.

Secuencia de flujo de información:

1. Empresa A diseña y pone en marcha un web service.
2. La empresa hace uso de WSDL para describir el servicio.
3. A continuación registra el servicio en un repositorio UDDI o en un registro web XML.
4. La empresa B localiza y solicita el servicio registrado al consultar los repositorios UDDI y/o web XML.
5. El servicio o usuario cliente escribe una aplicación que realice binding del servicio registrado utilizando SOAP (en el caso de UDDI) y/o ebXML
6. Se intercambian las empresas A y B datos y mensajes XML sobre HTTP.

Para que un web service se encuentre habilitado debe residir residen en un servidor remoto. Los web services utilizan estándares que indican como deben ser las peticiones o aplicaciones informáticas instaladas en otro ordenador, brindando un conjunto de protocolos que permiten a las aplicaciones integrar su funcionalidad y sus datos a otras aplicaciones a través de Internet.

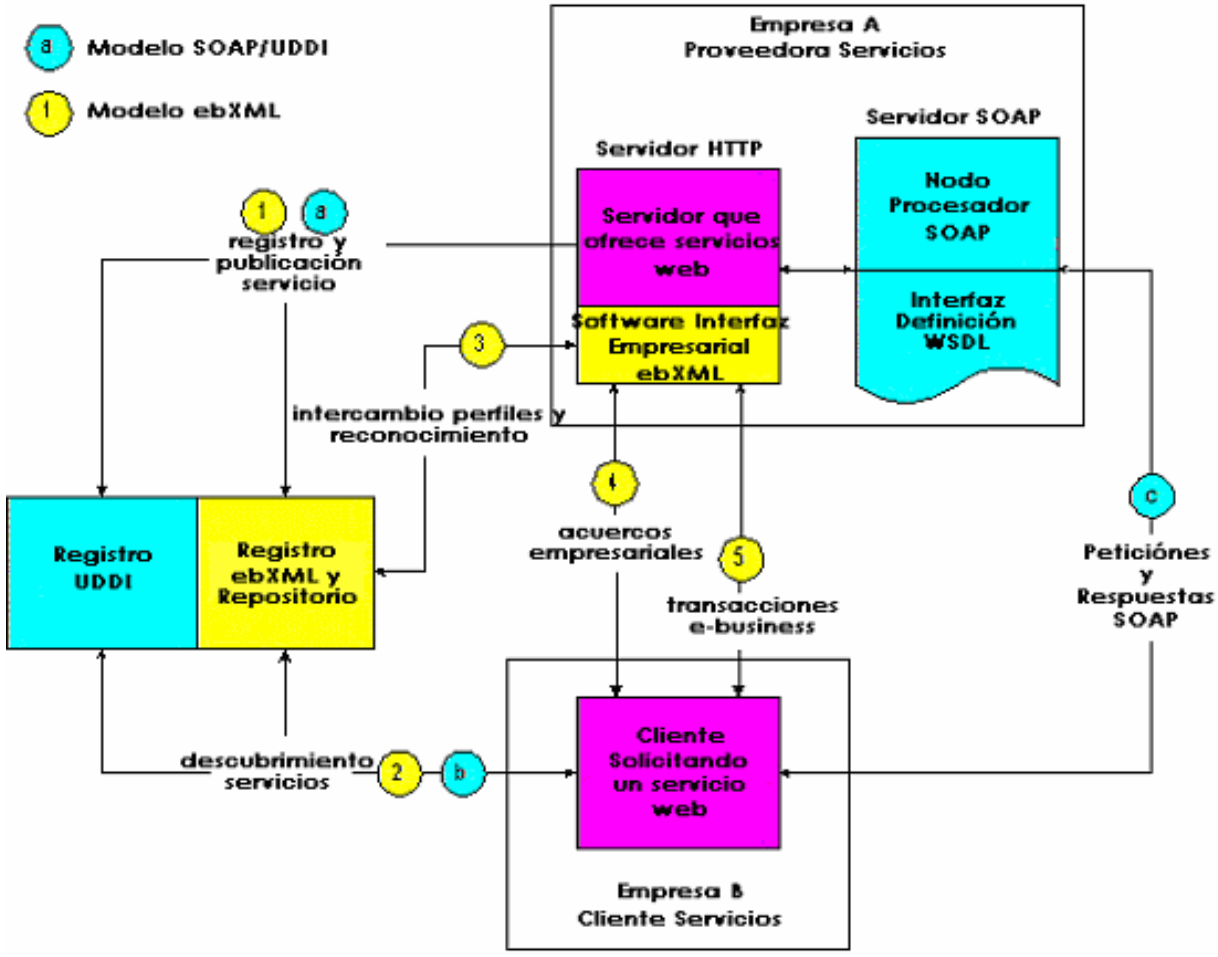


Figura 2. Secuencia de Flujo de Información.

Secuencia del funcionamiento de los web services

1. Cliente pregunta al registro UDDI en que lugar se encuentra un web service que realice una función determinada.

2. El registro UDDI, le indica al cliente en que servidor puede encontrar el web service.

3. El cliente hace la petición al servidor del lenguaje necesario para interactuar con el web service a este se le conoce como documento WSDL.

4. El servidor provee el documento WSDL necesario para interactuar con el Web service.

5. El Cliente invoca la petición al web service usando el protocolo SOAP.

6. Web service retorna una respuesta mediante el protocolo SOAP.

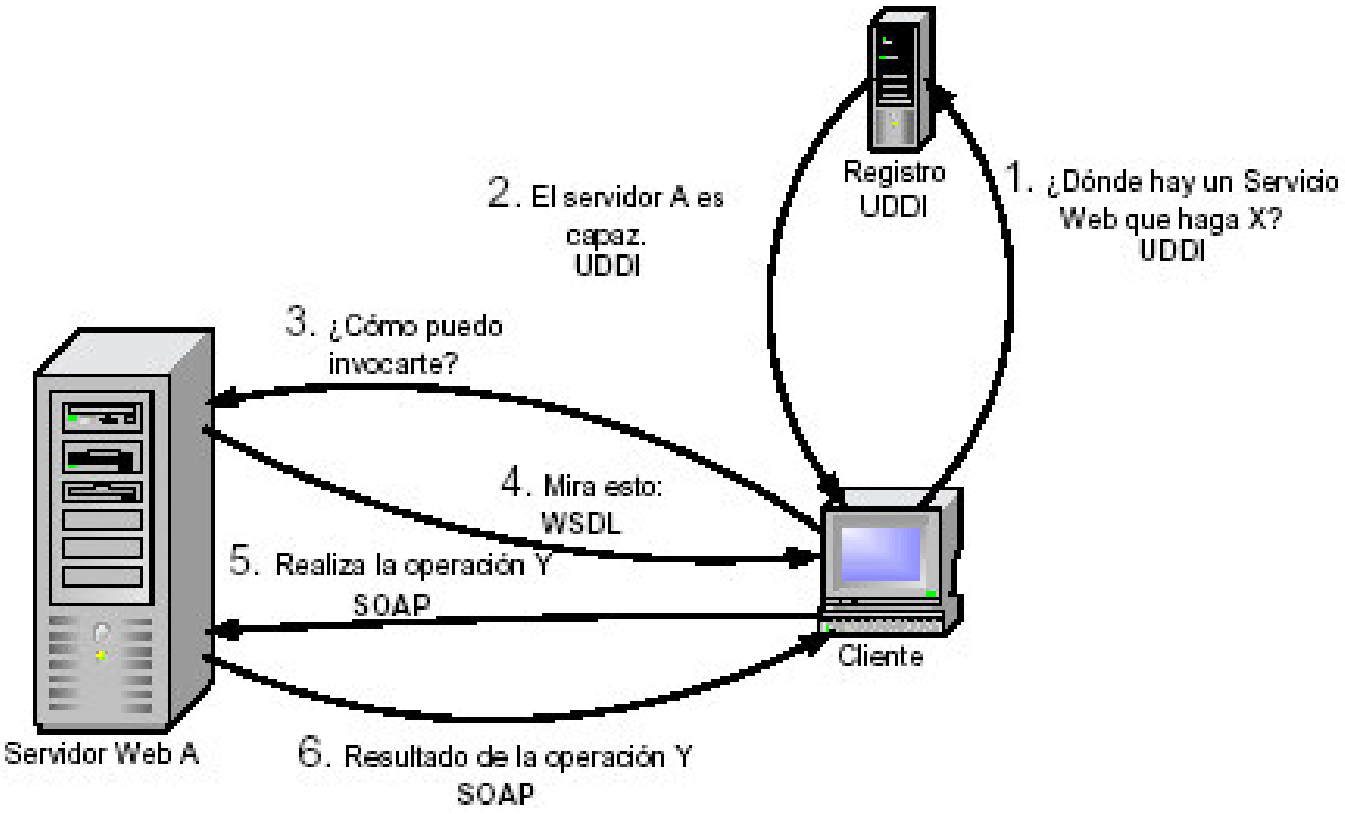


Figura 3. Secuencia de funcionamiento de los Web Services.

3. CARACTERÍSTICAS DE LOS WEB SERVICES.

Algunas de las principales características que resaltan los web services son:

- Integración entre aplicaciones.
- Estructurados en forma de componentes de software en la red.
- Lenguaje y sintaxis independiente de la plataforma.
- Implementación mediante XML, se obtiene como resultado, que cualquier plataforma use esta tecnología.
- Alta disponibilidad.
- Tolerancia a fallas.
- Escalabilidad.
- Buen Rendimiento (performance).

Desarrollo de aplicaciones distribuidas altamente integradas que interactúan por XML entre los web services existentes.

Ofrecen funciones a usuarios empleando un protocolo estándar que, en casi todos los casos, es SOAP.

Usan un lenguaje propio llamado WSDL (lenguaje de descripción de web services) que permite describir sus interfaces con suficiente detalle, esta se proporciona normalmente en un documento XML.

Se registran para que los futuros usuarios los encuentren fácilmente. Este registro se realiza a través de UDDI (descripción, descubrimiento e integración universales).

3.1 ¿Para qué sirven los web services?

Los primeros web services solían ser fuentes de información que podían incorporarse fácilmente en aplicaciones para cotizaciones, previsión del tiempo, resultados deportivos, etc.

Por ejemplo, una hoja de cálculo de Microsoft® Excel en la que se resumiría la siguiente información: acciones, plan de pensiones, cuentas bancarias, préstamos, etc. Si esta información estuviera disponible a través de web services, Excel podría actualizarla continuamente. Parte de dicha información sería gratuita y otra necesitaría que el usuario se subscribiera al servicio. La mayor parte de esta información está disponible ahora en Internet. Sin embargo, los web services posibilitan que sea más fácil y fiable tener acceso a ella. Los web services permiten que los usuarios creen aplicaciones mas potentes usándolos como elementos constituyentes, ejemplo, un usuario puede desarrollar una aplicación de compra para obtener automáticamente de varios fabricantes información sobre precios, permitir seleccionar un fabricante, enviar el pedido de compra y realizar un seguimiento hasta el momento de recibir el envío.

Es posible que la aplicación del fabricante, además de exponer sus servicios en la web, utilice los web Services para comprobar el crédito del cliente, cargar el importe del envío en su cuenta y enviar el pedido a través de una empresa de transportes.

La tecnología de los web services proporciona la base para la integración y agregación de aplicaciones. Desde estas especificaciones base, las empresas están generando soluciones reales y obteniendo valor real de las mismas. Aunque se ha trabajado mucho para que los web services sean una realidad, es necesario hacer mucho más. Actualmente los web services tienen un gran éxito entre la gente, pero todavía hay temas sobre los que los desarrolladores deben seguir trabajando; por ejemplo, la seguridad, gestión operacional, transacciones, mensajería fiable, etc.

La arquitectura Global XML Web Services Architecture (GXA) permitirá que los web services evolucionen ofreciendo un modelo consistente y de propósito general para añadir nuevas capacidades avanzadas a los web services actuales de modo modular y extensible.

Se cree que en el futuro, algunos de los web services más impresionantes admitirán aplicaciones que utilicen el medio web para realizar tareas que no pueden realizarse actualmente. Por ejemplo, uno de los servicios que el proyecto Microsoft .NET My Services admitirá es un servicio de calendario. Así, si su dentista y mecánico expusieran sus calendarios mediante este web service, el usuario podría concertar citas en línea desde su calendario para una limpieza dental y mantenimiento rutinario. Y es fácil imaginar los cientos de aplicaciones que podrían diseñarse cuando se tiene el conocimiento para programar el medio Web.

Ejemplo: Contratar un viaje.

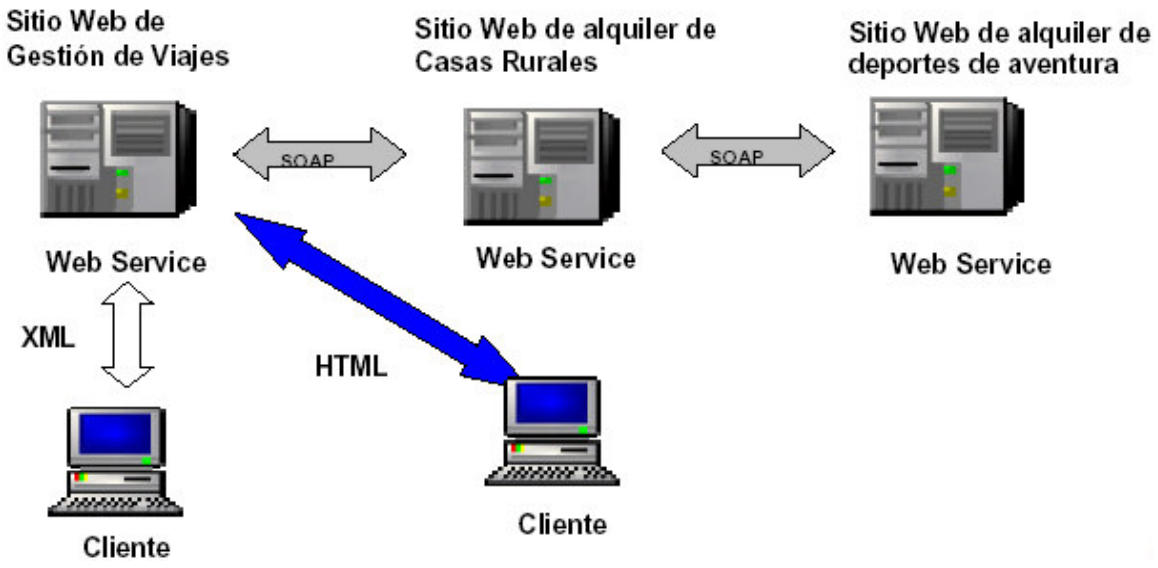


Figura 4. Ejemplo de contratación de viaje.

4. PLATAFORMAS DE DESARROLLO

4.1 WebSphere de IBM (http://www.ibm.com).

IBM ofrece varias herramientas para la creación de los sistemas de web services, en lugar de establecer un solo producto como solución, proporciona una gamma de productos entre los cuales se destacan DB2 (usado como gestor de bases de datos), Lotus Notes (groupware), Tivoli (gestor de bases de datos), y la aplicación en donde se percibe mas fácilmente, WebSphere (Infraestructura de Internet). Como un ejemplo, DB2 incorpora varios herramientas para la conversión continua de los contenidos de bases de datos entre XML, y el servidor de dominio dentro de Lotus puede exportar algunos servicios como web services.

WebSphere es el producto de IBM más importante en cuanto a web services se refiere. WebSphere es una gamma de herramientas, todas ellas relacionadas por el hecho de ser útiles para el diseño y creación de una de web services.

Entre sus componentes esta un servidor web, un servidor directorio, herramientas de autorización de pagina web, servicios de seguridad, gestión de sesión y un amplia variedad de herramientas para la creación de un sitio de comercio electrónico. El componente más importante de WebSphere en relación con los web services, es el Servidor de la Aplicación WebSphere (WebSphere Applicatton Server). Este servidor de aplicación incluye herramientas como un marco J2EE para el desarrollo y la implementación de Enterprise Java Beans, controladores JDBC para la conexión a bases de datos, un contenedor servlet, un kit para desarrollo de Java, servicios de transacción y un grupo de herramientas enfocadas especialmente al desarrollo de una aplicación basada en el modelo de web services descritos con anterioridad.

Estas herramientas especificas para los web services incluyen bibliotecas Java para la producción y el análisis de XML, la transmisión de mensajes SOAP, la creación de archivos WSDL y para establecer comunicación con un Registro de! Servidor por medio de UDDI.

El conjunto de herramientas de SOAP que incluye IBM en el Servidor de la Aplicación WebSphere es el mismo que ofrece Apache (Servidor Web) de forma gratuita. No obstante, como sucede en el caso del JDK que se envía con WebSphere, la versión se selecciona cuidadosamente para asegurar la compatibilidad con el resto de WebSphere. Por esta razón, resulta más conveniente el uso del JDK y SOAP que se incluyen en WebSphere cuando se vayan a usar otras herramientas de WebSphere.

WebSphere incluye una herramienta llamada wsdlgen que se utiliza para generar archivos WSDL para describir un web service. La información de un archivo WSDL incluye descripciones operativas de las funciones que un web service es capaz de realizar, los protocolos que se pueden utilizar para comunicarse con él y los extremos en Internet donde se puede contactar.

Un archivo WSDL es legible, pero al tratarse de XML, pierde esta condición. Por lo tanto, la herramienta wsdlgen es útil ya que actúa como un "asistente" que facilita la creación del archivo WSDL.

Asimismo, en WebSphere se incluye la biblioteca Java uddi que se utiliza para la comunicación con un Registro de Servicio UDDI. A pesar de que se puede usar un registro UDDI privado, existe un registro UDDI, único y global, que puede utilizarse contactando con cualquiera de los "sitios de operación" de UDDL Estos sitios de operación son web services, y utilizan SOAP (y por lo tanto XML) para comunicarse. No obstante, UDDI es un protocolo sofisticado, por lo que ÍBM ha implementado uddi4j, que facilita la tarea de escribir una aplicación y se comunica con un sitio de operación para publicar y buscar web services. A pesar de que no se requiere para UDDI, WSDL es un complemento natural de éste al actuar como el protocolo con el que se puede describir un web service y uddi4j agrega herramientas que ayudan en el uso de WSDL para la descripción del web service a registrar.

4.2 e-Speak de Hewlett Packard (http://www.hp.com).

HP ha sido una de las empresas pioneras en el área de los web services incluso antes que fueran populares, inicio ofreciendo el producto llamado e-Speak, siendo este una plataforma abierta para el desarrollo y uso de servicios electrónicos. Un servicio electrónico opera de forma similar a un web service, Un servicio electrónico es aquel que se encuentra disponible en Internet y desarrolla una función específica. Se puede ubicar dinámicamente, invocar y crear por medio de otros servicios electrónicos. e-Speak nació como un proyecto de investigación en el año 1995 y en 1998 se inició el proyecto de software para transformarlo en un producto. En Mayo de 1999 salió al mercado el primer producto de e-Speak.

Desde entonces, HP cambio hacia estándares de base XML para web services como SOAP y UDDI, ya que cuentan con una mayor aceptación por parte de ¡a industria; HP ahora tiene un nuevo producto llamado HP Web Services Plataform, este se construyo de acuerdo a esos estándares.

Java ha desarrollado e-Speak. por lo que sus clientes necesitan utilizar su aplicación API Java (conocido como Interfaz E-Speak de Java, J-ESI) para la creación de servicios electrónicos y clientes. Asimismo, cuenta con mecanismos de base XML para la interacción con servicios e-Speak.

4.3 HP Web Services Platform.

La plataforma de web services de HP es una plataforma XML compuesta por los siguientes componentes:

- Mensajería
- Interacción
- Procesamiento

- Seguridad
- Transacción

Mensajería

La mensajería esta formada por oyentes y buzones de entrada para la recepción de mensajes. Un oyente es un componente estándar que se usa de base de recepción. Los oyentes se pueden configurar para el funcionamiento en una negociación de equilibrio de carga para la escalabilidad en algunos protocolos estándar de transporte como HTTP, HTTPS, SMTP y FTP.

Interacción

Ayuda a la transmisión de una petición a un componente de aplicación. Este puede utilizarse en envíos a aplicaciones (por ejemplo, peticiones SOAP RPC) como en conversaciones de mensajes multi-nodo (por ejemplo situaciones de negocio con ebXML). Además cuenta con un controlador de interacción que es compatible con conversaciones entre socios de negocios. La conversación es descrita usando el protocolo WSCL (Web Service Conversation Language). Es a este protocolo al que HP presenta como propuesta para la estandarización de la descripción de la conversación. Ya que este describe conversaciones de nivel empresarial y especifica documentos XML intercambiados, así como su secuencia.

Procesamiento

Este es compatible en tres niveles de modelos de procesamiento de aplicación:

- Solución de alto nivel y costo, por medio de un motor de proceso de gestión/sistema de flujo de trabajo en HP Process Manager.
- Solución de secuencia de comandos creada alrededor de Cocoon 2 XSP. Esta se podría usar para desarrollar la lógica de negocios de niveles intermedios.
- Solución J2EE de bajo nivel bajo y menor costo, en la que la lógica empresarial se desarrolla en una Java Bean/EJB.

Seguridad

Este componente se enfoca en los aspectos relacionados con la seguridad y cuenta con implementaciones de los estándares de seguridad.

- Firma Digital XML.
- Especificación de gestión clave XML o XKMS (XML Key Managment Specification).
- Encriptación XML.

Transacción

Se basa en los aspectos que surgen en la calidad del servicio QoS (Quality Of Service). Para encontrar mas detalles acerca de HP Web Services Platform usted puede visitar http://www.hp.com/go/webservices.

4.4 .NET y .NET Framework de Microsoft (http://www.microsoft.com).

La plataforma de implementación y desarrollo de web services que ofrece Microsoft es .NET Framework, .Net se anuncio por primera vez en Julio del 2000. La plataforma .NET se forma en base de bloques construidos con .NET Framework. La plataforma .NET consta de los siguientes componentes:

- Sistemas operativos en servidores, escritorios y dispositivos
- .Net Enterprises Servers
- .Net Building Block Services
- Visual Studio .Net
- .Net Framework

Sistemas operativos en servidores, escritorios y dispositivos.

Este nivel se conforma por los sistemas operativos de Microsoft para servidores, como el Servidor Windows 2000, sistemas operativos para ordenadores como Windows XP, Windows 2000 Professional o Windows ME y sistemas operativos para mecanismos como Windows CE.

.Net Enterprises Servers.

Este nivel se compone de los servidores .Net Enterprise Servers como BizTalk 2000 Server (EAl o Enterprise Application Integration y B2B Server), Commerce Server 2000 (servidor de soluciones de comercio electrónico), SQL Server 2000 (servidor de base de datos), Exchange Server 2000 (servidor de correo electrónico y colaboración), Application Server 2000 (gestión de sitios Web de alta disponibilidad y granjas de servidores). Content Management Server 2001 (gestión de contenido Web), Host Integraron Server 2000 (conexión a sistemas de legado), Internet Securíty and Acceleration Server 2000 (cortafuegos y servidor proxy), Mobile Información Server 2001 (entrada de conexión a aparatos móviles) o SharePoint Portal Server 2001 (gestión de documentos, búsqueda de empresas, y motor de portal).

Para encontrar mas detalles acerca de los servidores .NET Servers usted puede visitar http://www.microsoft.com/Servers

.Net Building Block Services

Es una colección de todos los web services que Microsoft espera poner a la venta. Estos pueden utilizarse por desarrolladores como funciones típicas para la construcción de aplicaciones que generen, evitando de este modo dedicar demasiado tiempo a diversos aspectos que tienen en común las aplicaciones, como por ejemplo como sucede con la autenticación o inicio de sesión. Uno de estos web services esta disponible en su anterior edición: el servicio Passport. Éste permite acceder únicamente a través de sitios o aplicaciones Web.

Visual Studio .Net

Es la herramienta para creación y desarrollo rápido de web services y otras aplicaciones (es la siguiente versión del Visual Studio 6 IDE o Integrated Development Environment).

.NET Framework j

El último elemento de la plataforma .NET es el marco .NET Framework. Este es la nueva plataforma de desarrollo por parte de Microsoft y contiene una nueva interface de programas Windows y API, incorporando tecnologías de componentes COM+, desarrollo Web ASP (Active Server Pages), seguridad, implementación sencilla y los protocolos que usan los web services como SOAP, WSDL y UDDI.

Componentes de La arquitectura .NET Framework:

- Web services
- Formularios Web
- Formularios Windows
- Información y clases de XML
- Clases de bases
- Common Language Runtime (CLR)

Los web services, formularios Web y Windows conforman la primera capa de .NET Framework, estos formularios se usan como interfaces para las aplicaciones, además los formularios web brindan clases para diseñar y desarrollar formularios de este mismo tipo (de base HTML operando en un navegador). Los formularios Windows brindan clases para diseñar y desarrollar aplicaciones propias de Windows Graphical User Interface (GUI). Ambos formularios proporcionan clases para la creación, implementación y uso de los web services por medio de los protocolos SOAP, WSDL y UDDI.

Las clases de bases son similares que los grupos de clases en ATL (Active Template Library) y MFC (Microsoft Foundation Clases) o algunas de las clases de base java y se incluyen funciones entrada/salida, de seguridad, comunicaciones de red, procesamiento de grupos.

La información y clases XML agregan funcionalidades a las clases bases ya que proporcionan un conjunto de clases para la manipulación y manejo de información en XML, incluye la información guardada en bases de datos por medio de SQL Server y ADO.NET. La compatibilidad relacionada con datos XML consiste en el acceso y manipulación de datos XML, búsquedas con la incorporación de Xpath y transformación de XML con XSLT.

Common Language Runtime, opera sobre del sistema operativo, realiza varias tareas como activar objetos, recolección de basura, manejo de memoria, ejecución de código, comprobación de seguridad de objetos, etc.

CLR permite usar todos los lenguajes que se puedan representar en Common Intermediate Language (CIL) o lenguaje intermedio de Microsoft en el que se haya compilado el código (esta funcionalidad es posible gracias a los tipos de datos en común como Long o Boolean). CLR sólo opera en sistemas operativos de Windows (a pesar de que existen indicaciones de que otras compañías han creado versiones de CIL para Linux).

4.5 Sun Microsystems y Sun ONE Studio (http://www.sun.com).

La plataforma para el desarrollo de web services por parte de Sun Microsystems es Sun ONE (Open Network Environment), es una plataforma creada en torno a la gama del servidor Sun-Netscape Alliance iPIanet y la nueva API XML para web services conocida como JAX Pack.

La plataforma Sun One esta compuesta por las siguientes capas:

- **Creación y ensamblaje del servicio:** incorpora las herramientas para el desarrollo de web services y aplicaciones. Incluye las herramientas de Forre Developer de Sun.

- **Reparto del servicio:** incluye tecnologías que ubican, conectan, agregan, presentan, comunican, personalizan, notifican y reparten contenidos. El servidor iPIanet Portal Server se ha creado para esta capa.

- **Aplicaciones y web services:** se diseño para la transformación de aplicaciones de productividad de transacciones de negocios tradicionales en web services colaborativos. Los productos de IPianet Commerce (iPIanet BuverXpert, BillerXpert, SellerXperr, MarketMaker) son los que colaboran en esta capa.

- **Contenedor del servicio:** incluye productos que facilitan la Implementación de web services y los pone a disposición del usuario. Esta capa incluye la Aplicación iPIanet y los Servidores Web.

- **Integración del servicio**: ayuda a conectar sistemas y aplicaciones actuales. Los producios que forman pare de esta capa incluyen el Servidor de Integración iPIanet y el iPIanet ECXpert. El servidor de integración permite el acceso a los datos y la lógica almacenados, como SAP o incluso aplicaciones personalizadas.

- **Identidad y política:** se ocupa de los perfiles de gestión de usuarios, usuarios, contexto, funciones y seguridad. Los producios de Gestión del Usuario iPIanet: Servidor del Directorio iPianet, Mera-Directorio, Enrutador de Acceso al Directorio. Administrador, Servidor de Proxy v el Sistema de certificado de Gestión iPIanet se ocupan de esta capa.

- **Plataforma:** hace referencia a la plataforma recomendada que el sistema operativo necesita para ejecutar estos web services. Sun recomienda la plataforma para el sistema operativo Sun y Sun Cluster 3.0, aunque las herramientas y los productos mencionados en las capas anteriores funcionan en la mayoría de las plataformas de sistema operativo UNIX y Windows.

Los componentes en esta pila se basan en las tecnologías estándar XML como SOAP, WSDL, UDDI, etc. y en el API de Java. Sun también ha definido un conjunto de API Java para XML, ambas API orientadas hacia documentos, como JAXP, JAXB, así como las orientadas hacia procedimientos como JAXM. JAXR y JAX-RPC. Para más detalles acerca de Sun 0' NE consulte http://WWW.SUn. com/sunone,' .

Agrupaciones que desarrollan estándares para Web services:

- W3C. World Wide Web consortium Fundada en 1994 con 500 miembros Es la principal desarrolladora de estándares para Web services.

- OASIS Organization of the Advancement Structured Information Standards, avances de estándares de la información estructurada trabajan con tecnologías XML consiste en 400 miembros que mejoran XML.

- IETF Internet Engineering Task Force, Fuerza de Tarea de Ingeniería en Internet. Trabaja sobre las arquitecturas y tecnologías de Internet.

- ISO Internacional Organization for Standardization tiene 140 países miembros se encarga de desarrollar estándares para mejorar el comercio internacional.

5. IMPLEMENTACIÓN DE WEB SERVICES.

Para implementar web services se necesita:

- El medio físico de comunicaciones, o llamada Intranet Administrativa.
- Uno o más registros UDDI de web services. Tiene que existir sincronización de servicios entre los distintos
- registros.
- La descripción del servicio mediante un documento en XML que se recoge en un archivo WSDL (lenguaje de descripción de web services)
- Un protocolo de intercambio de información en un entorno distribuido, que se soporte sobre http, SOAP (Simple Object Access Protocol).

Las dos implementaciones más extendidas del mercado son *Sun ONE y Microsoft .NET* que ofrecen todo una serie de productos, para implementar web servicies.

5.1 Protocolos que permiten implementar Web Services.

Hay un convenio generalizado que manifiesta que los web servicies se invocan en Internet por medio de protocolos estándar basados en XML. Hoy en día hay dos grandes tendencias: XML-RPC y SOAP. A la hora de programar un web service, hay que decidir qué protocolo usar, porque un protocolo es incompatible con el otro. De modo que si se programa un servicio web con XML-RPC, no podrá ser invocado desde un lenguaje de programación que trabaje con SOAP, como por ejemplo .Net de Microsoft.

Tanto SOAP (Protocolo de acceso a objetos simple, Simple Object Access Protocol) como XML-RPC son lenguajes de mensajería basada en XML, estandarizados por el consorcio W3C, que especifican todas las reglas necesarias para ubicar web services, integrarlos en aplicaciones y establecer la comunicación entre ellos.

Brevemente, la diferencia entre SOAP y XML-RPC es su complejidad. XML-RPC está diseñado para ser sencillo. SOAP por el contrario está creado con idea de dar un soporte completo y minucioso de todo tipo de web services. La curva de aprendizaje de XML-RPC es muy suave, por lo que un programador novato en este campo, puede conseguir resultados satisfactorios con poco esfuerzo. Con SOAP no pasa esto, pero a cambio, se dispone de más potencia. Por ejemplo, con XML-RPC no se puede elegir el conjunto de caracteres a utilizar en los web services que se desarrollen. En SOAP se puede elegir entre US-ASCII, UTF-8 y UTF-16.

Por el contrario, en favor de SOAP se puede añadir que el famoso web service de Google, está basado enteramente en SOAP, y que el proyecto Apache, dispone ya de un módulo de trabajo con SOAP, que se puede usar como una librería cliente para invocar servicios SOAP disponibles en cualquier lugar o como una herramienta del lado del servidor para implementar servicios SOAP accesibles.

5.2 Componentes de los web services.

Los principales bloques de construcción de los web services presentan tres aspectos paralelos:

a) Concepto de Ubicación.
b) Concepto de Descripción.
c) Concepto de Llamada.

Cada bloque consta de una serie de capas, en la figura siguiente se estudian estos bloques como entidades.

La arquitectura Web services posee cuatro capas:

- Descubrimiento(Discovery) UDDI –Universal Description, Discovery & Integration.
- Descripción(Description) se encarga de conocer qué hacen exactamente, y cómo se usan mediante WSDL – Web Services Description Language.
- Invocación de Llamadas(Invocation) SOAP –Simple Object Access Protocol.
- Transporte mediante el protocolo http.

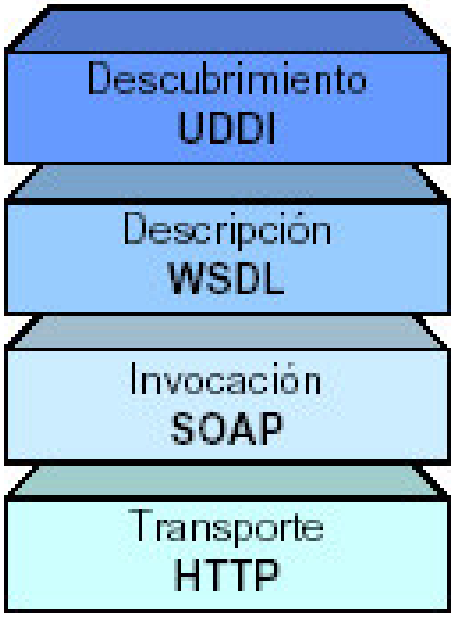


Figura 5. Capas de estructura de Web Services.

Los elementos constitutivos fundamentales de los web services son:

- **El Servicio (web service):** es la aplicación proporcionada para uso de los solicitantes. Su implementación se lleva a cabo en una estación accesible por red. Utiliza un lenguaje de descripción de servicios o funciones.

- **El Proveedor de Servicios:** es la plataforma responsable de proporcionar el acceso al servicio.

- **El Solicitante de Servicios:** es quien quiere hacer uso de cierta funcionalidad. Es el cliente el que busca y solicita el web service.

- **El Registro de Servicios**: es una base de datos que contiene descripciones de los web services que se encuentran disponibles, donde los Proveedores publican sus web services y los solicitantes buscan la meta información sobre los mismos.

Las operaciones básicas asociadas a los Web Services son:

1. **Publicar:** un servicio sólo puede ser utilizado si se da a conocer. Para ello el proveedor lo debe registrar.

2. **Descubrir:** El solicitante interactúa con el registro para encontrar el servicio que busca.

3. **Enlazar:** Finalmente el servicio ha de ser invocado. En la operación de enlace, el solicitante invoca al servicio en tiempo de ejecución utilizando la meta información que proporciona el registro.

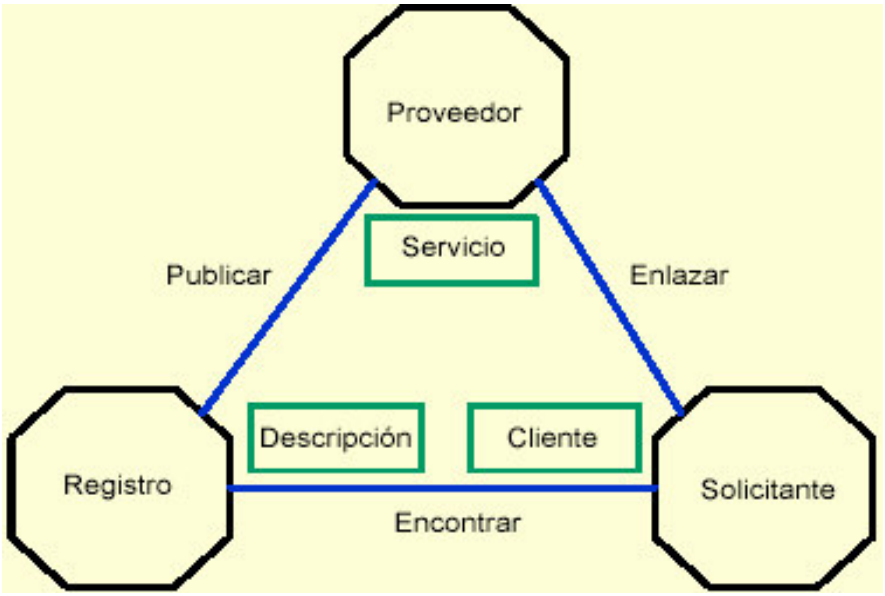


Figura 6. Estructura de Interacción.

6. LENGUAJES DE PROGRAMACIÓN.

6.1 C#.

Con .NET Framework, Microsoft presenta un nuevo lenguaje llamado C#. Microsoft ha creado y diseñado C # expresamente para el desarrollo de .NET, C# puede considerarse como una simplificación de C++. Sus raíces sintácticas provienen de C sin embargo, se han eliminado o simplificado muchos aspectos misteriosos, confusos y peligrosos, de C++.

Los programadores de C++ y Java no deberían tener dificultades para leer el código C#. Sin embargo, a continuación se citan algunas de las diferencias más importantes entre la sintaxis de C# y C++:

Sintaxis de acceso a los miembros simplificada. C++ utiliza el operador "->" para referirse a los miembros de la clase o de la estructura cuando un objeto está asignado a la memoria de bloque y "." para referirse a los miembros de la clase o estructura cuando el objeto está asignado a la memoria de pila. C# siempre utiliza la sintaxis "." para referirse a los miembros de la clase como datos y métodos. Por ejemplo:

myObj.SomeMethod( );
llama a el método SomeMethod( ) de la clase aportada por myObj.

Atributos del parámetro.

Los parámetros se pueden ser atribuir con las palabras clave ref. y out. Por defecto, si un parámetro no tiene el atributo ref u out, se transfiere por valor. Al crear un parámetro con ref, se transfiere por referencia. Al crear un parámetro con out, también se transfiere por referencia, sin embargo se asume que se definirá cuando el método sea invocado. Por ejemplo, en el método siguiente el atributo out se utiliza para indicar que el strNewPhoneNumber se transfiere por la referencia y se asume que no es válido en la invocación del método:

public bool CheckForSplit(string strPhoneNum, out string strNewPhoneNun);

Tipos del sistema de tipos comunes.

C# utiliza tipos compartidos que se implementan en el sistema de tipos comunes. Muchos de estos tipos se han estructurado según palabras clave de C++ que resultan familiares. Por ejemplo, el tipo System.Int32 del sistema de tipos comunes se corresponde con la palabra clave de C# int y se utiliza como el tipo int de C++. Las matrices y las cadenas son excepciones a este tipo de estructuración que tienen una sintaxis diferente a la que los programadores de C++ están acostumbrados. Para leer el código del ejemplo, será suficiente saber que string es un tipo de construcción interna y que las matrices deben declararse con paréntesis detrás del tipo. Tras la declaración, la matriz se asigna y dimensiona utilizando la palabra clave new. Por ejemplo:

```
int[ ] myArray;
myArray = new int [3] ;
```

Todo es un objeto.

C# permite tratarlo todo, incluidos los tipos nativos, como objetos. Hay incluso algunos métodos que se pueden llamar en todos los tipos. ToString( ) es uno de ellos. Lo cual quiere decir que se puede hacer lo siguiente:

```
int myInt = 7;
string  myString = myInt.ToString ( );
```

6.2 Visual Studio .NET 2003.

Visual Studio.NET 2003, esta constituido por cuatro lenguajes de programación:

1. Visual Basic .NET 2003.
2. Visual C# .NET 2003.
3. Visual C++ .NET 2003.
4. Visual J# .NET 2003.

Visual Studio .NET 2003, es la herramienta de Microsoft para crear e implementar software para la plataforma Microsoft .NET. Incluye una gama de funciones, desde modeladores que ayudan a componer visualmente las aplicaciones empresariales hasta la implementación de una aplicación en el más pequeño de los dispositivos. Utilizados por compañías de todos los tamaños en el mundo entero, Visual Studio .NET y la plataforma .NET Framework de Microsoft Windows proporcionan una herramienta, para diseñar, desarrollar, depurar e implementar aplicaciones para Microsoft Windows® y Web.

Los programadores pueden utilizar Visual Studio .NET para:

- Crear aplicaciones basadas en Windows
- Crear aplicaciones para Pocket PC
- Crear aplicaciones Web
- Crear aplicaciones Web para dispositivos móviles.
- Utilizar web services XML en cualquiera de las aplicaciones mencionadas.
- Evitar conflictos entre archivos .DLL.

Visual Studio .NET es un entorno de desarrollo creado exclusivamente para permitir la integración con web services. Al hacer posible que las aplicaciones compartan datos a través de Internet, los web services permiten a los programadores ensamblar aplicaciones a partir de código nuevo y existente, independientemente de la plataforma, el lenguaje de programación o el modelo de objetos.

Visual Studio .NET 2003 está disponible en las siguientes ediciones:

ENTERPRISE ARCHITECT	ENTERPRISE DEVELOPER	PROFESSIONAL
Visual Studio .NET Enterprise Architect Proporciona la capacidad total de Visual Studio .NET Enterprise Developer, más funciones adicionales para diseñar, especificar y comunicar arquitectura y funcionalidad de aplicaciones.	Visual Studio .NET Enterprise Developer Proporciona un entorno de desarrollo empresarial en equipo para crear con aplicaciones importantes orientadas a cualquier dispositivo y que se integren en cualquier plataforma.	Visual Studio .NET Professional. Permite a los programadores crear aplicaciones para Windows, Web, dispositivos Web móviles, Pocket PC y otros dispositivos incrustados que utilizan .NET Compact Framework.

Visual Studio .NET 2003 permite la creación y el uso de web services para crear e integrar aplicaciones empresariales en cualquier plataforma o dispositivo. Además incluye versiones para programadores de Windows Server 2003, SQL Server, Microsoft Exchange Server, Microsoft Commerce Server y Microsoft Host Integration Server, facilitando la creación y prueba de aplicaciones antes de implementarlas.

WSDK es un componente, que se incluye con Visual Studio .NET 2003, este permite a los programadores crear aplicaciones utilizando web services, proporcionando compatibilidad con varios aspectos principales de web services, entre ellos:

- Seguridad basada en mensajes (WS-Security).
- Enrutamiento e independencia de la topología de red (WS-Routing).
- Archivos adjuntos en mensajes SOAP (WS-Attatchments).

Visual C++ .NET 2003 ofrece la máxima capacidad, rendimiento, control y flexibilidad para crear aplicaciones que aprovechen Windows directamente. La nueva compatibilidad con el diseñador de formularios Windows Forms ofrece a los programadores de C++ una productividad sin precedentes al crear completas aplicaciones basadas en Windows. Con la programación basada en atributos y el servidor ATL, una extensión de las bibliotecas ATL, puede utilizar la sintaxis de C++ para crear aplicaciones Web ISAPI. Además, los programadores de C++ pueden orientar sus aplicaciones a Windows .NET Framework utilizando las Extensiones administradas para Visual C++.

En Visual Studio .NET 2003, los programadores que utilicen Visual C++ notarán también que la métrica principal para medir la portabilidad del código, compatibilidad con ANSI/ISO, se ha mejorado notablemente. El estándar ANSI/ISO C++ es el estándar generalmente aceptado para el lenguaje C++ y todos los proveedores de compiladores de C++ miden la compatibilidad con este estándar.

Visual C# .NET 2003 proporciona un lenguaje moderno e innovador, ideal para construir componentes orientados a objetos y marcos de trabajo empresariales. Basado en el estándar ECMA C#, Visual C# .NET ofrece mayor productividad a los programadores de C y C++. Gracias a la compatibilidad de primera clase con componentes que incluyen propiedades, métodos, indicadores, atributos, control de versiones y eventos, Visual C# .NET 2003 proporciona una base sólida para crear aplicaciones .NET.

Visual J# .NET 2003 es una herramienta de desarrollo para aquellos programadores de Java que deseen crear aplicaciones y servicios en Windows .NET Framework. Visual J# .NET 2003 se suma a más de 20 lenguajes anunciados anteriormente con su capacidad de orientar aplicaciones a Windows .NET Framework y servicios Web XML de primera clase.

Visual Basic .NET 2003.

Es ideal para programadores de Visual Basic 6.0, por usar sintaxis bastante similar, así como para programadores nuevos en el entorno de desarrollo de Microsoft .NET, Visual Basic .NET 2003 brinda diseñadores visuales, y un entorno de desarrollo integrado (IDE). Permite a los programadores utilizar web services que se ejecuten en cualquier plataforma y crearlos al igual que cualquier clase en Visual Basic 6.0.

Posee interoperabilidad COM integrada a Windows .NET Framework, los programadores pueden seguir utilizando la mayor parte de los componentes que han usado durante años. La interoperabilidad COM proporciona comunicación bidireccional entre las aplicaciones escritas con Visual Basic 6.0 y las escritas con Visual Basic .NET, sin necesidad de escribir de nuevo el código. Visual Basic 6.0 y Visual Basic .NET pueden residir sin problemas en el mismo equipo, lo que facilita la transición.

Visual C#.NET 2003.

Es una herramienta y un lenguaje de programación que permite generar software conectado a .NET para Microsoft Windows®, Web y una amplia gama de servicios. Usa sintaxis similar a la de C++, y un entorno de desarrollo integrado (IDE) capaz de crear soluciones informáticas para una varias plataformas y dispositivos.

Visual C#.NET está basado en C++ y resulta muy familiar a los programadores que han trabajado con C++ y Java. Es un lenguaje de programación intuitivo orientado a objetos que ofrece un sistema de tipos unificados, código "no seguro" que permite un control al programador y construcciones de lenguaje fáciles de entender para la mayoría de los programadores. Usa comentarios en formato XML, los programadores de C# pueden elaborar documentación sobre el código fuente.

Posee un modelo de herencia que permite a los programadores volver a utilizar el código de cualquier lenguaje de programación compatible con .NET.

Visual C#.NET 2003, permite crear web services que integren procesos empresariales y los pongan a disposición de aplicaciones que se ejecuten en cualquier plataforma. Se puede incorporar fácilmente cualquier número de web services, que estén catalogados y disponibles en cualquier directorio UDDI (integración, descubrimiento y descripción universal) independiente.

Visual C++ .NET 2003

Microsoft Visual C++ .NET 2003 permite crear aplicaciones basadas en Microsoft Windows® y conectadas a Microsoft .NET, aplicaciones Web dinámicas y web services mediante el lenguaje de programación C++.

Este entorno de desarrollo incluye compiladores con compatibilidad a las normas ISO, implementación de bibliotecas STL (Standard Template Library), biblioteca ATL (Active Template Library) estándar, bibliotecas MFC (Microsoft Foundation Class) y un entorno de desarrollo integrado que permite editar y depurar el código fuente.
Algunas de las funciones de Visual C++ .NET 2003:

- Capacidad de utilizar y ampliar Microsoft Windows .NET Framework.
- Diseñadores visuales para crear Formularios Windows y componentes.
- Depurador y varios compiladores de la industria que ofrecen opciones para la generación de código en plataformas de 32 y 64 bits.

Se puede incorporar funciones de Windows .NET Framework, incluida la recolección de elementos no utilizados, los atributos y los subprocesos, en aplicaciones de C++. Visual C++ ofrece una función exclusiva que le permite llamar a código de C++ y componentes de ActiveX® utilizando la tecnología de interoperabilidad en .NET.

El compilador de Visual C++ .NET 2003 es compatible con la definición del lenguaje C++ de ISO y permite crear fuentes de bibliotecas y código de C++. Entre las nuevas características, destacan varias funciones de plantillas definidas por la norma ISO, tales como Partial Template Specialization (Especialización parcial de plantillas) y Partial Ordering of Function (Ordenamiento parcial de funciones). Estas funciones permiten escribir menos código y reciclable. Las bibliotecas de C++, incluidas Loki, Boost y Blitz, son compilables con Visual C++ .NET 2003, proporcionando a los programadores de Visual C++ la capacidad de incorporar estas fuentes a sus aplicaciones.

Visual C++ posibilita incorporar las funciones nuevas a las aplicaciones mediante bibliotecas. Visual C++ ofrece una variedad de bibliotecas, como por ejemplo:

- Implementación STL totalmente compatible con las normas ISO (que proporciona algoritmos y clases de contenedores genéricos).
- ATL y MFC (ahora actualizadas para Windows XP y Windows Server 2003).
- Servidor ATL (que permite web services y contenido Web dinámico no administrado).
- La plataforma Windows .NET Framework está completamente habilitada para los programadores de Visual C++, que pueden utilizar y expandir con esta biblioteca a partir de fuentes de C++.

Visual J# .NET 2003.

Es una herramienta para programadores que usen lenguaje Java y que desean crear aplicaciones y web services en Microsoft Windows .NET Framework. Está dirigido a la versión 1.1 de Windows .NET Framework, está integrado con Visual Studio .NET 2003 y proporciona compatibilidad adicional para generar aplicaciones Web para dispositivos móviles.

Algunos aspectos que Visual J# .NET 2003 proporciona a los programadores de Java:

Una forma fácil de aprovechar las ventajas de Windows .NET Framework, como por ejemplo, la compatibilidad nativa con los web services.

a) Capacidad de crear una variedad de aplicaciones utilizando un modelo de programación unificado. Visual J# .NET 2003 permite generar aplicaciones basadas en Microsoft Windows®, aplicaciones Web e incluso aplicaciones Web para dispositivos móviles que se pueden implementar de forma dinámica en más de 200 tipos de dispositivos móviles.

b) Oportunidad para los clientes de Microsoft Visual J++® y otros programadores de Java de aprovechar la inversión actual en conocimientos y código, así como el uso completo del entorno de desarrollo de Microsoft tanto ahora como en el futuro.

Visual J# .NET 2003 incluye tecnología que permite a los usuarios migrar sus aplicaciones Java a Windows .NET Framework. Las aplicaciones existentes creadas mediante Visual J++ pueden modificarse fácilmente para ejecutarse en Windows .NET Framework, interactuar con otros lenguajes y aplicaciones conectados a Microsoft .NET e incorporar funcionalidad .NET, como por ejemplo, ASP.NET, ADO.NET y Formularios Windows. También es posible utilizar Visual J# .NET 2003 para generar aplicaciones conectadas a .NET totalmente nuevas.

Visual J# .NET 2003 incluye herramientas para actualizar automáticamente y convertir los proyectos y soluciones existentes de Visual J++ 6.0 al formato de Visual Studio .NET 2003. Estas herramientas garantizan a los programadores actuales de Visual J++ 6.0 una transición sencilla a Visual J# .NET 2003 y la creación de aplicaciones y componentes conectados a .NET.

Visual J# .NET 2003 no es una herramienta diseñada para crear aplicaciones que se puedan ejecutar en una máquina Java virtual. Las aplicaciones y servicios generados con Visual J# .NET 2003 sólo se podrán ejecutar en Windows .NET Framework y no en una máquina Java virtual. Microsoft ha diseñado y creado Visual J# .NET 2003 de forma independiente. Esta herramienta no ha sido certificada ni aprobada por Sun Microsystems, Inc.

6.3 Sun ONE Studio 5, Standard Edition

Sun ONE Studio 5, Standard Edition es la siguiente versión de la familia de productos Sun ONE Studio for Java. Ha sido creado como el entorno de desarrollo de la pila de productos Sun ONE, específicamente para Sun ONE Application Server y los desarrolladores encargados de la creación de aplicaciones y web services en n-niveles. Contiene un entorno de desarrollo integrado (IDE) potente e intuitivo para Java, proporciona un conjunto de funciones y características que permiten el desarrollo de software desde aplicaciones de escritorio hasta aplicaciones empresariales basadas en estándares y web services.

Características:

- Preconfigurado con Sun ONE Application Server 7; menores ciclos iterativos de desarrollo gracias a herramientas integradas, Respuesta y rendimiento mejorados significativamente.
- Centrado en Java 2 Platform, Enterprise JavaBean 2.0 Workshop; rápido desarrollo de J2EE 1.3 Enterprise Applications.
- Desarrollo, prueba y publicación de web services.
- Acceso a bases de datos con desarrollo de JDBC simplificado.
- Compatibilidad con XML; creación y depuración de documentos XML, XML Schema y DTD utilizando las herramientas integradas.
- Interoperabilidad para el Desarrollo Orientado de Objetos JNDI/CORBA/RMI (provee módulos opcionales, actualizables desde el centro de Actualizaciones de la compañía).
- Configuración de compatibilidad sin fisuras con web services ampliados para el Sun ONE Application Server 7.
- Servidor web de registro UDDI.
- Compatibilidad completa con JAX-RPC, incluidos: encabezados, archivos adjuntos, sesiones de filtro dinámico de paquetes y seguridad.
- Integración inalámbrica mejorada para dispositivos con SOAP activado.

- Módulo de interfaz de compatibilidad integrado para permitir la resolución y notificación de problemas eficaz.

- Nuevas funciones y características de Netbeans 3.4 1

- Cómodo acceso a la información del servicio técnico gracias al nuevo módulo Bug Submitter

Sun ONE Web Services Platform Developer Edition.

Sun™ ONE Web Services Platform Developer Edition proporciona un entorno de prueba y desarrollo de web services integrados para crear pruebas de concepto que se integren en varios sistemas de información. El producto proporciona herramientas integradas y servidores de software intermedio y aprovecha los estándares más recientes de Java, XML y web services. Las herramientas de Sun ONE Web Services Platform Developer Edition se han creado sobre la plataforma NetBeans para maximizar la productividad de los desarrolladores. La salida de estas herramientas se puede desplegar en diferentes servidores Sun ONE que permiten que los negocios simulen fácilmente un entorno de producción para una comercialización rápida.

Características:

- Integrated Installer permite una instalación optimizada para los desarrolladores de toda la plataforma de web services.

- Desarrolla, ensambla y despliega aplicaciones de web services basados en componentes y estándares Java 2 Platform, Enterprise Edition (J2EE™) para Sun ONE Application Server y otras plataformas líderes en el mercado. Compatible con el desarrollo para la especificación J2EE 1.3-

- Sun ONE Application Framework (JATO) facilita el proceso de desarrollo de aplicaciones Web, que son seguras, escalables y mantenibles. Se trata de una infraestructura de aplicaciones Web Java™ 2 Platform, Enterprise Edition (J2EE™) dirigido al desarrollo de aplicaciones Web de empresa.

- Sun ONE Connector Builder; permite integrar los sistemas de información de empresa existentes, en una infraestructura de web services, resulta más fácil realizar la migración de las organizaciones de TI a una arquitectura basada en servicios-

- Sun ONE Portlet Builder permite el desarrollo de portales de empresa personalizados-

- Sun ONE Portal Server proporciona una solución de portal activado para identidades para gestionar y administrar usuarios, normas y servicios de abastecimiento, además de proporcionar un inicio de sesión único, administración delegada y web services-

- Sun ONE Integration Server EAI Edition proporciona integración de Sun ONE para web services a través de mensajería SOAP, que puede exportar una definición de servicio por WSDL a UDDI u otros clientes SOAP-

7. ESTÁNDARES Y TECNOLOGÍAS ASOCIADAS.

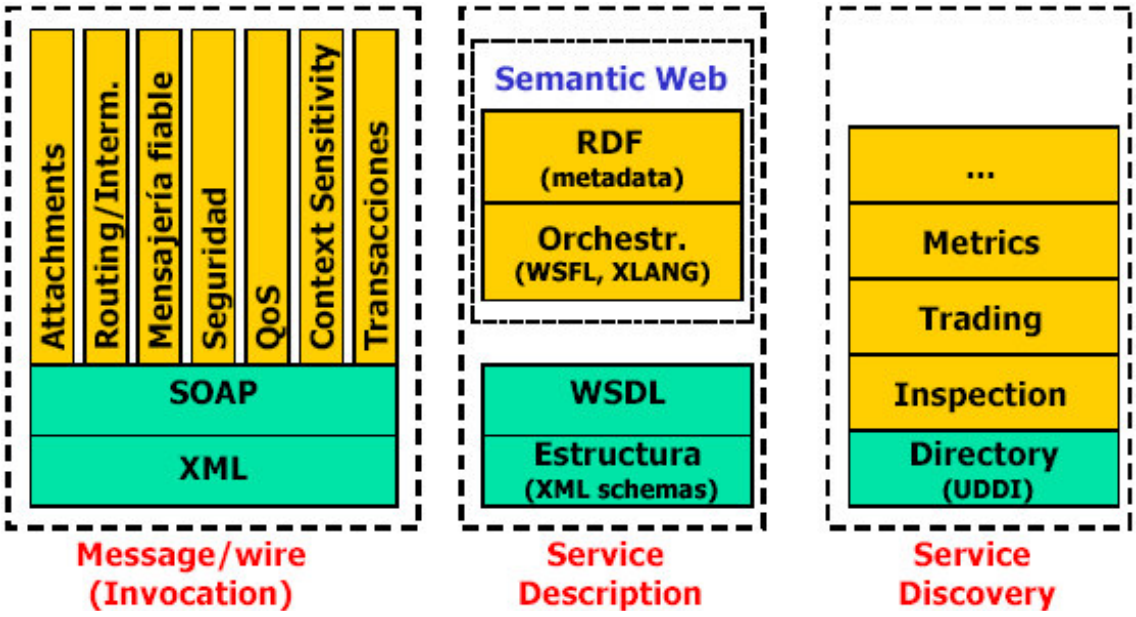


Figura 7. Estándares y Tecnologías asociadas a Web Services.

Los web services, se basan en los siguientes estándares de Internet:

1. SOAP
2. WSDL
3. UDDI
4. XML

7.1 SOAP (Simple Object access Protocol).

Es el protocolo de comunicaciones que usan los web services. Define los mecanismos por los cuales un web service, es invocado y cómo devuelve los datos. Los clientes SOAP llaman a los servidores SOAP, pasando objetos en formato XML. SOAP es un estándar definido por W3C y creado a partir de un grupo de trabajo integrado por varias asociaciones. Se encarga de indicar como deben ser los mensajes que circulan entre las distintas aplicaciones, se construyo para facilitar la llamada a funciones remotas.

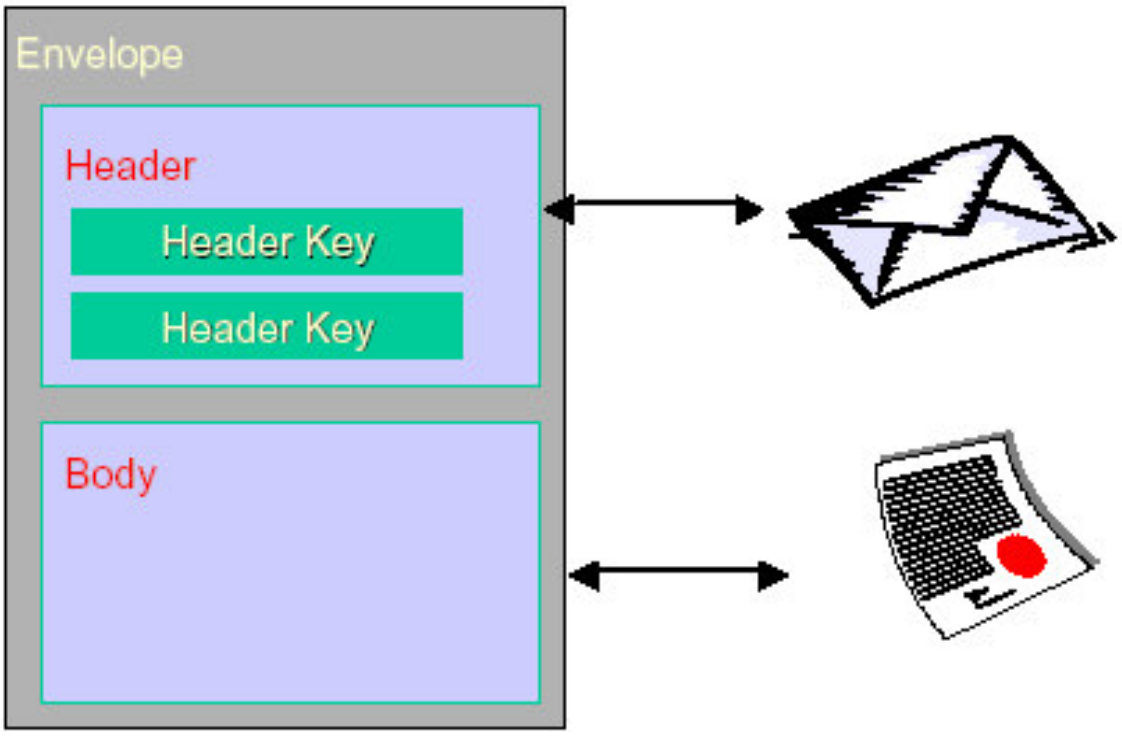


Figura 8. Modelo Soap

SOAP como protocolo de mensajería entre ordenadores.

SOAP es un protocolo de cable sencillo basado en XML, se diseño para conexión entre ordenadores independientes de subsistemas operativos, lenguajes de programación o modelos de objetos utilizados (e incluso con la carencia total del modelo de objeto). A pesar de que su nombre pueda parecer que requiere del uso de determinados objetos, SOAP especifica el formato del mensaje que accede e invoca a los objetos, en vez de especificar los objetos en si.

SOAP es extensible para la comunicación de ordenador, que emplea por igual los estándares actuales de Internet: XML para el formato de mensajes, http y otros protocolos de Internet para el transporte de mensajes.

SOAP especifica el formato del mensaje de la comunicación entre ordenadores. Los Mensajes de SOAP son básicamente transmisiones de dirección única emisor-receptor.

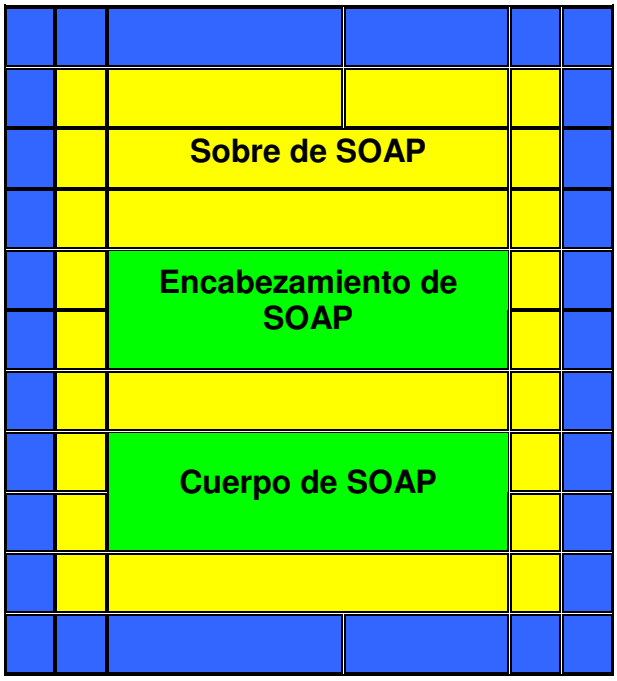


Figura 9. Estructura de SOAP.

En mayo del año 2000, el W3C (World Wide Web Consortium http://www.w3c.org/) reconoció la propuesta de SOAP presentada de forma conjunta por un conglomerado de empresas (Ari a Inc., CommerceOne Inc., Compaq Computer Corp., Micro$oft Corp., Development Corp., IBM Corp, Hewlett-Packard Co., etc). Siendo desarrollado en base a un estándar abierto.

Ventajas de SOAP:

- ♣ Fácil de implementar y usar.
- ♣ Utiliza los mismos estándares de la web: HTTP para la
- ♣ comunicación, protocolos de autenticación y
- ♣ encriptación.
- ♣ Atraviesa "firewalls" y routers ya que piensan que es
- ♣ una comunicación HTTP.
- ♣ Tanto los datos como las funciones se describen en
- ♣ XML.
- ♣ No depende del sistema operativo y procesador
- ♣ Describe un estándar WSDL que define los objetos y métodos disponibles a través de
- ♣ páginas XML disponibles en la web.
- ♣ Se puede implementar en casi cualquier lenguaje de programación.
- ♣ Dos formatos de mensajes:
- ♣ Un mensaje que se envía desde la aplicación cliente a la aplicación servidor. Y un mensaje que se envía desde la aplicación servidor a la aplicación cliente.

- SOAP provee un mecanismo independiente del protocolo, que se use para la comunicación, para asociar el encabezado al cuerpo del mensaje: a esto se le conoce como "envelope".
- SOAP no impone ninguna restricción acerca de cómo debe formatearse el cuerpo del mensaje.

Funcionamiento de SOAP.

En principio una petición/respuesta de SOAP puede usar cualquier protocolo de transporte (HTTP, SMTP, etc.). Las cabeceras difieren de protocolo a protocolo, pero el contenido de SOAP es el mismo. La petición de SOAP en XML consiste de tres partes:

1. **Envelope -** El sobre de SOAP define un marco de referencia general para expresar que es lo que contiene el mensaje, quién debe recibirlo, y si es opcional u obligatorio.

2. **Codificación -** Las reglas de codificación de SOAP definen un mecanismo de estandarización que puede ser utilizado para intercambiar instancias de tipos de datos definidos por la aplicación.

3. **Estándar -** La representación RPC de SOAP define una convención que puede ser utilizada para representar los llamados y respuestas de procedimientos remotos.

Un mensaje SOAP/HTTP es aceptado típicamente por un Java servlet que se ejecuta en el servidor web Apache (Tomcat). En su momento el servlet revisa si hay alguna petición de SOAP y la transfiere al motor SOAP de Apache. El motor analiza el contenido de texto de XML y utiliza la dirección destino del servicio Web para buscar e invocar el servicio y brindar el resultado.

La respuesta SOAP/HTTP se devuelve como un documento XML dentro de una respuesta HTTP. El documento XML está estructurado como cualquier respuesta HTTP excepto que el Cuerpo contiene incrustado el resultado del método ejecutado.

Dado que SOAP utiliza XML para codificar mensajes, es relativamente sencillo procesar los mensajes en cada paso del proceso. Además, la facilidad de depuración de mensajes SOAP permite la convergencia rápida de las diversas implementaciones de SOAP, la cual es importante en la interoperabilidad a gran escala.

Internet fue diseñada para que múltiples sistemas se pudiesen comunicarse entre si. Lo ideal es que una aplicación pueda usar componentes de otras aplicaciones, e incluso desarrollados en otros lenguajes. Eso se consigue actualmente mediante CORBA y DCOM. No obstante un desarrollo con estos estándares no es fácil de lograr y, sobre todo, la comunicación entre módulos situados remotamente no suele funcionar. El más típico es que haya un cortafuego por el medio. Entonces el tráfico de este tipo no puede pasar. Como SOAP trabaja sobre HTTP, este problema se minimiza: HTTP es un protocolo opera muy bien con los "cortafuegos".

Para asegurarse que la comunicación SOAP sea segura se debe utilizar el protocolo HTTPS en lugar de HTTP, de ésta manera SSL nos brinda su protección. Se puede resumir que SOAP es un protocolo ligero, simple y extensible. Se usa para comunicación entre aplicaciones dentro y fuera de la misma red. SOAP está diseñado para comunicarse vía HTTP, es decir a través del puerto 80 lo que facilita su tránsito. El estándar no está ligado a ninguna tecnología de componentes propietarios ni lenguaje de programación específico. El formato está basado en XML y coordinado por W3C. La mayoría de las plataformas ya disponen de implementaciones de SOAP.

Mensajes SOAP.

Un mensaje SOAP es un documento XML que consta de los siguientes elementos:

- **Envelope (sobre)** que define el contenido del mensaje.
- **Header (cabecera)** que es opcional y que contiene meta información referente al mensaje.
- **Body (cuerpo)** que contiene la información de la llamada y de la respuesta.

Ejemplo de mensaje SOAP petición:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/encoding/">
        <SOAP-ENV:Header>
        </SOAP-ENV:Header>
        <SOAP-ENV:Body>
                <catalogo:buscaIsbn xmlns:catalogo="http://catalogo.org/cat">
                        <catalogo:isbn>
                                84-4553-3334-2X
                        </catalogo:isbn>
                </catalogo:buscaIsbn>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Ejemplo de mensaje SOAP respuesta:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAPENV=”
http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Header>
        </SOAP-ENV:Header>
        <SOAP-ENV:Body>
        <catalogo:buscaIsbnResponse
                xmlns:catalogo="http://catalogo.org/cat">
```

```
        <catalogo:titulo>
                Catalogar materiales especiales
        </catalogo:titulo>
        <catalogo:autor>Marta de Juanes</catalogo:autor>
        </catalogo:buscaIsbnResponse>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

7.2 WSDL (Web Service Definition Language).

Describe la interfaz externa de un web service y cómo utilizarlo. Se puede definir un archivo WSDL como un documento XML que describe un conjunto de mensajes SOAP y la forma en que éstos interactúan.

Puesto que la notación que utiliza WSDL es XML significa que es un idioma de programación neutral, basado en estándares (W3C) y que puede utilizarse desde una gran variedad de plataformas y lenguajes.

Además de describir el contenido WSDL define todos los elementos necesarios para utilizar el servicio (interfaz, lugar en el que está disponible, protocolo de comunicaciones...), El Lenguaje de Descripción de web services (WSDL) es el equivalente de un resumen en XML, describiendo los web services, donde se ubican, y cómo se pueden invocar.

Especificación XML para la formación del documento de descripción de un web service. Identifica los métodos, funciones y parámetros necesarios para invocar un determinado servicio. Así, un usuario puede crear una aplicación cliente que comunica con el web service.

Para que un cliente se pueda enlazar o comunicar con un web servicie es necesario algo más que la definición formal del interfaz. Se deben especificar toda una serie de metainformaciones del servicio: dirección de red, los protocolos, los requisitos de seguridad, etc. WDSL (Web Services Definition Language) cumple con estas necesidades. Al igual que para cualquier componente de un modelo de computación distribuida ,se necesita un lenguaje de definición del interfaz (IDL). WSDL establece todos los requerimientos necesarios. WSDL proporciona solo descripciones funcionales del servicio (detalles de enlace, prototipo, protocolo).

WDSL describe lo siguiente:

- Información de interfaz que describe todas las funciones públicamente accesibles.
- Información de los tipos de datos para todos los mensajes de petición y respuesta.
- Información acerca del protocolo de transporte a utilizar.
- Información para localizar un servicio específico Representa un contrato entre el peticionario del servicio y el servicio proveedor.
- El cliente puede localizar un servicio web y llamar a alguna de sus funciones.

El lenguaje de descripción de web services WSDL es un lenguaje XML que contiene información acerca de la interfaz, semántica, y administración de una llamada a un web service.

Una vez se encuentre desarrollado el web service, se publica su descripción y se construye un enlazamiento o puntero en un depósito UDDI (Universal Description, Discovery and Integration) para que los usuarios potenciales lo puedan utilizar.

Cuando alguien piensa en utilizar este web service, solicitan el archivo WSDL para conocer la ubicación del servicio, llamado de funciones, y cómo acceder al web service. Luego utilizan la información en el archivo WSDL para construir una petición SOAP (Simple Object Access Protocol) y enviarla hacia el proveedor de servicio.

Una de las ideas principales detrás de los web services, es que las aplicaciones futuras estarán conformadas de una colección de servicios habilitados en la red. Mientras haya dos servicios equivalentes que se anuncien a la red de una forma estandarizada, en teoría un cliente podría seleccionar uno de ellos en base a criterios establecidos de antemano como precio o rendimiento.

Los documentos WSDL consisten de siguientes elementos:

Definitions - El elemento <definitions> contiene la definición de uno o más servicios. En la mayoría de los casos, un archivo WSDL define un servicio únicamente. Seguido de la etiqueta de definición se encontrarán declaraciones de algunos atributos. Dentro de la etiqueta <definitions> las siguientes secciones conceptuales:

- <message> y <portType>, describe qué operaciones provee el servicio.
- <binding>, describe cómo se invocan las operaciones.
- <service>, describe dónde se ubica el servicio.
- <documentation>, cualquier elemento WSDL puede incluir información del servicio para el usuario.

Resumiendo, WSDL es equivalente a la URL en los servicios XML-RPC. Es la dirección de Internet donde hay que invocar el método o el proceso.

El WSDL es un borrador (no está completamente aprobado por el W3C), aunque ya se usa en fase de explotación de forma extensiva.

El WSDL aunque actualmente no está completamente aprobado por W3C está en fase de uso extenso por el público general ya que es casi la única tecnología que puede dar respuestas seguras a las preguntas que tienen los proveedores acerca de los servicios de la empresa:

- ¿Qué servicios on line ofrece la empresa?
- ¿Cómo usar estos servicios?
- ¿Qué datos son necesarios para utilizar los servicios?
- ¿Cómo se debe proporcionar los datos necesarios?
- ¿En qué formato los servicios contestarán al usuario?

Afortunadamente, WSDL proporciona el mecanismo para llevar a cabo todas estas tareas.

A continuación se examinara el proceso de escribir un archivo WSDL para un web service con finalidad de exponer los web services que existen al público general. Considere el siguiente caso práctico para ayudar a entender como WSDL puede mejorar la integración con proveedores y partners comerciales:

Se tiene un servicio tradicional existente, una página web y se desea exponer las funciones on line. Se dispone de una interfaz WSDL y se deseas implantar lógica del web service de acuerdo con los elementos que ya se ha decidido exponer.

La creación de WSDL se divide en cuatro pasos:

Paso 1: La Interfaz de Servicios.

Como proyecto de ejemplo, se creara la interfaz de servicios para un proyecto ficticio de información de productos de ferretería al retail (y se llamara este servicio ProductService). El proyecto se encarga de servir precios a los asociados de diferentes productos que tiene en la plataforma, de modo que el almacenamiento de datos en el sistema de administración del web service del proyecto contiene una tabla con dos columnas, producto y precio. Se simplifica el ejemplo para centrar atención en WSDL. El servicio dispone de dos métodos, que se expondrán mediante WSDL:

- ♣ getProductListing () – Este método proporciona una matriz de series, cada una de las cuales representa el identificador de producto.

- ♣ getProductPrice (productID) – Este método toma el identificador de producto y devuelve su precio.

WSDL llama a estos métodos - operaciones. A continuación, empieza la creación del archivo de la interfaz WSDL.

Los archivos WSDL tienen como elemento principal <definitions>, en las que se debe proporcionar una descripción completa de los servicios. Antes que nada, se debe incluir diferentes declaraciones de <namespace> en el elemento de <definitions>. Las tres declaraciones de <namespace> externos que debe definir son WSDL, SOAP y XSD (definición de esquema XML).

Existe otro espacio de nombres, <targetNamespace>, que hace referencia a ProductService (contiene todos los nombres de elementos y atributos que se definirán especialmente para ProductService). No obstante, WSDL es el principal espacio de nombres que se utiliza en la mayor parte de tareas de creación de WSDL.

WSDL ampliamente utiliza el concepto de espacios de nombres (namespaces). WSDL es una implantación de esta idea, puesto que los espacios de nombres proporcionan una flexibilidad ilimitada, y esto es exactamente lo que necesita un formato portátil para el intercambio de datos electrónicos.

El elemento <definitions> contiene uno o más elementos <portType>, cada uno de los cuales es en realidad un conjunto de operaciones que se desea exponer. De forma alternativa, se puede considerar un solo elemento portType como una agrupación lógica de métodos en clases.

Ejemplo, si la solución para la gestión de una cadena de suministros requiere interactuar tanto con los clientes como con los proveedores, lo más probable es que se defina de forma separada las funciones para interactuar con ellos; es decir, se definirá un elemento portType para los clientes y otro para los proveedores. Se debe llamar a cada elemento portType un servicio, de modo que el archivo WSDL completo sea un conjunto de servicios.

Se debe proporcionar un nombre para cada servicio, lógicamente. En este caso, sólo existe un servicio (por tanto, sólo un elemento <portType>). Se debe utilizar el atributo name de este elemento portType para asignar un nombre a nuestro servicio de venta de productos de telefonía móvil.

Dentro de cada servicio puede haber varios métodos u operaciones, a las que WSDL hace referencia mediante elementos <operation>. La aplicación de ejemplo tiene dos métodos de exposición, getProductListing y getProductPrice. Por lo tanto, se debe proporcionar dos elementos <operation>, cada uno de los cuales tendrá un nombre distinto. Se ha utilizado el atributo name del elemento <operation> para denominar cada operación.

Ejemplo: Paso 1 La Interfaz de Servicios.

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions  name="ProductService"
  targetNamespace="http://activalink.net/ProductService-interface"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://activalink.net/ProductService"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
      <portType name="ProductService_port">
            <operation name="getProductListing">
                  .....
                  .....
            </operation>
            <operation name="getProductPrice">
```

```
            .....
            .....
        </operation>
    </portType>
</definitions>
```

Paso 2 : Creación de Parámetros .

Una vez definidas las operaciones (o métodos), se debe especificar los parámetros que enviará y los parámetros que devolverá. En términos de WSDL, todos los parámetros se denominan "mensajes". Resulta útil pensar que se envían mensajes y que, como resultado, se reciban mensajes. Las llamadas de métodos son operaciones que se llevan a cabo para preparar "mensajes" de vuelta en respuesta a mensajes de entrada.

Se recuerda del primer paso que éste tiene dos operaciones para exponer. La primera operación, getProductListing, no admite ningún parámetro y devuelve una matriz de identificadores de productos. Por lo tanto, se debe definir un elemento <message> que contenga una matriz de identificadores.

Se observan los diferentes elementos de <message> en el siguiente código de ejemplo. El primero de estos elementos tiene un atributo de nombre igual a ProductListing (un nombre lógico para este mensaje) y un elemento <part> con los identificadores de productos, que significa que ProductListing es un mensaje de una parte, en la que el nombre de la única parte presente es "productID". Un mensaje puede tener cualquier número de partes, siempre que no se olvide asignarle nombres diferentes para que la identificación sea exclusiva.

Se ha incluido otro atributo del elemento <part>, que es type. Se considera este atributo "type" como tipos de datos en C/C++. Se Ha especificado el tipo de datos productIDs como tns:Vector. Recordando que se habrá especificado unos cuantos espacios de nombres en el elemento <definitions> raíz, uno de los cuales era TNS. Esto hace referencia al espacio de nombres ProductService. Significa que se puede crear nuestro propio espacio de nombres mientras se diseña la interfaz WSDL.

Ejemplo una matriz de productos devueltas por la operación getProductListing. WSDL utiliza varios tipos de datos primarios que define la definición de esquema XML (XSD), por ejemplo entero, flotante, largo, corto, byte, serie, Booleano, etc.) y permite utilizarlos directamente o para crear tipos de datos complejos basados en estos tipos primarios antes de utilizarlos en mensajes. Por este motivo necesita definir su propio espacio de nombres cuando haga referencia a tipos de datos complejos. En este caso, debe crear un tipo de datos complejo para una matriz de productos.

Por lo que respecta al cómo, se utiliza XSD para crear el espacio de nombres. Para este fin se utilizo el elemento xsd:complexType con el elemento <types> para definir un tipo de datos denominado Vector. Vector utiliza dos tipos de datos primarios, serie (datos del elemento) y entero (número de elementos). Por ello, Vector pasa a ser parte del espacio de nombres y puede hacerse referencia a él mediante el alias TNS.

De forma similar, se definieron los otros dos mensajes, Product y ProductPrice, en el siguiente código de ejemplo. Estos dos mensajes utilizan sólo tipos de datos primarios de identificador de producto del espacio de nombres XSD y, por consiguiente, no es necesario definir más tipos de datos complejos para utilizarlos.

Es posible que se haya observado que, durante la creación de los elementos de <message>, no se especifico si estos mensajes eran parámetros de entrada o valores devueltos. Esta tarea se tratará en el elemento <operation>, dentro del elemento <portType>. Así, como puede ver en el siguiente código de ejemplo , se añado los elementos de <input> y <output> a cada una de las dos operaciones. Cada elemento de entrada hace referencia a un mensaje por su nombre y lo trata como un parámetro que el usuario proporcionará cuando invoque esta operación. De forma similar, cada elemento de <output> hace referencia a un mensaje; trata el mensaje como el valor devuelto de la llamada de la operación.

Ejemplo: Paso 2 Creación de Parámetros.

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions  name="ProductService"
  targetNamespace="http://activalink.net/ProductService-interface"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://activalink.net/ProductService"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
      <types>
            <xsd:schema targetNamespace=" http://activalink.net/ProductService"
              xmlns="http://www.w3.org/1999/XMLSchema/">
                <xsd:complexType name="Vector">
                      <xsd:element name="elementData" type="xsd:String" />
                      <xsd:element name="elementCount" type="xsd:int" />
                </xsd:complexType>
             </xsd:schema>
      </types>
      <message name="ProductListing">
            <part name="productID" type="tns:Vector">
      </message>
      <message name="Product">
            <part name="productID" type="xsd:String">
      </message>
      <message name="ProductPrice">
            <part name="price" type="xsd:String">
      </message>
      <portType name="ProductService_port">
            <operation name="getProductListing">
                  <output message="ProductListing"/>
            </operation>
            <operation name="getProductPrice">
```

```
            <Input message="Product"/>
            <output message="ProductPrice"/>
        </operation>
    </portType>
</definitions>
```

Paso 3 : Mensajes y Transporte.

Una vez se Hayan definido las operaciones y los mensajes de forma abstracta, sin preocupase de los detalles de implantación. De hecho, la tarea de WSDL consiste en definir o describir web services y, a continuación, proporcionar una referencia para una infraestructura externa con el fin de definir el modo en que el usuario de WSDL accederá a la implantación de estos servicios. Puede considerar esta infraestructura como un enlace entre las definiciones abstractas de WSDL y su implantación.

Actualmente, la técnica de enlace más popular consiste en utilizar el protocolo SOAP (Simple Object Access Protocol). WSDL especificará un servidor SOAP que disponga de acceso a la implantación real del web service y, a partir de aquí, será responsabilidad exclusiva de SOAP llevar al usuario del archivo WSDL a su implantación.

El tercer paso en la creación de WSDL consiste en describir el proceso de enlace de SOAP con un archivo WSDL. Incluirá un elemento de enlace, <binding>, dentro del elemento <definitions>. Este elemento de enlace tendrá un nombre y un tipo. El nombre identificará este enlace y el tipo identificará el elemento portType (el conjunto de operaciones) que desea asociar a este enlace. En el código de ejemplo siguiente, se observará que el nombre del elemento <portType> coincide con el atributo de tipo del elemento <binding>.

El elemento de enlace WSDL contiene una declaración de las tecnologías externas que se utilizarán para fines de enlace. Puesto que utiliza SOAP, aquí utilizará el espacio de nombres de SOAP. En terminología WSDL, la utilización de un espacio de nombres externo se denomina elemento de extensión.

En el código de ejemplo siguiente se observa un elemento <soap:binding/> vacío. Este elemento tiene como finalidad declarar que va a utilizar SOAP como enlace y servicio de transporte.

El elemento <soap:binding> tiene dos atributos: estilo y transporte. El atributo de estilo es opcional y describe la naturaleza de las operaciones en este enlace. El atributo de transporte especifica HTTP como el servicio de transporte de nivel inferior que utilizará este enlace.

Un cliente SOAP leerá la estructura de SOAP del archivo WSDL y se coordinará con un servidor SOAP en el otro extremo, de modo que debe tener especialmente en cuenta la interoperatividad (WS-I).

Después del elemento <soap:binding/> vacío, hay dos elementos <operation> WSDL, uno para cada una de las operaciones del paso 1. Cada elemento <operation> proporciona detalles de enlace para operaciones determinadas.

Por tanto, se han proporcionado otro elemento de extensión, <soap:operation/> (nuevamente, un elemento vacío que se relaciona con la operación en la que tiene lugar). Este elemento <soap:operation/> tiene un atributo soapAction que utilizará un cliente SOAP para realizar una solicitud de SOAP.

En el paso 2, la operación getProductListing sólo tiene una salida y ninguna entrada. Por consiguiente, habrá que proporcionar un elemento de <output> para esta operación. Esta salida contiene un elemento <soap:body/> (nuevamente, un elemento vacío que se relaciona con la salida en la que tiene lugar). El cliente SOAP necesita esta información para crear solicitudes de SOAP. El valor del atributo de espacio de nombres de <soap:body/> debe corresponderse con el nombre del servicio.

Ejemplo: Paso 3 Mensajes y Transporte.

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions  name="ProductService"
  targetNamespace="http://activalink.net/ProductService-interface"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://activalink.net/ProductService"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
      <types>
          <xsd:schema targetNamespace="http://actialink.net/ProductService"
            xmlns="http://www.w3.org/1999/XMLSchema/">
              <xsd:complexType name="Vector">
                  <xsd:element name="elementData" type="xsd:String" />
                  <xsd:element name="elementCount" type="xsd:int" />
              </xsd:complexType>
          </xsd:schema>
      </types>

      <message name="ProductListing">
          <part name="productID" type="tns:Vector">
      </message>
      <message name="Product">
          <part name="productID" type="xsd:String">
      </message>
      <message name="ProductPrice">
          <part name="price" type="xsd:String">
      </message>

<portType name="ProductService_port">
      <operation name="getProductListing">
          <output message="ProductListing"/>
      </operation>
      <operation name="getProductPrice">
```

```
            <Input message="Product"/>
            <output message="ProductPrice"/>
        </operation>
</portType>

        <binding name="ProductService_Binding" type="ProductService_port">
            <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
            <operation name="getProductListing">
                <soap:operation soapAction="urn:ProductService" />
                    <input>
                        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
 namespace="urn:ProductService" use="encoded" />
                    </input>
                    <output>
                        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
 namespace="urn:ProductService" use="encoded" />
                     </output>
            </operation>

            <operation name="getProductPrice">
                <soap:operation soapAction="urn:ProductService" />
                    <input>
                        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ProductService" use="encoded" />
                    </input>
                    <output>
                        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ProductService" use="encoded" />
                    </output>
            </operation>
        </binding>
</definitions>
```

Paso 4: Resumen.

Se ha generado un archivo WSDL que describe de forma completa la interfaz del servicio. Ahora WSDL requiere el paso adicional de crear un resumen del archivo WSDL. En WSDL, este archivo se denomina archivo de implantación, que se utilizará a la hora de publicar el servicio Web en un registro UDDI en la cuarta parte de esta serie de artículos. Puede Observar el código de ejemplo del paso 4, un archivo de implantación WSDL. Estas son sus principales características:

- El elemento <definitions> raíz es idéntico al del código de ejemplo 3 (un archivo de interfaz WSDL), salvo que el código de ejemplo 4 (el archivo de implantación) hace referencia a un espacio de nombres targetNamespace diferente, que hace referencia a su archivo de implantación.

- Existe un elemento <import> que hace referencia al archivo de interfaz del código de ejemplo 3 (con el nombre de archivo ProductService-interface.wsdl) y a su espacio de nombres.

- Existe un código <service> con un nombre lógico para este servicio. Dentro del elemento de servicio se encuentra un elemento de puesto que hace referencia al enlace SOAP que se ha creado en el código de ejemplo 3.

Ejemplo: Paso 4 Resumen.

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions name="ProductService"
  targetNamespace="http://activalink.net/ProductService-interface"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://activalink.net/ProductService"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">

    <import location="http://localhost:8080/wsdl/ProductService-interface.wsdl"
     namespace="http://activalink.net/ProductService-interface" />
        <service name="MobilePhoneService">
            <documentation>Servicio de Información</documentation>
            <port binding="ProductService_Binding"
              name="ProductService_ServicePort">
          <soap:address location="http://localhost:8080/soap/servlet/rpcrouter" />
        </service>
    </import>

</definitions>
```

Resumen en términos breves de como puede ser utilizada la tecnología WSDL:

- Es "vocabulario" de XML, orientado a describir en forma estructurada, la funcionalidad de un web service y la forma en que esa funcionalidad se hace disponible.

- Describe un servicio, como una colección de "puntos finales" (puertos) capaces de intercambiar mensajes.

- Cada punto tiene una definición abstracta (port type) y una definición concreta (binding).

- Permite describir en forma abstracta operaciones y mensajes, prescindiendo de las especificaciones de protocolo y tipos de datos.

- Vincula las descripciones abstractas a una implementación concreta de protocolos y tipos de datos, permitiendo la reutilización de las definiciones abstractas.

- Es extensible tanto en lo que respecta a tipos de datos (XSD) como a protocolos y formatos de mensajes.

- Provee documentación sobre el servicio que describe.

Componentes WSDL de un web service:

- Service - Conjunto de "ports" relacionados que implementan el se rvicio.

- Port - "Port type" + "Binding" es un descripción abstracta de una acción soportada por el servicio. Cada operación se corresponde a un mensaje de input o de output.

- Port type - Colección de "operations" o "signatures" de los métodos que definen el intercambio ordenado de los mensajes.

- Bindings - especifica los protocolos usa cada "port" y el "encoding".

- Message - descripción de los datos que van a ser transmitidos. Son una colección de "data values" de un tipo particular (utilizando XML Sche ma como mecanismo de estandarización).

7.3 UDDI (Universal Discovery, Descripcion and Integration).

Es un elemento básico sobre el que se asientan los web services, hace posible que empresas pueden tanto publicar como encontrar web services UDDI provee un mecanismo para que los negocios se "describan" a si mismos y los tipos de servicios que proporcionan, luego se pueden registrar y publicarse en un Registro UDDI. Los web services son registrados para que los posibles usuarios puedan encontrarlos, Tales negocios publicados pueden ser buscados, consultados o "descubiertos" por otros negocios utilizando mensajes con SOAP. Es un protocolo para registros basados en web que contiene información acerca de web services. Un registro UDDI es como la lista de los web services. Una entrada en UDDI es un archivo XML que, realizando una similitud con las páginas amarillas, describe el proveedor y los servicios que ofrece.

El proyecto Universal Description, Discovery and Integration (UDDI) es una iniciativa de la industria para crear una plataforma independiente que, a modo de marco de trabajo abierto, especifica servicios, descubrir negocios e integrar servicios empresariales a través de Internet.

UDDI es el primer esfuerzo de la industria organizado por los principales proveedores de plataformas y software, así como empresas líderes en el e-business y operadores de mercado. Las empresas miembros actúan como catalizadores iniciales para desarrollar rápidamente la plataforma UDDI y los sistemas relacionados. El propio protocolo UDDI es la piedra angular que permitirá a las empresas encontrar y comerciar de una forma rápida, fácil y dinámica con otras organizaciones utilizando sus aplicaciones preferidas. Más de 220 compañías ya son miembros de la comunidad UDDI.

En pocas palabras, el UDDI es una especificación para un registro distribuido de información sobre web services. UDDI es un sistema ideado para describir servicios (junto con WSDL) y localizar empresas que ofrezcan los servicios de descripción, localización e integración universal en un directorio; UDDI también permite guardar las interfaces de esos servicios descritas en WSDL y usar SOAP para llevar a cabo las comunicaciones. La tecnología UDDI se usa activamente por empresas como Microsoft, Dell, Fujitsu, HP, Hitachi, IBM, Intel, Microsoft, Oracle, SAP y Sun y etc.

En esencia, UDDI consiste de dos partes. En primer lugar, UDDI es una especificación técnica para construir directorios distribuidos de empresas y servicios web. Los datos se almacenan en un formato XML específico, y la especificación UDDI añade los detalles del API para buscar datos existentes y publicar otros nuevos. En segundo lugar, el Registro de Negocios UDDI es una implementación totalmente funcional de la especificación UDDI puesta en marcha en Mayo de 2001 por Microsoft e IBM. Este registro UDDI permite que cualquiera se registre en él.

Los web services están tomando cada vez más importancia día a día. Se diseñan como "cajas negras" que oculten la complejidad de los sistemas finales y permitan una fácil comunicación. Sin embargo es fundamental tener un medio de localizar esos servicios, tarea más difícil conforme crece el número de servicios disponibles.

La especificación UDDI simplifica esa tarea, permitiendo a una organización publicar información sobre los servicios que ofrece y localizar información sobre que web services necesita utilizar. UDDI es simplemente un repositorio de documentos XML (y un esquema) que define un mensaje SOAP para el registro y petición de información.

Los documentos XML guardados en el sistema UDDI son alojados por compañías que aceptan mantener un nodo y siguen la especificación dada por el consorcio UDDI.org. Actualmente Microsoft e IBM mantienen nodos públicos que siguen la especificación de versión 1, y Compaq-HP mantiene un nodo de versión 2.

Los datos manejados por UDDI se dividen en tres categorías:

1. **Páginas Blancas:** Con información general sobre una empresa (nombre, descripción, información de contacto, dirección y teléfono).

2. **Páginas Amarillas:** Es muy similar a su equivalente telefónico, e incluyen categorías de catalogación industrial tradicionales, ubicación geográfica, etc. Mediante el uso de códigos y claves predeterminadas, los negocios se pueden registrar y así facilitar a otros servicios la búsqueda usando estos índices de clasificación.

3. **Páginas Verdes:** Con información técnica sobre un servicio web. Generalmente esto incluye un apuntador a la especificación externa y una dirección en la que invocar el servicio.

UDDI puede ser usado por:

- Programas y programadores para localizar información acerca de web services
- Buscar compañías de un determinado sector con un tipo de web service específico.
- Preparar sistemas que sean compatibles con otros web services.
- Describir web services con el fin de que otros los puedan utilizar.

Un registro UDDI puede ser:

a) ***Público, disponible en Internet.*** El propietario del registro permite publicar en él servicios y que cualquiera los utilice. En este sentido la Administración debe proporcionar un registro en Internet de servicios públicos para la comunidad de desarrolladores.

b) ***Privado, restringido a una determinada organización.*** En este caso el registro facilita compartir los servicios dentro de la organización. De este tipo sería el registro a crear en la Intranet Administrativa.

c) *Híbrido, disponible más allá de la organización pero con restricciones.* De este tipo sería el registro a crear para la comunicación con otras Administraciones. Ejemplos de servicios descritos en este tipo de registro serían la "consulta de precios del catálogo central de suministros" o la "consulta de oficinas de registro".

Esta clasificación no tiene diferencias técnicas, sino de ámbito de actuación y de uso. Podemos resumir diciendo que un web Service, se describe con un archivo WSDL, se registra en UDDI y se muestra en web a través de SOAP.

UDDI puede ayudar a resolver los siguientes problemas que surgen a la hora de diseñar un web service para cualquier proyecto que tiene en mente la integración universal con partners:

- Descubrimiento de partners más adecuados de entre los muchos presentes en Internet o en la plataforma. Los asociados suyos podrán buscar a fabricantes o distribuidores y descubrirlos.

- Obtener información sobre cómo contactar con la empresa de interés.

- Conseguir nuevos clientes y facilitar el acceso a los actuales incrementando los servicios ofertados y extendiendo el mercado al que se puede acceder.

- Describir servicios y procesos empresariales en un entorno seguro y fácil de usar. Los proveedores de forma automática pueden saber que web services, se ejecutan en la aplicación empresarial del fabricante e integrarse fácilmente.

Un ejemplo de cómo se podría usar es el siguiente: supóngase que se creara un web service estándar a través de UDDI para la venta de tuercas. Los fabricantes podrían registrar sus servicios en un directorio UDDI siguiendo ese estándar (e la interfaz UDDI).

Así, las tiendas, accediendo al repositorio UDDI a través de la interfaz, podrían comunicarse con el servicio ofrecido por cualquier fabricante de tornillos para hacer las compras de producto o hacer uso de cualquier función ofrecida por el fabricante (el servidor de web services).

El primer paso para utilizar esa tecnología es decidir cómo modelar la compañía y sus servicios en UDDI. Para ello, han ideado el concepto de tModels (modelos tecnológicos) que representan las interfaces, abstracciones técnicas y metadatos del servicio. Como se observa, es lo que se ha dicho que hacía el lenguaje WSDL; ya que UDDI utiliza WSDL como tModel (aunque podría utilizar otros).

Lo primero que hay que hacer es determinar los tModel o archivos WSDL para el web service. Si el servicio está basado en un archivo WSDL que ya está registrado, entonces sólo se necesita saber la URL hasta ese archivo y el identificador que le asignó UDDI al regístralo anteriormente. Si el servicio está basado en un archivo WSDL no registrado, habrá que registrarlo previamente.

El modelo funcional UDDI/SOAP, no es el único modelo para descubrimiento y mensajes en los web services. El modelo ebXML ha sido también desarrollado para hacer lo mismo, así como también brindar una interfaz de software para negocios, funciones de seguridad robusta y otras, que permiten transacciones reales de comercio electrónico. El ebXML y UDDI/SOAP son tecnologías complementarias y al final se pueden utilizar ambos métodos a la vez.

Cómo su nombre lo indica, el estándar UDDI provee un mecanismo para que los negocios se "describan" a si mismos y los tipos de servicios que proporcionan y luego se pueden registrar y publicarse en un Registro UDDI. Tales negocios publicados pueden ser buscados, consultados o "descubiertos" por otros negocios utilizando mensajes con SOAP.

Una vez descubiertos los negocios con quien se pueden asociar, los negocios pueden utilizar este mecanismo para "integrar" sus servicios en conjunción con sus socios y proveer los servicios a sus clientes.

La información que contiene un registro UDDI consiste en este momento de cuatro estructuras de datos aunque la versión UDDI 2.0 tiene una estructura de datos adicional:

a) **BusinessEntity.** Esta estructura captura la información sobre un negocio o entidad y que es utilizada por el negocio para publicar información descriptiva sobre si misma y los servicios que ofrece.

b) **BusinessService.** Esta estructura representa los servicios o procesos de negocios que provee la estructura businessEntity.

c) **BindingTemplate.** Esta estructura representa los datos importantes que describen las características técnicas de la implementación del servicio ofrecido.

d) **TModel.** El papel principal de esta estructura es la de representar una especificación técnica.

e) **PublisherAssertion.** Esa estructura particular a UDDI 2.0 relaciona diferentes empresas que tienen acuerdos comerciales o proporcionan servicios conjuntamente.

Durante el proceso de registro hay que saber las categorías e identificadores de la empresa de acuerdo a las clasificaciones del North American Industry Classfication System, Universal Standard Products and Services Codes, ISO 3166, Standard Industry Classification, y GeoWeb Geographic Classification.

Habrá que decidir también qué nivel de seguridad necesitamos. Una vez que se tenga modelada la empresa y los servicios, obtendrá una cuenta en el sistema de registro UDDI a través del sistema web.

La búsqueda de web services consiste en localizar descripciones de servicios en WSDL registrados en UDDI. En ese proceso, los clientes web pueden determinar qué tipos de servicios existen, qué pueden hacer, y cómo interactuar con ellos.

UDDI permite acceder al servicio de localización mediante una interfaz de programa (API) o mediante una interfaz web.

La arquitectura técnica de UDDI consiste en tres partes:

1. **Modelo de datos UDDI:** Un esquema XML que describe las empresas y los web services.

2. **API UDDI:** Un API basado en SOAP para buscar y publicar datos UDDI.

3. **Nube de servicios UDDI:** Sitios que proporcionan implementaciones UDDI y sincronizan sus datos.

El registro UDDI proporciona un servicio de directorio lógicamente centralizado pero físicamente distribuido. Esto significa que los datos entregados a uno de los nodos raíz de UDDI son replicados automáticamente en los demás nodos raíz (opcional para redes privadas).

Básicamente, UDDI permite dos acciones: pedir información y publicar información. Cada acción se lleva a cabo mediante ciertos métodos. Los métodos de búsqueda y descubrimiento de datos son: find_business, find_service, find_binding, find_tModel y los métodos para la publicación son los siguientes: save_business, save_service, save_binding, save_tModel, delete_business, delete_service, delete_binding y delete_tModel.

El funcionamiento del servicio UDDI se lleva acabo utilizando mensajes SOAP con el contenido apropiado para llamar al método correspondiente.

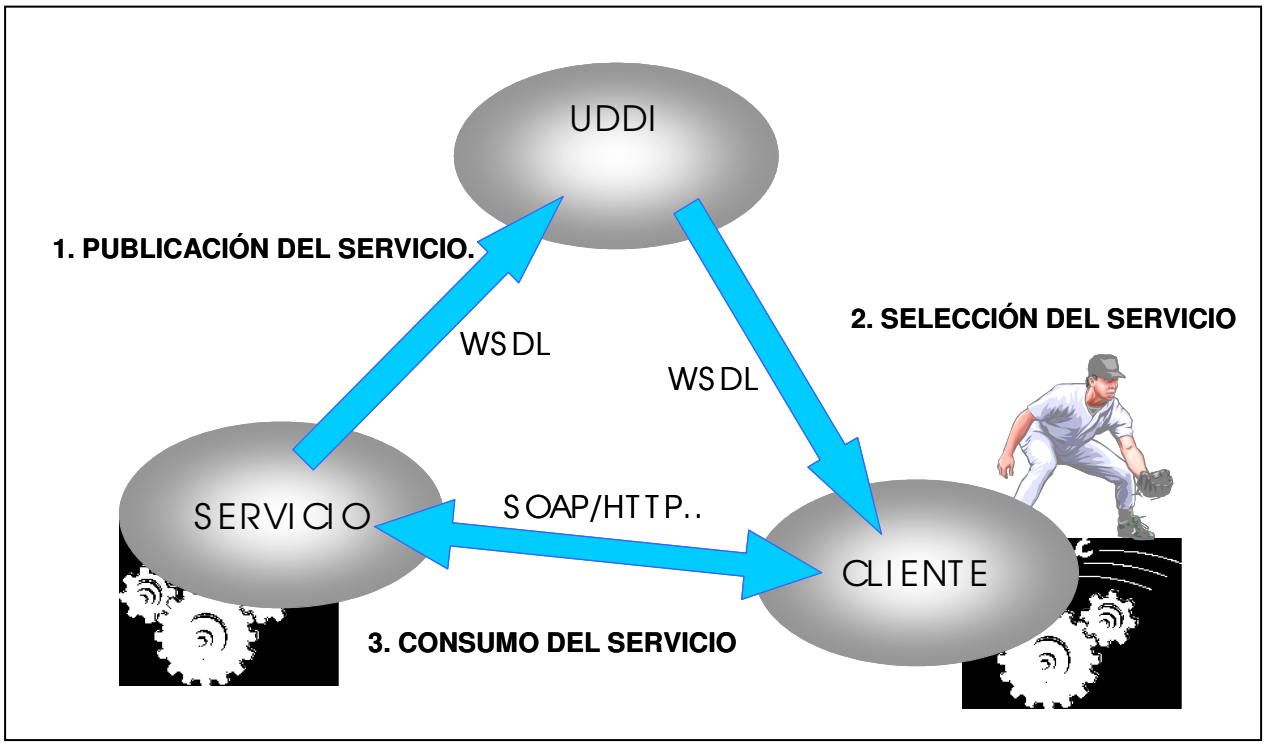


Figura 10. Funcionamiento UDDI

Por ejemplo, si desea localizar la empresa activaLink, puede enviar el siguiente mensaje XML en el cuerpo del mensaje SOAP:

```
<find_business generic="1.0" xmlns='urn:uddi-org:api'>
    <name>activaLink</name>
</find_business>
```

El mensaje SOAP devuelto por el servicio UDDI incluye las empresas que contienen "activaLink" en su nombre y los servicios registrados. La información se devuelve en una estructura XML denominada businessInfos.

```
<businessList generic="1.0" operator="DUNS" truncated="false"
  xmlns="urn:uddi-org:api">
 <businessInfos>
  <businessInfo businessKey="0036B46Z-EB27-42EK-AC09-9953CFF462A3">
   <name>Activa Link, S.L.</name>
  <description xml:lang="es">
```

Activa Link ofrece servicios no comerciales de Investigación Forense de equipos Informáticos y plataformas Unix, BSD, GNU/Linux y Windows.

```
  </description>
  <serviceInfos>
    <serviceInfo businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
                 serviceKey="1FFE1F71-2AF3-45FB-B788-09AF7FF151A4">
     <name>Servicio Registro de Dominios</name>
    </serviceInfo>
    <serviceInfo businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
                 serviceKey="8BF2F51F-8ED4-43FE-B665-38D8205D1333">
     <name>Servicio Contable con Empresas</name>
    </serviceInfo>
   </serviceInfos>
  </businessInfo>
 </businessInfos>
</businessList>
```

Desde este punto puede continuar y obtener información sobre un servicio específico. Por ejemplo, buscar el servicio de registro de dominios. Se puede tomar serviceKey de los resultados anteriores y utilizar <get_service> para buscar un servicio en concreto:

```
<get_serviceDetail generic='1.0' xmlns='urn:uddi-org:api'>
   <serviceKey>1FFE1F71-2AF3-45FB-B788-09AF7FF151A4</serviceKey>
</get_serviceDetail>
```

Esto devuelve el siguiente <bindingTemplates> que contiene información completa acerca del servicio de registro de dominios ofrecido por la empresa activaLink:

```
<serviceDetail generic="1.0" operator="DUNS" truncated="false"
  xmlns="urn:uddi-org:api">
```

```
<businessService businessKey="0036B46Z-EB27-42EK-AC09-9953CFF462A3"
                 serviceKey="1FFE1F71-2AF3-45FB-B788-09AF7FF151A4">
<name>Servicio Registro de Dominios</name>
<description xml:lang="en">TLD UDDI SOAP/XML message-based domain
registration interface.</description>
<bindingTemplates>
<bindingTemplate bindingKey="0036B46Z-EB27-42EK-AC09-9953CFF462A3"
        serviceKey="1FFE1F71-2AF3-45FB-B788-09AF7FF151A4">
<description xml:lang="en">Servicio de Registro TLDs</description>
<accessPoint URLType="https">https://api.activalink.net/register</accessPoint>
      <tModelInstanceDetails>
          <tModelInstanceInfo tModelKey="uuid:4CD7E4BC-648B-
443EAAC8AE23">
<description xml:lang="en">Interfaz registro de TLDs</description>
          </tModelInstanceInfo>
      </tModelInstanceDetails>
</bindingTemplate>
</bindingTemplates>
<categoryBag>
<keyedReference keyName="KEYWORD" keyValue="API"
 tModelKey=" uuid:4CD7E4BC-648B-443EAAC8AE23">
</keyedReference>
</categoryBag>
</businessService>
</serviceDetail>
```

Ahora comprueba que empieza a obtener información detallada sobre el propio web service en línea. Estos datos indican que existe en realidad el punto de acceso. También se nos informa de que se puede obtener acceso sin problema a través de HTTP al punto de acceso a la consulta UDDI y que los puntos de acceso de publicación se encuentran protegidos por SSL.

Asimismo, se puede utilizar la información de tModelKey para buscar todas las empresas registradas que proporcionan el mismo servicio de registro de dominios por ejemplo:

```
<find_business generic=' 1.0' xmlns=' urn:uddi:api' >
<tModelBag>
      <tModelKey>uuid:4CD7E4BC-648B-443EAAC8AE23</tModelKey>
</tModelBag>
</find_business>
```

Si la intención es crear una aplicación que permita conexiones dinámicas a los web services que proporcionan diferentes socios empresariales externos así como internos, la opción más acertada en este caso sería conectar la plataforma al registro UDDI.

7.4 XML: Lenguaje extensible de etiquetas.

Es un estándar para la representación estructurada de datos y crear etiquetas, definido por el World Wide Web Consortium (W3C). Las características especiales son la independencia de datos, o de la separación de los contenidos de su presentación. Es un metalenguaje que permite diseñar un lenguaje propio de etiquetas para múltiples clases de documentos. Los documentos XML se componen de unidades de almacenamiento llamadas entidades (entities), que contienen datos analizados (parsed) o sin analizar (unparsed). Los datos analizados se componen de caracteres, algunos de los cuales forman los datos del documento y el resto forman las etiquetas. Las etiquetas codifican la descripción de la estructura lógica y de almacenamiento del documento. XML proporciona un mecanismo para imponer restricciones en la estructura lógica y de almacenamiento.

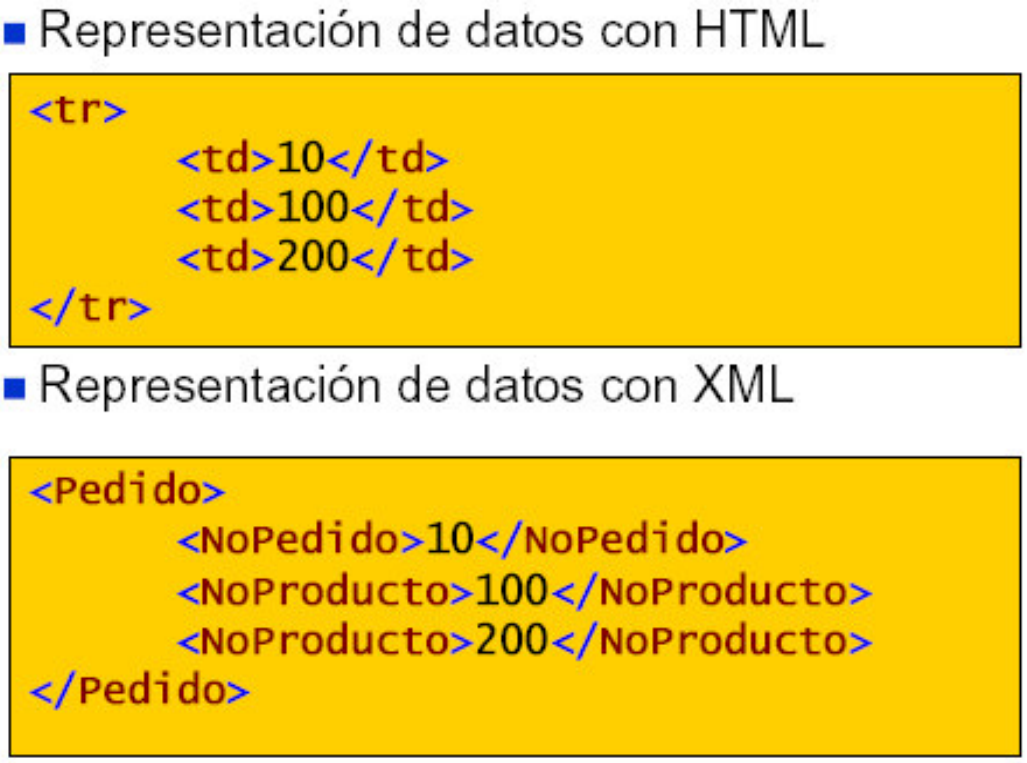


Figura 11. Representación de datos con HTML y XML.

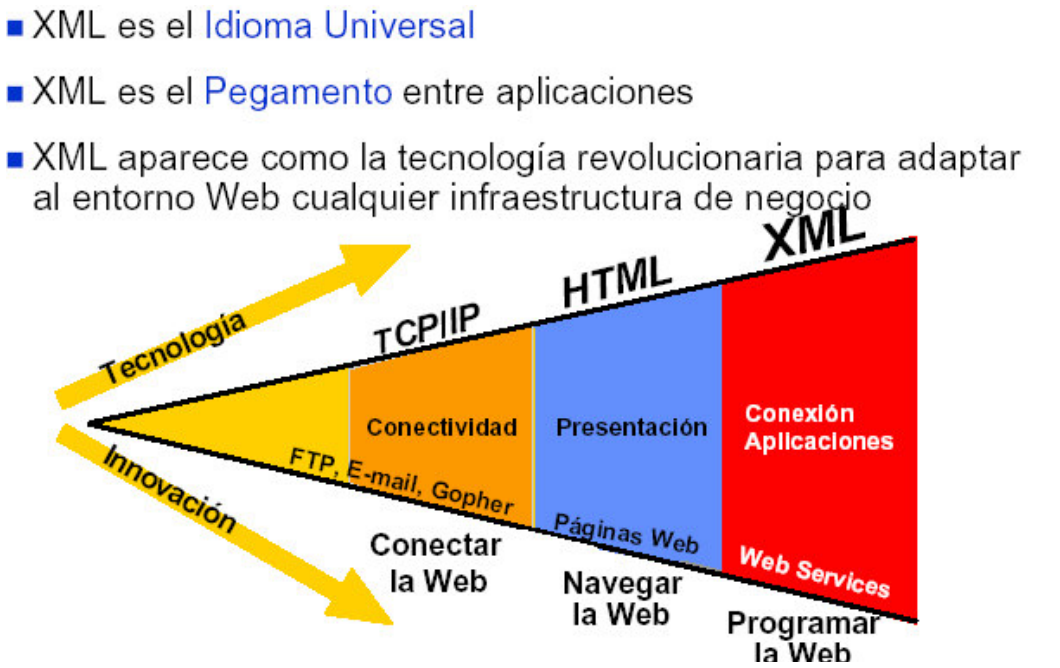


Figura 12. Tecnología XML

Figura 13. Aplicaciones XML

XML Representa datos jerárquicamente (en un árbol), proporciona contexto a los datos (datos autodescriptivos), separando la presentación (HTML) de los datos (XML), es una extensión o un subset de SGML vrs HML que es una implementación de SGML.

Diferencias entre XML y SGML.

XML es un subconjunto de SGML concebido para aplicaciones en Internet. XML impone unas reglas léxicas y sintácticas más rígidas, "parsers" más sencillos: No se pueden omitir las marcas de finales. Forma compacta para los elementos vacíos. No existe el operador [&]. Distingue entre mayúsculas y minúsculas. Los valores de atributo deben ir entre comillas, etc.

Diferencias:

- HTML es una aplicación SGML.
- HTML mezcla información estructural y de presentación.
- HTML tiene marcas sintácticas con un significado fijo.
- En XML es un subconjunto propio de SGML.
- El significado de las marcas puede cambiar.
- Solo se codifica estructura de datos, no forma de presentación.
- HTML es una aplicación XML o reformulado como aplicación XML.

XML tiene un impacto fundamental sobre:

- Integración de aplicaciones
- Gestión de contenidos
- Intercambio de datos
- Portales corporativos

Dos líneas de aplicación principales son:

- Arquitectura de web services
- Utilización en B2B

Los lenguajes XML no se crean documentos en XML, XML se utiliza para crear lenguajes de marcado propios (aplicaciones XML).

Se crean documentos utilizando estos lenguajes. Estos lenguajes se definen especificando los elementos y los atributos permitidos. Esta especificación se realiza mediante reglas gramaticales. Un conjunto concreto y bien formado de tales reglas forman un esquema XML. Un esquema define un conjunto coherente de documentos (tipo de documentos).

8. LOS WEB SERVICES Y LA TECNOLOGIA ACTUAL

8.1 Influencia en los factores tecnológicos, económicos y sociales

Las aplicaciones web existentes actualmente ya no son suficientemente flexibles para satisfacer las crecientes necesidades empresariales. El modelo actual de negocio electrónico apenas puede facilitar la integración de las aplicaciones de Internet con el resto de software de las empresas. Para que las empresas puedan extraer el máximo beneficio de Internet las páginas web empiezan a evolucionar y éste es el punto dónde surge el concepto de web services.

El uso de los web services aporta ventajas significativas a las empresas. El principal objetivo que se logra, es la interoperabilidad y la integración. Mediante los web services, las empresas pueden compartir servicios software con sus clientes y sus socios de negocio. Esto ayudará a las compañías a escalar sus negocios, reduciendo el coste en desarrollo y mantenimiento de software, y sacando los productos al mercado con mayor rapidez. La integración de aplicaciones hará posible obtener la información demandada en tiempo real, acelerando el proceso de toma de decisiones. La evolución de Internet hacia los web services, mejorará los resultados globales de las empresas, reduciendo sus gastos y guiándolas hacia una mejora progresiva de la calidad. La adopción de la tecnología de web services por la industria es el primer paso hacia una economía global.

La actual revolución de los web services anuncia un nuevo estallido en el crecimiento de la industria de las tecnologías de la información, ya que la amplia gama de servicios basados en Internet pueden ser combinados en un infinito número de productos. La clave se encuentra en las tecnologías que se desarrollan a partir de estándares abiertos para toda la industria. Tal y como el PC revolucionó la industria de las telecomunicaciones, el nuevo estándar XML promete transformarla otra vez con una generación de web services basados en él. El impacto de los web services se anuncia masivo. Recientemente, Finantial Times lo calificaba en un artículo de análisis como de "cambio tectónico en la industria del software". En esta nueva era, se ha pasado de una estructura vertical (en la que el usuario se veía forzado a elegir la tecnología de entre un reducido número de fabricantes), construida sobre códigos y arquitecturas cerradas, a otra horizontal. En esta última, cimentada sobre las tecnologías abiertas de XML, (en la que se multiplican las oportunidades de negocio para toda la industria y las ventajas para el consumidor final), los expertos destacan tres áreas con mayor posibilidad de crecimiento. Son las siguientes: desarrollo de software, negocios entorno a la distribución de web services y centros de recepción de datos en línea.

Los web services al ser utilizados por la sociedad salvadoreña pueden aplicarse y contribuir a muchos sectores, dando como resultado la integración de la información entre muchas entidades gubernamentales o empresariales logrando así que la información este centralizada y al alcance del que la necesite, a continuación se detallan algunos ejemplos en que los web services pueden aplicarse en varios ámbitos:

Los datos se pueden establecer en forma centralizada y única por la entidad correspondiente, con el fin que cada una se especialice en brindar información de un solo tipo ejemplos:

La base de datos de los duicentros en la cual se registra información personal por medio del documento único de identidad puede ser utilizada por varios sistemas informáticos para distinguir a cada salvadoreño.

El registró de contribuyentes del impuesto a la transferencia de bienes y muebles y a la prestación de servicios por parte del ministerio de hacienda (Dirección general de impuestos internos). En el cual se detalla la actividad económica de cada salvadoreño, empresas y sociedades anónimas de capital variable de este país.

La inscripción de vehículos por parte del viceministerio de transporte (Dirección general de tránsito) administrada por la empresa SERTRACEN, ya que se registran las especificaciones de vehículos como marca, año, color, placa, capacidad, # de chasis, etc.

El registro de patronos y empleados del país se lleva a cabo por medio del ISSS y las AFP, en el cual se detallan el número de empleados de la empresa, el sueldo de cada empleado, los beneficiarios en caso de muerte

Teniendo la información centralizada, las formas de integrar los web services son muchas ya que no se tendrá que guardar la información general en cada sistema sino que solo será consultada cuando se necesite, puede existir un web service que tenga la función de brindar la información personal del individuo como nombre completo, estado civil, dirección, teléfono, fecha de nacimiento, sexo, tipo de sangre, etc. Estos datos pueden ser utilizados por muchos sectores, supóngase que un empleado desee afiliarse a una AFP o al ISSS, por medio de este web service se accesara la información personal, solo será necesario adicionar información extra como salario, numero patronal o razón social a la que pertenece, beneficiarios inscritos, etc.
Otro caso seria cuando un paciente necesite concertar una cita en el ISSS donde se establezca el consultorio, el medico que lo atenderá, la hora o número de turno, el hospital a asistir, lo puede hacer por medio de un web service, que a su vez informe al medico del tipo de paciente y de su historial clínico como los últimos medicamentos que se le recetaron, las enfermedades que ha sufrido, los especialistas que lo han asistido, resultados de exámenes practicados, diagnósticos aplicados, cuando estuvo ingresado en algún hospital, etc.

Una persona que quiera reservar su estadía en un hotel, puede establecer varios servicios con anticipación como la habitación a usar, el tiempo de estancia, este a su vez puede contactar otro web service que proporcione información acerca lo que ofrecen diversas compañías para rentar automóviles o si lo prefiere contactar a empresas que ofrezcan servicio de taxis, descargar automáticamente de su tarjeta de crédito o debito el costo total de los servicios a usar, servicios de alimentación, guías informativos para conocer la cuidad o hacer un recorrido interesente del lugar.

Un estudiante común de la universidad don bosco puede realizar todo el proceso de inscripción de asignaturas para su próximo ciclo mediante un web service que muestre su boleta de notas, hoja de asesoria para determinar que materias puede inscribir, solvencia de pagos y biblioteca, los grupos teóricos y laboratorios, docentes asignados, horarios, aulas y cupos hábiles, pago a colecturia descargado automáticamente desde su cuenta bancaria o tarjeta de crédito, entre otras. Al efectuar este proceso la información contenida puede ser usada por múltiples subsistemas internos, por ejemplo un docente puede observar el listado de los alumnos que están inscritos en su materia, la condición de matricula; el departamento de contabilidad podría saber cuanto dinero ha ingresado a la cuenta de la universidad, el departamento de atención al estudiante podría estimar las posibles rutas de buses para transportar en las horas de mayor demanda según el grado de asistentes, la biblioteca universitaria agilizaría el proceso de autorización de prestamos de libros u otro material a los estudiantes inscritos, los decanatos de las diferentes facultades se enterarían mas rápido de la cantidad de demanda estudiantil, con el objetivo de abrir nuevos grupos, establecer nuevos horarios o contratar nuevo personal, entre otras.

Cuando se informe a la policía nacional civil sobre el hurto o robo de un automotor un web service podrá accesar a sertracen para conocer las características del vehículo (modelo, marca, año, etc.), Podrá conectarse con la empresa que vendió este vehículo, determinar cuantos y cuales fueron los propietarios, en que año se efectuó la compra, el monto de adquisición, informar a las distintas delegaciones, puestos de policía, retenes en carreteras, fronteras del país, etc.

Un patrono al momento de decidir si contrata a un empleado puede accesar la información personal que necesite, solvencias policiales, antecedentes penales, Informarse en que escala se encuentra mediante el record que tiene el empleado en el ministerio de trabajo, saber los logros que este ha realizado, el historial de salarios, los motivos de despido o renuncia de trabajos anteriores, los nombres de jefes números telefónicos y correo electrónicos para solicitar referencias de trabajo, etc.

Los web services podrán incorporar varias tareas en una sola aplicación para realizar transacciones bancarias, como consultas de saldo, pagar tarjetas de crédito, impuestos (Alcaldía, energía eléctrica, vivienda, agua, etc.) para pagar la universidad, energía eléctrica, agua, tarjetas de créditos, solicitar un préstamo, sin que importe el banco o compañía de pago ya que estos tendrán la capacidad de comunicarse con los sistemas informáticos propios de cada empresa.

8.2 Beneficios sobre la tecnología que actualmente se utiliza

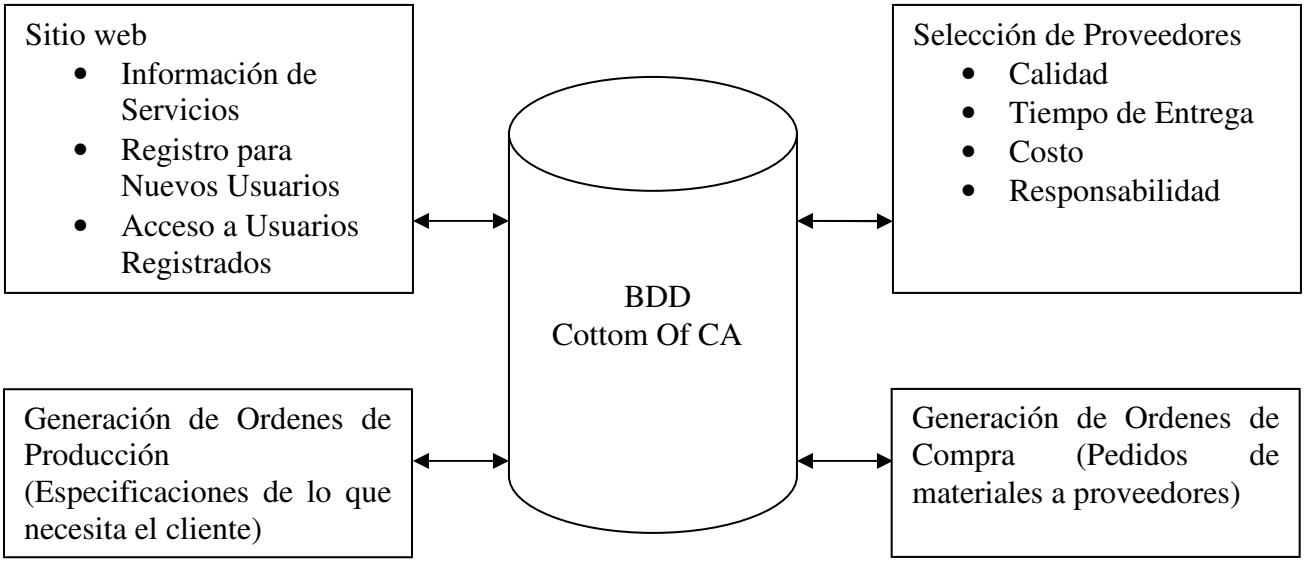
WEB SERVICES	TECNOLOGÍA ACTUAL (APLICACIONES)
Integración entre aplicaciones Pueden comunicarse con el software existente y efectuar procesos en funciones informáticas específicas las cuales logrando así una interacción con varias aplicaciones a la vez en una sola operación.	Por cada tarea especifica en una aplicación diferente la información se encuentra repetida debiéndose procesar en cada sistema, las aplicaciones no pueden interactuar o integrarse unas con otras en su totalidad.
Se encuentran estructurados en forma de componentes o módulos de software en la red para efectuar una acción, estos la pueden realizar cuantas veces se necesite con solo la llamada del web service eliminando la necesidad de crear software cada vez que se necesite ejecutar una actividad determinada.	Se encuentran estructurados como un todo en la red, cada aplicación diferente puede realizar las mismas operaciones de manera independiente, lo que puede hacer que operaciones similares que se ejecutan en distintos software se repitan
Están escritos en un lenguaje universal y capaz de ser interpretado por el software de manera independiente de la plataforma en que fueron creados, como resultado de esto se obtienen las aplicaciones que interactúan en diferentes sistemas operativos, teléfonos móviles, equipos de telecomunicaciones o cualquier otro dispositivo que interactué con las computadoras.	Se encuentran escritos en lenguaje propio del entorno de programación en que fueron diseñados y creados, en algunos casos las aplicaciones solo son compatibles con la otras que se realizaron en la misma versión del lenguaje de programación; son creados para operar en un sistema operativo especifico o de la misma familia, otros solo funcionan en un entorno de trabajo por defecto como un navegador web, en una consola o servidor
Los web services necesitan un protocolo para ser transportados usualmente son creados para operar bajo el protocolo HTTP pero estos no solo pueden operar bajo este sino que lo pueden hacer bajo cualquier protocolo que trabaje en la capa de transporte como lo son TCP, SMTP, JABBER, etc.	El software de la actualidad esta diseñado para operar en un protocolo de transporte ya establecido sin dar una alternativa de ejecutarse en un ámbito de trabajo diferente de este u otro protocolo de transporte para el cual fue hecho, el ejemplo más común en este aspecto son los sistemas web, los cuales dependen únicamente del protocolo HTTP, sin poder funcionar si no se utiliza este

Gracias a que son escritos usando un lenguaje universal y utilizan estándares abiertos, los web services ofrecen un esquema de escalabilidad muy amplio ya que pueden incrementar sus funciones espontáneamente de una forma práctica y útil; un web service puede incorporar funciones de web services ya existentes dando como resultado el crecimiento de las capacidades en las tareas a realizar	La gran mayoría de las aplicaciones actuales y mucho mas las de tipo propietarios no gozan de escalabilidad ya que solo el creador puede agregar capacidad o mejoras al software, cuando estas son realizadas el creador vende el software nuevamente aduciendo que es una nueva versión, además estos sistemas no se pueden incorporar fácilmente a otros que no sean de la misma familia, y cuando ofrecen cierto tipo de acoplamiento lo hacen posible mediante clases como MFC, componentes ActiveX, API, etc.
Se inscriben a una base de datos en línea llamada registro UDDI (descripción, descubrimiento e integración universal) con el objetivo que futuros usuarios los encuentren fácilmente, observen sus características de funcionamiento y respuesta. Ahí se detalla el servidor en el que se encuentra alojado el web service para poder ser llamado o invocado cuando se que requiera	Las aplicaciones artesanales casi nunca detallan sus características y capacidades, la forma de saber que hace cada software, se realiza por medio de sitios de cada fabricante o centros de ventas autorizados

8.3 Costos al migrar aplicaciones implementando web services y web tradicional

Este tópico es muy amplio para medirlo con certeza o precisión, ya que hay muchos factores a tomar en cuenta como el tamaño de lo que se necesita hacer, el software para el desarrollo de las aplicaciones, este documento se basara en una comparación al ejecutar una aplicación implementando web services o implementando aplicaciones web tradicionales o que se usan en estos tiempos con el propósito de agilizar el proceso de pedidos de una empresa.

El problema de pedidos: la empresa "Cottom Of CA" se dedica a exportar ropa confeccionada a tiendas del mercado americano, existe una variedad de estilos y marcas en las prendas de vestir que ahí se ofrecen, como lo son Adidas, Marvels, Nike, JcPenney, etc. Actualmente los clientes se informan de los servicios de Cottom Of CA por medio de su web site oficial http://www.cottomofca.com, hacen el contacto por teléfono con personal encargados de cuenta aquí en El Salvador y una vez hayan pactado o hecho el negocio mandan las especificaciones de las prendas (marca, color, estilo, tallas, cantidad, etc.) por correo electrónico, Cottom Of CA se dedica a comprar todos los materiales que contendrá la prenda (etiquetas de marca, ganchos, bolsas, tela, viñetas, etc.) al proveedor que mas le parezca, y así poder entregar su producto terminado listo solo para colocarlo en las tiendas de venta. La empresa requiere un software que le permita tomar los pedidos que el cliente necesite, teniendo lista esta información poder seleccionar el proveedor de materiales que más le convenga.
Diagrama a bloques de lo que se necesita:



Sitio web: La información que se brinda ahí es de carácter informativa, como formas de contacto, ubicación de la empresa, estándares o certificaciones que la respaldan, cantidad de empleados, maquinaria y tecnología usada para la confección, muestras de prendas elaboradas, registro de clientes mas conocidos o con los que se han establecido contratos permanentes, etc. Se quiere anexar un modulo que permita registrar un nuevo cliente o proveedor, en el cual se detalle información general como medios de contacto, dirección, números de cuentas bancarias o tarjetas de créditos para realizar pagos, etc. También se necesita incluir el modulo de acceso en línea para los usuarios ya registrados.

Generación de Ordenes de Producción: aplicación en la cual el cliente especifica la cantidad de prendas a ordenar, el doblado y empaquetado, medio de transporte, fecha de cancelación y recibo, la tienda o lugar de destino, los colores, las tallas, diseños, logotipos o imágenes o dibujos animados, el estilo, el peso, etc. Incluye un programa que permita establecer comunicación en línea entre el encargado de cuentas y el cliente especificando que solo tendrán lugar en horas de oficina, para tratar temas como establecer el contrato entre el cliente y la empresa, mostrar las distintas formas de pagos, ofrecer promociones, combos o descuentos especiales, definir los auditores autorizados y fechas en que se visitaran las instalaciones de la planta, establecer los acuerdos legales entre ambas entidades, entre otras.

Selección de Proveedores: Componente que permita observar al encargado de cuenta los proveedores candidatos para la compra de materiales basándose en varios criterios como la calidad del servicio y producto, el tiempo de entrega, la responsabilidad con que se acuerda, las formas de pago y el limite de tiempo, los costos dependiendo de la cantidad a ordenar, transporte, etc. Esto con el objetivo de informarse todas las características que los clientes ofrecen y así poder elegir uno en específico para suplir lo que se requiere.

Generación de ordenes de compra: consiste en un modulo que sea capaz de generar el pedido de materiales cuando ya se ha seleccionado el proveedor que mas convenga, especificando la cantidad de material requerida, las fechas de creación, recibimiento y cancelación, la forma de pago y el limite de tiempo, el valor pactado por unidad de medida, y el costo total en la orden.

Requisitos mínimos del sistema:

- Procesador Pentium III 600 MHz
- 256 MB RAM
- 500 MB Libre en disco duro
- Acceso a Internet necesario para características en línea, ancho de banda
- Modulo JSP
- Jakarta TomCat
- Sistema Operativo Red Hat Linux 9.0

Costos estimados usando un sistema web tradicional actual:

DESCRIPCIÓN	PRECIO ($)
Anexo al sitio web de la empresa capacidad de registrar nuevos clientes y proveedores	$ 1000
Anexo al sitio web de la empresa acceso para los usuarios ya registrados	$ 1500
Modulo para generación de ordenes de Producción	$ 2000
Selección de Proveedores	$ 3000
Generación de ordenes de compra	$ 2000
Total	$ 9500

Costos estimados implementando web services:

DESCRIPCIÓN	PRECIO ($)
Anexo al sitio web de la empresa capacidad de registrar nuevos clientes y proveedores	$ 750
Anexo al sitio web de la empresa acceso para los usuarios ya registrados	$ 1000
Modulo para generación de ordenes de Producción	$ 1500
Selección de Proveedores	$ 3000
Generación de ordenes de compra	$ 1500
Total	$ 7750

La estimación de estos costos se basa en varios aspectos tales como la cantidad de horas de trabajo por parte del profesional de software (El Sueldo promedio por hora que debería ganar un profesional en informática en El Salvador es $18) al usar web services no serán la misma cantidad que si usa implementa el método con sistemas web tradicionales ya que existen algunos módulos de software en los que no es necesario hacer software nuevo porque se cuenta con la ventaja de solicitar la ayuda de web services existentes que realicen la función en común, por ejemplo la autenticación de usuarios se puede ejecutar llamando un web service de Microsoft titulado PASSPORT .Net.

8.4 Estadísticas sobre las tecnologías de web services en los últimos años

Según un estudio publicado la consultora IDC sobre el creciente papel que juegan las TI (Tecnologías de la Información) en la economía global, se prevé que el mercado mundial de ordenadores, dispositivos de red, software y servicios superará el trillón de dólares por primera vez este año, y pasará de 1,4 trillones en el 2005.
El informe, El Estudio de Impacto Económico, revela que la inversión mundial creció más del diez por ciento anual durante la mayor parte de la última década, lo que supone un crecimiento mucho más rápido que el de la economía global.

El análisis de IDC destaca que durante la segunda mitad de la década de los noventa se crearon en todo el mundo millones de empleos en el sector (que requerían un alto nivel de formación). De las 28 naciones estudiadas (*), 18 registraron un crecimiento de las cifras de empleo superior al 50 por ciento o superior. De ese grupo de países, el informe de IDC estima que han surgido cerca de 200.000 nuevas empresas tecnológicas. En España, el sector de las TI creó cerca de 190.000 nuevos empleos entre 1995 y 2001, y el número de compañías surgidas en ese intervalo de años fue de 3.600, pasando de un total de 11.318 en 1995 a 14.997 en 2001. Los Web Services basados en XML tienen un desarrollo fuerte en esta industria. Desarrollo y TIs. Dentro de las TI, Internet representa una oportunidad única para muchos países en desarrollo. Los expertos consideran que la Red y las innovaciones generadas por las nuevas tecnologías pueden acercar el volumen de ingresos de sus
Compañías nacionales a las de las naciones desarrolladas, e incluso permitir a esos países saltar estadios de desarrollo tecnológico.

Tal como concluyó un informe de Naciones Unidas: "La revolución de Internet es relevante no sólo para los sectores de alta tecnología e intensivos en información, sino también para la organización de la vida económica en general. Sus efectos positivos se dejan sentir en la mayoría de los sectores de la economía y los países en desarrollo tienen mayores posibilidades de compartir sus beneficios de forma temprana" (Conferencia de Naciones Unidas sobre Comercio y Desarrollo. Informe sobre E-Commerce y Desarrollo 2001).
Por otra parte, el auge de las industrias de las TI es independiente a los ciclos económicos y de las condiciones de cada región del mundo. Crecimientos de este sector medidos en distintos países varían desde el 10 hasta un 59 por ciento anual entre 1995 y 2001.

Y se espera que entre los años 2002 y 2005, se eleve a tasas que van desde un 5,7 por ciento hasta el 26,8 por ciento. En España, los ingresos de las empresas de nuevas tecnologías se doblarán en el periodo que va del 1999 al 2005 y se espera un crecimiento anual sostenido del 13,2 por ciento. En el año 2001 creció un 7,4; este año se espera que aumente al 7,5 por ciento; que lo haga en un 10 por ciento en 2003; en un 11,1 por ciento en 2004 y el 10,2 por ciento en 2005. Ese año, el sector facturará en nuestro país 18.000 millones de euros, frente a 12.479 millones en 2001. También se espera en tres años alcanzar 545.000 empleos generados por la industria de las TI. De forma más desglosada, el sector del hardware crecerá entre 2001 y 2005 el 18 por ciento, con una facturación de 514.000 millones de euros. El de software aumentará el 74 por ciento y generará 382.000 millones. Por su parte, el de servicios mejorará el 53 por ciento con 742.000 millones de aumento.

(*) Argentina, Australia, Bélgica, Brasil, Canadá, Chile, China, Colombia, Costa Rica, República Checa, Francia, Alemania, India, Israel, Italia, Japón, Corea, Luxemburgo, Malasia, México, Holanda, Singapur, República de Sudáfrica, España, Suecia, Taiwán, Reino Unido, Venezuela.

Mientras que el sector privado será el motor de la revolución del XML y el catalizador primario del crecimiento de los web services, las Administraciones Públicas pueden jugar un papel crucial en crear un ambiente en el que los web services y el XML puedan prosperar. En este sentido, un informe de Oxford Analítica, firma internacional de consultoría especializada en investigación económica y social analizó la actuación de diferentes administraciones con respecto a sus políticas de TI. El documento estudia la situación de cuatro países de diferente tamaño, población, nivel de desarrollo, ubicados en diferentes regiones del mundo en los cuales el fomento de las nuevas tecnologías ha sido un punto focal de sus agendas políticas. La consultora analizó las industrias de las nuevas tecnologías de Chile, Estonia, India e Irlanda, mercados donde las TIs han jugado un papel importante en la economía. India, por ejemplo, representa el máximo esfuerzo en la generación de recursos humanos en sus programas universitarios de TI se gradúan más de 70.000 estudiantes por año lo que lo ha convertido en uno de los principales polos de la industria del software en el mundo. Chile se ha distinguido por garantizar un mercado competitivo gracias a una política basada, en gran medida, en la atracción de capital extranjero y privatización del mercado de las telecomunicaciones. Irlanda comenzó en los años 70 un camino de fuerte inversión en la capacitación de su fuerza de trabajo y regular un mercado laboral flexible que multiplica su atractivo para firmas que buscan base en la Unión Europea, especialmente provenientes de Estados Unidos. Por último, la privatización de las telecomunicaciones en Estonia potenció el desarrollo de una red digital que la ha ayudado a ser el país con mayor uso de Internet en proporción con la población. De acuerdo con Oxford Analítica, para conseguir impulsar esa economía digital, los gobiernos deben promulgar y hacer cumplir las leyes de propiedad intelectual; invertir en educación y formación para el uso de las TI; impulsar la comercialización de investigaciones realizadas con fondos públicos, abrir los mercados de las telecomunicaciones a la competencia; apoyar el comercio internacional de bienes y servicios; promover mercados de capital flexibles y competitivos y estimular la inversión. Se puede afirmar que la combinación de estándares XML abiertos a fabricantes y creadores, buenas políticas gubernamentales y las alianzas en la industria de las tecnologías de la información compone una clara promesa de éxito para el futuro.

XML como impulsor de la próxima revolución de las TI

Apertura: XML es una tecnología abierta disponible para todo el mundo sin ningún coste. Microsoft, en cooperación con IBM y otros líderes de la industria, contribuyó al desarrollo de la arquitectura XML básica y la generación de soporte para el estándar dentro de la comunidad de TI, dejando el desarrollo posterior en manos del World Wide Web Consortium (W3C), un organismo líder en la fijación de estándares para Internet.

Estandarización: XML y sus protocolos relacionados han sido adoptados como estándares de la industria. Microsoft ha puesto parte de su tecnología .NET a la disposición de la Asociación Europea de Fabricantes de Ordenadores (ECMA), la principal agrupación sobre estándares. Los miembros de ECMA adoptaron recientemente ambas tecnología como estándares ECMA, lo que los deja completamente disponibles para quien las quiera utilizar.

Interoperabilidad: El éxito de Internet se debe en gran parte a su arquitectura descentralizada y a la simplicidad y ubicuidad de sus principales protocolos. El marco de referencia XML mantiene esas cualidades mientras garantiza la interoperabilidad e interconectividad.

Microsoft, parte implicada

Cuando Bill Gates y Paul Allen fundaron Microsoft en 1975 entendieron que con los productos Microsoft colaboraban a difundir las entonces nuevas tecnologías de la información. Actualmente, los socios de Microsoft en esta aventura son más de 750.000 fabricantes de hardware, desarrolladores de software y proveedores de servicios en todo el mundo. Esta gran familia de partners constituye parte activa de esa nueva economía llena de oportunidades. Su compromiso con Microsoft fomentará los ingresos de estas firmas, reducirá los costes finales y brindará nuevos productos al mercado más rápidamente.

8.5 Metas y Retos de los web services

Metas en los negocios de alto nivel:

- ♣ Reducir costos de integración
- ♣ Incrementar la flexibilidad de la empresa para capitalizar las oportunidades de negocio
- ♣ Reducir el tiempo para lanzar al mercado nuevos productos y servicios
- ♣ Incrementar el retorno sobre la inversión (ROI) para actuales y futuros inversionistas

Retos del entorno actual

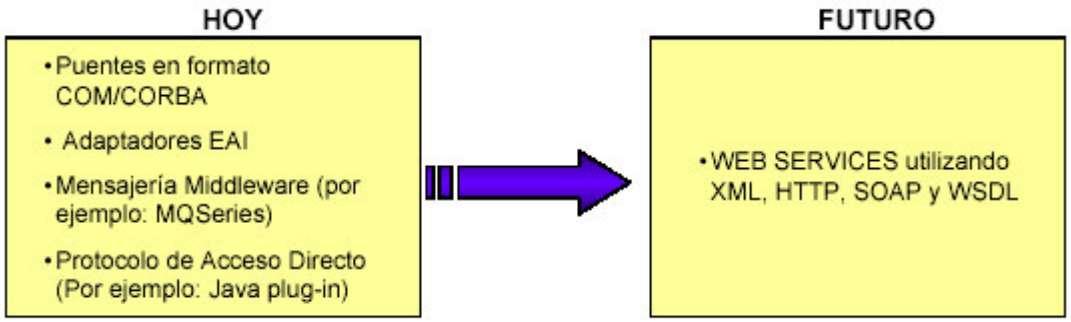
- ♣ incrementar el foco de mercado en el desempeño financiero de las empresas
- ♣ Incrementar en la complejidad del portafolio de aplicaciones existente sin pagar a propietarios de estándares de programación de interfaces.

- ♣ Incrementar el foco en la integración de la información dentro de un entorno de negocio extremadamente dinámico y competitivo.
- ♣ Grandes inversiones en la arquitectura de la tecnología actual. TI.
- ♣ Dificultad para mantener e implementar EAI en una organización distribuida.

El problema desde un punto de vista técnico:

- ♣ Las interfaces de las aplicaciones no son limpias, desde que todo es integrado escribiendo código las interfaces de los módulos pueden definirse incorrectamente o documentarse mal.
- ♣ El desarrollo es una labor extremadamente intensa y extensa, ya que el desarrollo e implantación de aplicaciones requiere de instalación de software, codificación manual y posibles cambios a las tecnologías existentes.
- ♣ Un gran número de aplicaciones heredadas aun existen en el interior de las organizaciones, las empresas han invertido grandes recursos en aplicaciones heredadas que requieren un gran trabajo para la integración con aplicaciones nuevas para poder utilizar su capacidad en los negocios.
- ♣ Las empresas y sus aliados han adoptado múltiples estándares de programación, los distintos lenguajes y estándares entre los ambientes de la organización hacen extremadamente difícil lograr la interoperabilidad entre las aplicaciones y los servicios, en el interior de la empresa y los aliados del negocio.

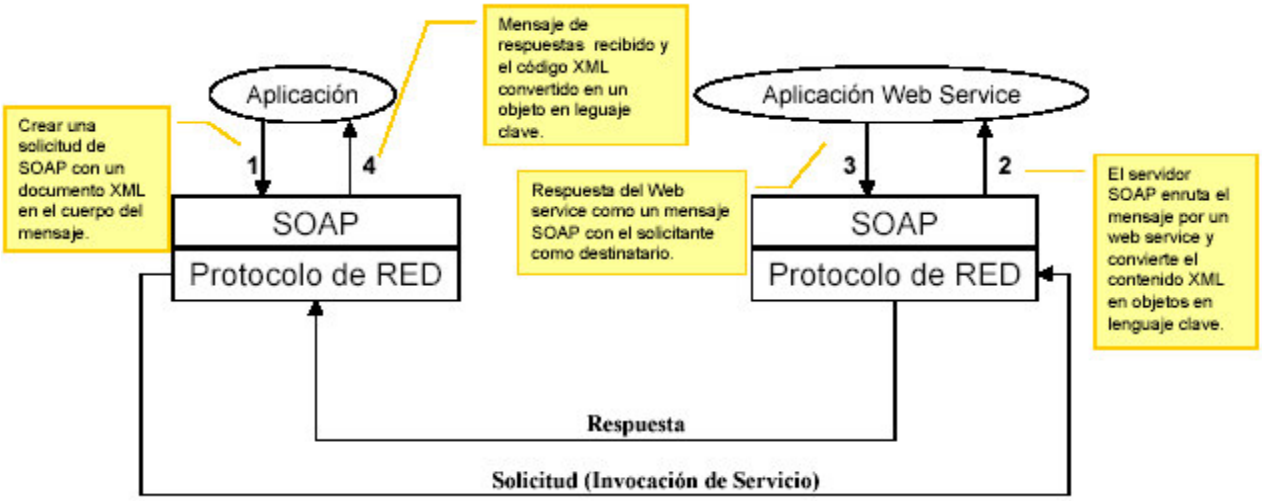
Las estrategias de Integración de Aplicaciones Nuevas (EAI) están girando alrededor de los web services debido a la posición de gran valor que estos ofrecen a las empresas hoy en día.



Adaptar las aplicaciones existentes para que funcionen como web services permitirá que los usuarios construyan nuevas y más poderosas aplicaciones que utilicen web services como una construcción en bloques. Ejemplo, un usuario puede desarrollar una aplicación de compras para obtener automáticamente la información sobre precios de varios vendedores, permitirle al usuario seleccionar el vendedor al que le quiere comprar, enviar una orden de compra y rastrear el envió mientras esta se realiza.
La aplicación de ventas también permitirá al vendedor, además de ofrecer sus servicios en Internet, revisar el crédito del cliente en línea, cargar el monto a la cuenta del cliente y coordinar el envió de la mercancía con una compañía de envíos.

Mensajes XML utilizando SOAP

Los requerimientos básicos para que un nodo pueda funcionar como proveedor o solicitante de mensajes XML de computación distribuida son la habilidad para construir o analizar un mensaje SOAP, o ambas o la habilidad para comunicarse por medio de una red.



Arquitectura orientada a los web services:

Servicio

La aplicación que brindan los proveedores

Proveedor de Servicios

El propietario del Servicio

Solicitante de Servicio

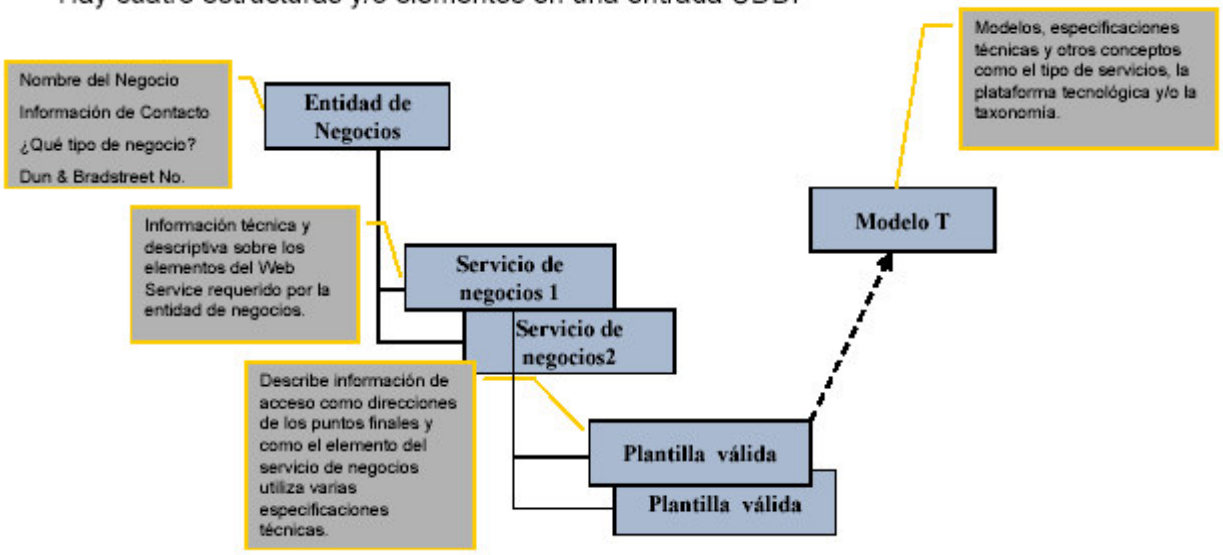
La persona o aplicación buscando o invocando el servicio ofrecido por el proveedor.

Registro UDDI

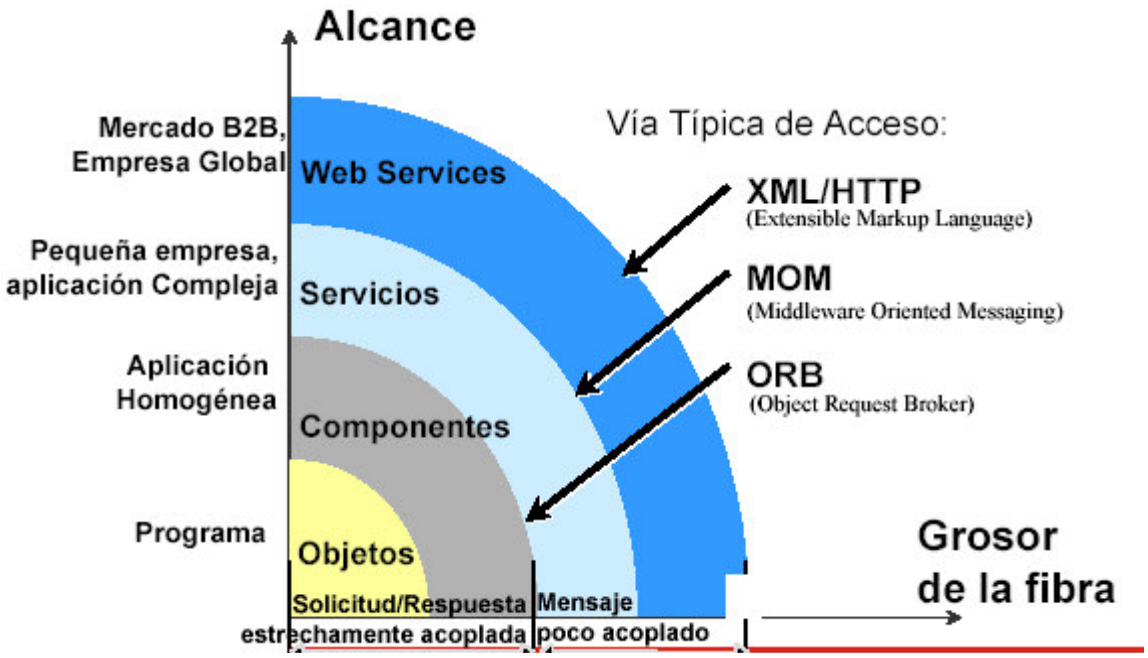
Un repositorio que permite la búsqueda de descripciones WSDL para encontrar servicios y la información relevante a éstos.

Elementos de una entrada UDDI:

Hay cuatro estructuras y/o elementos en una entrada UDDI



Los web services permiten la integración de empresas globales, sus proveedores y aliados de negocios utilizando un sistema en común de mensajes y libre.



Propuesta de valor de los web services:

- Abertura de puentes entre opciones de negocio

Los Web Services conectarán los datos de la cadena de producción directamente con los datos de ventas, para permitirle a las empresas manejar producción en tiempo real y así mismo tomar decisiones en esa misma frecuencia evitando pérdidas.

- Desbloqueo de datos para clientes mejor atendidos.

Utilizando Web Services para levantar una base de datos de perfiles de clientes y un catalogo de productos, las empresas pueden generar precios bajos para contratistas, generando lealtad y haciendo ofertas proactivas basadas en proyectos específicos

- Hacer del outsorcing de aplicaciones algo real.

Utilizando una plataforma de Web Services las empresas pueden crear notas investigativas personalizadas , para generar fidelidad al conectar la fuerza de ventas con el equipo de investigación interno.

- **Para el CEO**

Los Web Services son *estrategia*

El servicio al cliente automatizado puede reducir dramáticamente las solicitudes de soporte al mismo tiempo que mejora la calidad de del servicio y la experiencia del consumidor.

Los tableros de instrumentos ejecutivos brindan una ventana de desempeño en tiempo real.

- **Para el COO**

Los Web Services son *productividad*

Construir portales basados en roles, que pongan la información necesaria en las manos de los empleados, vincular el desempeño del empleado con datos en tiempo real, mejorar el tiempo de mercadeo vinculando a los asociados en el proceso de desarrollo; subcontratar servicios para los negocios ajenos a las metas de la organización.

- **Para el CFO**

Los Web Services son *renovación*

Cortar los costos de transacción con logrso directos y automatizados; reducir las solicitudes de soporte con servicio al cliente automatizado; reemplazar persobnal administrativo con interfaces de auto servicio.

- **Para el CIO**

Los Web Services son *control*

Transformar las funciones de las TI en servicios tecnológicos; protegerse contra las fugas de seguridad administrando el acceso al registro de las interfaces y al flujo de mensajes SOAP.

- **Para el CMO**

Los Web Services son *influencia*

Poner la marca corporativa en interfaces SOAP; anticiparse a las necesidades de los clientes; emparentar las interfaces con el browser.

Las oportunidades para implementar web services deben cubrir tres criterios básicos:

Recurrente: Es sindicado y repetido a través de una gran base de usuarios

Las empresas pueden ofrecer herramientas par el manejo de garantías basadas en Web Services para miles de usuarios y distribuidores a través de un mecanismo de salida para registrar, rastrear e interponer quejas.	Compartir diseños entre ingenieros de Compaq e Intel no tendría sentido porque la colaboración en ese caso seria de persona a persona y única, no masiva.

Dinámico: Información que cambia constantemente y de cuya actualización los usuarios se benefician constantemente.

BP Amoco puede implementar un Web service Para brindar herramientas de manejo de riesgo como cambios para ayudar a los clientes mitigar las fluctuaciones de los precios.	DuPont publicando tablas de datos químicos que muy raramente cambian y tienen poco valor para los clientes.

Desconectado: Hace un puente para una realción de negocios que no esté conectada por Internet o vía EDI.

La herramienta de rastreo de entregas de Ford Motor Company utiliza un web service para vincular los sistemas logísticos de UPS a la red expansiva de Ford de distribuidores individuales.	Los Web services no le ayudan a Ford a ahorrar dinero intercambiando datos de con abastecedores como Delphi Automotive Systems, porque para este tipo de transacciones ya existe una red de intercambio de datos EDI.

Las tres caras de los web services:

			
	WS entre Socios	**WS Privados**	**WS Públicos**
¿Qué es?	**Conecta clientes conocidos, socios y unidades remotas de negocios.**	**Integración del personal interno con las aplicaciones heredadas en una empresa pequeña.**	**Provee para encontrar y enganchar nuevos socios.**
Beneficios Clave	■ **Brinda conexiones de bajo costo para B2B y un mejor flujo de información entre asociados.** ■ **Reduce la fricción generada por el traslado y paso de información entre múltiples hileras de de la cadena de socios proveedores**	■ **Brinda un estándar para aplicaciones internas, integración y adjuntar datos para servicios** ■ **Brinda conexiones de bajo costo entre aplicaciones.**	■ **Provee un directorio para encontrar nuevos socios y colabora con la base ad hoc** ■ **Provee la habilidad apra enganchar rápidamente colaborar y desenganchar socios**
¿Dónde Aplica?	■ **Cadenas de proveedores ampliamente fragmentadas.** ■ **Productos Complejos** ■ **Conectividad limitada entre firmas pequeñas**	■ **Muchos sistemas internos** ■ **Numerosas locaciones** ■ **Confianza al heredar soluciones que requieren integración interna.**	■ **Ciclo de vida del producto corto** ■ **Pocas barreras ara ingresar** ■ **Relaciones de negocios dinámicas**

	Socio	Privado	Público	Comentarios
Sector Público	**Alta**	**Media**	**Alta**	**Inicialmente de uso interno para extraer las apliciones de las diferencias entre plataformas. Gradual evolution into G2G and G2B use to streamline information fBaja.**
Servicios de Negocios	**Baja**	**Baja**	**Alta**	**Los servicios de mercadeo y consultoría utilizarán Web services para encontrar y ser encontrados por clientes.**
Computación y electrónica	**Alta**	**Media**	**Media**	**La firmas compartirán directamente datos de proveedores y productores para administrar el inventario y el flujo de caja.**
Consumo de Productos empacados	**Alta**	**Alta**	**Baja**	**Las Grandes inversiones en la integración de B2B para el 2004 traerán servicos de socios a un costo más bajo.**
Servicios Financieros/Seguros	**Media**	**Alta**	**Media**	**Las Instituciones financieras utilzarán Web services privados para integrar canales y productos.**

	Socio	Privado	Público	Comentarios
Vehiculos a motor	Alta	Media	Baja	Muchas marcas utilizan Web services para conectarse con distribuidores, proveedores, vendedores y consumidores.
Petroleo y Químicos	Media	Baja	Baja	Muchos distribuidores publican sus Web services como socios para automatizar la facturación con los refinadores.
Telecomunicaciones	Media	Alta	Baja	Los Web services privados introducirpan servicios al consumidor desde una simple cadena (venue)
Transporte	Media	Media	Alta	La necesidad de manejar capacidad de enbarque en tiempo real los lleva´ra a utilizar Web Services públicos.
Utilitarios	Baja	Media	Media	Los utilitarios no utilizan muchos servicios para socios porque intercamibos como la ICE proveen información negociando servicios de contratación

Tendencias de los web services:

Los vendedores de aplicaciones corporativas (ORACLE, SAP, SIEBEL) ya están desarrollando su propia arquitectura alrededor de los web services.
La mayoría de los sistemas desarrollados utilizan XML, SOAP, y algunas veces WSDL.
El mantenimiento de documentos WSDL y otros XML para web services se convertirán en un requerimiento importante para los vendedores de administración de contenido empresarial.

8.6 Implementación de Seguridad para web services

Los protocolos de transporte y de red que utilizan los web services para intercambiar información son los que ya se usan en Internet, TCP/IP. Este protocolo se creó en una época y en una situación donde la seguridad no era algo que tuviera una gran importancia.
En la actualidad y con la filosofía de los web services, se traslada a un entorno distribuido, en el que las aplicaciones pueden interaccionar entre sí. Pero a pesar de que suponga una gran ventaja y un gran paso, el poder publicar servicios que sean accesibles a todo el mundo representa un problema en seguridad y comienzan a surgir preguntas sobre cómo proteger estos servicios, cómo proteger al servidor y la red interna donde se encuentre este servidor de posibles ataques externos, cómo autenticar a los usuarios, qué usuarios pueden acceder a unos servicios u otros (roles de usuario), etc.
La torre de protocolos TCP/IP es bastante robusta, pero no está prevista para la seguridad (Ej. Autentificación, verificación, cifrado, etc. esto es cierto en IPv4, no así en IPv6 en el

que se han preocupado de dar una solución a este tema, pero al no estar todavía implantada esta nueva versión del protocolo se seguirá haciendo alusión a la versión vigente). Hacer spoofing de paquetes, interceptar paquetes, leer los datos de los paquetes, etc., es algo bastante fácil en el Internet de hoy en día. Los ataques más comunes son los ataques de negación de servicio, ya que son los más fáciles de ejecutar y los más difíciles de impedir, seguidos del sniffing de paquetes, escaneo de puertos y otras actividades relacionadas.

- Autenticación: se debe asegurar la autenticación tanto del usuario que invoca los web services como del proveedor de los servicios. Además esta autenticación debe ser única, es decir, que el usuario solo tenga que identificarse una vez para acceder a los distintos servicios (aunque estos estén distribuidos en multitud de servidores web)

- Definición de perfiles de usuario: se debe permitir el acceso a los recursos dependiendo del rol que ocupe ese usuario dentro de nuestro servicio web: acceso de clientes, proveedores, vendedores, competidores, hackers, administradores.

- Confidencialidad: se debe garantizar la confidencialidad de los datos intercambiados, ya que SOAP no cifra la información, viajando de este modo en claro por la red

- Integridad: se debe igualmente garantizar la integridad de los datos protegiéndolos de cualquier alteración por parte de terceros (cualquiera podría estar viendo las transacciones que realizamos y modificarlas sin problema)

- Disponibilidad del sistema y no repudio de las conexiones: las empresas deben asegurarse de que sus sistemas están disponibles y deben tener medios para saber quién ha invocado sus servicios en cualquier momento

No hay una única solución para la seguridad de Internet. Todas ellas requieren muchos elementos, incluyendo una política de seguridad y estándares que definan qué es lo que se quiera proteger, una serie de procedimientos que detallen cómo implementar la política y un conjunto de tecnologías que proporcionen protección.

La confidencialidad, integridad y autenticación son servicios que se usan para protegerse contra las amenazas. Si los datos están cifrados mientras viajan por la red, es imposible que un intruso los pueda observar o modificar. Las otras amenazas como la pérdida de identidad y la pérdida de disponibilidad, se pueden prevenir con una autenticación a nivel de red. Si los dispositivos pueden identificar la fuente o el origen de los datos, entonces es más difícil que un impostor pueda realizar la denegación de servicio. Las formas de que disponemos para implementar la seguridad a los web services se pueden clasificar en dos grandes grupos: por un lado está la seguridad extrínseca a los servicios, que es la que se ha venido utilizando en el resto de accesos hasta ahora. Y por otro lado, aparecen especificaciones (a falta de convertirse en estándares) que introducen el concepto de la seguridad intrínseca al servicio, son nuevos modelos de seguridad que se están desarrollando para proporcionar ese mayor nivel de seguridad que necesitan este tipo de servicios.

Seguridad extrínseca de los web services

Firewall de aplicación y VPN: la comunicación con los web services se realiza a través del puerto 80 ó 443, el filtrado de puertos tradicional que hacen los firewall de nivel de red no sirve de gran ayuda. Con los firewall de aplicaciones XML que actúan a nivel de aplicación será capaz de comprender los web services, las peticiones de servicio y el contenido de los mensajes XML intercambiados. Si se combina con los firewall tradicionales se puede mejorar en gran medida la seguridad de los web services ofrecidos.
En el caso de el web service tuviera unos clientes muy localizados podría optar en temas de seguridad por la que ofrece las conexiones a través de VPN (Virtual Private Networks).

Para poder brindar seguridad a este aspecto debe pensar en las técnicas de cifrado de datos

- Cifrado: La gran mayoría de los web services basan su seguridad y autenticación en el protocolo SSL (Secure Sockets Layer) que proporciona privacidad y fiabilidad entre el cliente y el servidor web. SSL es un protocolo independiente del nivel de aplicación. Proporciona una conexión segura que tiene tres grandes características:

  - Privacidad: el cifrado se usa después de un acuerdo inicial en el que se definen las claves secretas. Se usa criptografía simétrica para cifrar los datos (DES, 3DES...)
  - Autenticación: el servidor web puede autenticar al cliente que invoca sus servicios usando técnicas criptográficas asimétricas o de clave pública
  - Fiabilidad: en el transporte de los datos se incluye un chequeo de la integridad de los mismos, usando una clave MAC.

- Infraestructura de PKI y CRL: el uso más común de la clave pública es la firma digital, y se utiliza para prestar servicios como la autenticación de usuarios (para asegurarse de la identidad de un usuario, bien como firmante de documentos o para garantizar el acceso a servicios distribuidos en red, ya que sólo él puede conocer su clave privada, evitando así la suplantación), el no rechazo (para impedir que una vez firmado un documento el signatario se retracte o niegue haberlo redactado), la integridad de la información (para prevenir la modificación deliberada o accidental de los datos firmados, durante su transporte, almacenamiento o manipulación), la auditabilidad (para identificar y rastrear las operaciones, especialmente cuando se incorpora la marca de tiempo), y el acuerdo de claves secretas para garantizar la confidencialidad de la información intercambiada, esté firmada o no. La idea básica de una infraestructura de clave pública es que los datos sensibles sean protegidos mediante técnicas de cifrado.

- Como los certificados no deben durar eternamente, se editan con una fecha de expiración, a pesar de que en ocasiones deben ser revocados inmediatamente, como cuando un empleado abandona la empresa. Para tratar estas situaciones, la autoridad mantiene actualizada una lista de revocación de certificados (Certificate Revocation List – CRL), que el servicio web podrá consultar para asegurarse de que un determinado certificado sigue siendo válido justo en ese momento.

- Seguridad a nivel de mensaje: cada mensaje que se intercambia (a través de SOAP) realiza múltiples saltos y se enruta a través de numerosos puntos hasta alcanzar su destino. El conjunto de técnicas que se pueden usar para garantizar la seguridad a nivel de mensaje es el siguiente:

  - Cifrado XML: Evita que los datos se vean expuestos a lo largo de su recorrido

  - Firma Digital XML (XML Dsig): Asocia los datos del mensaje al usuario que emite la firma, de modo que este usuario es el único que puede modificar dichos datos. La firma digital es importante porque permite asegurar la integridad, la autenticidad y el no repudio de los datos. Cada persona que desee crear una firma digital debe tener un único par de claves; una de estas claves es la clave privada y se usa para firmar y la otra o clave pública se utiliza para autenticar al propietario de la firma.

  - XKMS y Certificados: XKMS (XML Key Management Specification) define web services que se pueden usar para chequear la confianza de un certificado de usuario. La especificación XKMS (XML Key Management) tiene como objetivo implementar la seguridad a los web services, permitiéndoles registrar y gestionar claves criptográficas usadas para firmas digitales y cifradas. Define una interface de web service de infraestructura de clave pública, facilitando así las aplicaciones relacionadas con registro y revocación, localización y validación de claves y de usuarios. Los componentes de un servidor XKMS se implementan en su gran mayoría en los proveedores de infraestructura de clave pública (PKI)

  - SAML y Autorización: SAML (Security Assertion Mark-up Language) hace posible que los web services intercambien información de autentificación y autorización entre ellos, de modo que un web service confíe en un usuario autentificado por otro servicio web. Este lenguaje lo usa la arquitectura "single sign-on" (SSO), en la que todos los algoritmos de seguridad se encuentran en un sólo servidor SSO que actúa como el único punto de autentificación para un dominio definido

  - Validación de los datos: Permite que los web services reciban datos dentro de los rangos esperados

Seguridad intrínseca de los web services

Los creadores SOAP eligieron no definir cómo resolver el problema de la seguridad con el fin de mantener el protocolo lo más simple posible, y el proporcionar seguridad no siempre lo es. Sin embargo, como este problema no puede estar sin solución en estos días, Microsoft, IBM y SUN deciden unirse para darle una. Se trata de estandarizar todas las tecnologías de seguridad existentes, como kerberos, tecnologías de clave pública. Adaptándolas e integrándolas al mundo de los web services. Así es como ha surgido WS-

Security, una especificación de seguridad diseñada para los web services, que permitirá cifrar la información y asegurar que los datos intercambiados entre el cliente y el servidor sean confidenciales.
Surgen a partir de esta especificación ocho estándares más de seguridad. WS-Policy, WS-Trust y WS-Privacy se centran en el marco de la certificación, compartición y confidencialidad de la información. WS-Secure Conversation, WS-Federation y WS-Authorization tienen como objetivo mejorar la interoperabilidad entre las distintas plataformas y sistemas. Y por último se encuentra WS-ReliableMessaging y WS-Addressing de muy reciente aparición.
Existen algunas especificaciones definidas más o en proceso de definición como WS-Coordination, WS-AtomicTransaction, WS-BusinessActivity y WS- Eventing, entre otras.

- Especificación WS-Security: La identidad, integridad y seguridad del mensaje y del cliente deben mantenerse en todos los saltos que de el mensaje hasta llegar a su destinatario. Los web services requieren de una solución que garantice la seguridad de extremo a extremo y esto es lo que trata de estandarizar la especificación WS-Security, cómo incluir seguridad al mensaje SOAP. No especifica la forma de la firma ni del cifrado ni de la autenticación, sólo cómo se puede embeber la información utilizada por estas tecnologías en el mensaje SOAP, con el fin de que más tarde el destinatario, y solo él, pueda entender su contenido

  - Autenticación: hay muchas maneras de identificarse en una red, podemos utilizar un nombre de usuario y un password, o los tickets de kerberos o bien los certificados de X.509. Dependiendo de qué método se elija también habrá distintas maneras de autenticar esa identidad.

  - Integridad: la solución dada por WS-Security se basa en la Firma Digital, que, como hemos visto, es un estándar ya existente, especificando cómo incluirlo en la cabecera SOAP.

  - Confidencialidad: WS-Security hace uso del estándar de cifrado XML, de forma que solo el destinatario del mensaje pueda leer los datos. WS-Security permite, por medio de este estándar, cifrar todo o parte de la cabecera del mensaje SOAP, el cuerpo del mensaje e incluso sus datos adjuntos.

- Especificación WS-Policy: expresa las capacidades, requisitos y características generales de las entidades de un sistema de web services basados en XML. Define un marco y un modelo para expresar estas propiedades como políticas. Un proveedor de servicio web generalmente tiene condiciones bajo las cuales proporciona el servicio y el que invoca el servicio puede usar esta información para decidir si usar el servicio o no.

- Especificación WS-Trust: define cómo adquirir tokens de seguridad. Los tokens son una parte fundamental de la especificación WS-Security. Para adquirir un ticket kerberos se debe solicitar a un Centro de Distribución de Claves Kerberos (KDC), mientras que para conseguir un certificado X.509 se debe solicitar a una autoridad

certificadora (CA). WS-Security define cómo embeber los tickets kerberos y los certificados X.509 dentro del estándar XML para la transmisión en un mensaje SOAP, no define la forma de usar SOAP para hablar con un KDC o una CA o cualquier otro servicio capaz de proporcionar un token, y este agujero es lo que trata de cubrir la especificación WS-Trust.

- Especificación WS-Privacy: describirá la forma de embeber privacidad en las descripciones de WS-Policy y cómo debe utilizarse WS-Security para asegurar la privacidad de partes de los mensajes.

- Especificación WS-SecureConversation: hay veces que para dar por finalizada una conversación entre cliente y servidor web basta con el intercambio de uno o dos mensajes SOAP, pero existen aplicaciones que necesitan intercambiarse gran cantidad de mensajes SOAP entre sí porque invocan una serie de operaciones. En estos casos es útil crear un contexto seguro compartido entre ambas aplicaciones. Algo que podría definir este contexto compartido sería un tiempo de vida para la clave de cifrado que se usará mientras dure el contexto. WS-SecureConversation define tipos de XML e iteraciones que permiten establecer un contexto seguro y la creación de claves específicas para ese contexto, es decir, claves que solo se usarán para una sesión en particular.

- Especificación WS-Federation: Esta especificación pretende definir mecanismos para gestionar la identificación, el registro, la autenticación y la autorización de los usuarios, claves, a través de diferentes dominios y web services. Permite unirse a estos dominios de seguridad, que usan los mismos o distintos mecanismos, para aceptar o rechazar identidades o atributos entre los web services participantes. Los mecanismos que define pueden utilizarlos participantes activos y pasivos. Estos participantes deben poder entender dichos mecanismos de seguridad y ser capaces de interactuar con diferentes proveedores de web services.

La seguridad que implantan las empresas para los web services ha consistido en una seguridad extrínseca, poniendo los servidores detrás de firewall que impidan el acceso a posibles intrusos, haciendo uso de las listas de acceso en los routers, utilizando redes privadas virtuales (VPN) cuando se conoce muy bien a los usuarios que van a invocar los servicios, accediendo con protocolos como HTTPS, autenticando a los usuarios que invocan los servicios a través de infraestructuras de clave pública que permitirán comprobar la vigencia de sus certificados, cifrando la información, definiendo roles de usuario, etc. El problema que este tipo de seguridad supone es que todas estas tecnologías han sido atacadas con anterioridad con lo que volvemos a encontrar vulnerabilidades que hacen inseguro al servidor web.

Quizás sea una de las causas por las que los web services estén tardando más en entrar en el mercado, o que la mayor parte de ellos sean para comunicar dos empresas entre sí, o para el acceso de sus usuarios internos, es decir que se restringe su uso a usuarios conocidos.

Si desea obtener más información sobre aspectos de seguridad en los web services puede consultar:

- Security in a Web Services World: A Proposed Architecture and Roadmap http://msdn.microsoft.com/library/?url=/library/en-us/dnwssecur/html/securitywhitepaper.asp?frame=true
- Web-Services: La Siguiente Generación De Internet http://revista.robotiker.com/articulos/articulo62/pagina1.jsp
- La seguridad en web services http://www.willydev.net/descargas/prev/WSSev.pdf
- Seguridad en web services http://www.criptonomicon.com/documentos/2004/20040610.html
- Una visión general II http://www.desarrolloweb.com/articulos/1538.php?manual=54
- ¿Son seguros los Servicios Web XML? http://www.desarrolloweb.com/articulos/1611.php?manual=54
- Seguridad en servicios web. Autenticación y autorización. Interoperabilidad http://www.desarrolloweb.com/articulos/1640.php?manual=54
- SecurityGuide - Chapter10 - Web Services Security http://www.willydev.net/Seguridad/Cap%C3%ADtulo%2010%20Seguridad%20de%20servicios%20Web.doc
- Seguridad en servicios web XML http://www.iec.csic.es/criptonomicon/susurros/susurros37.html
- XML Web Services Security Forum http://www.xwss.org/
- WS-Security http://www.verisign.com/wss/wss.pdf
- XML-Signature Syntax and Processing http://www.w3.org/TR/xmldsig-core/
- XML Encryption Syntax and Processing http://www.w3.org/TR/xmlenc-core/
- XML Web Services Security http://msdn.microsoft.com/vstudio/techinfo/articles/XMLwebservices/security.asp
- Transacciones seguras (I) http://www.htmlweb.net/seguridad/ssl/ssl_1.html
- SSL vs. S-HTTP http://www.iec.csic.es/criptonomicon/sslvsshttp.html
- Seguridad SSL http://www.idg.es/iworld/articulo.asp?id=68235&sec=iworld
- ¿Qué es la Tecnología SSL? http://www.ichotelsgroup.com/h/d/pc/1/es/c/3/content/dec/cn/0/es/tc/pp/ssl.html
- SSL Seguridad fácil y versátil http://www.mipunto.com/temas/3er_trimestre04/ssl.html
- Encriptación Criptografía y algoritmos de encriptación http://www.cryptoforge.com.ar/seguridad.htm
- Seguridad Encriptación http://www.osmosislatina.com/aplicaciones/seguridad.htm
- La firma digital: aspectos técnicos y legales http://www.marketingycomercio.com/numero14/00abr_firmadigital.htm
- Cómo afrontar la Seguridad en Redes Abiertas: Consideraciones Técnicas y Escenarios. http://www.tecnimap.com/ES/iframes/pdfs/c3.1-81.pdf

- Comprobación de la integridad de los ficheros con md5 http://bulma.net/body.phtml?nIdNoticia=1241
- Acceso remoto por VPN (Red privada virtual) http://www.uv.es/ciuv/cas/vpn/
- Redes Privadas Virtuales http://www.sci.uma.es/sistemas/servinst/vpn/
- VPN/Seguridad http://www.cisco.com/global/LA/LATAM/sne/pc/tecnologia/seguridad/index.shtml
- Implementando VPN en la empresa - Primera Parte http://www.symantec.com/region/mx/enterprisesecurity/content/expert/LAM_1564.html
- Estudio Preliminar de las VPN http://www.symantec.com/region/mx/smallbusiness/articles/vpn.html

9. APACHE WEB SERVER.

Uno de los servidores Web más utilizados en la actualidad es Apache, el cual se puede utilizar tanto en plataformas Windows como Unix/Linux. Apache es un servidor open source, y el más usado por los servidores en todo Internet, puede encontrar toda la información sobre Apache en su página web: http://www.apache.org/.

El funcionamiento de Apache viene es igual sin importar el sistema operativo en el que se ejecute, así que puede practicar con su computadora en el hogar o en el trabajo y ya cuando tenga experiencia con el servidor Apache, la programación en un lenguaje como Perl o PHP y CGI, podrá aplicar los conocimientos en cualquier servidor Web sin importar si se ejecuta en Windows, Unix o Linux, si está conectado a Internet o sólo a una red interna de la compañía (lo que se conoce comúnmente como Intranet) o si sólo funciona en una computadora sin conexión de red.

9.1 Instalación de Apache 2 en un Sistema Unix.

Primeramente deberá revisar que su sistema no tenga instalado Apache, este caso aplica especialmente a las distribuciones de Linux las cuales generalmente instalan Apache 2 vía un RPM al montar el Sistema Operativo.

La desventaja en este tipo de instalación prematura es que su estructura puede variar dependiendo de su distribuidor (Red Hat, Mandrake, Debian, Suse u otro), esto es, Red Hat puede incluir los archivos de configuración bajo el directorio /usr/local/apache2, mientras Mandrake en /usr/local/httpd y Debian en /usr/share/apache; para eliminar este RPM se puede ejecutar: rpm -e apache (o dependiendo del sistema rpm -e httpd).

Para dar mayor uniformidad a esta guía, se compilará el Código Fuente de Apache 2 directamente, en efecto aprovechando una de las principales cualidades del Software Open-Source.

Instalación Básica

1. Una vez obtenido el archivo Tar de http://httpd.apache.org/que contiene el Código Fuente (Source-Code) de Apache 2, este debe ser descomprimido en un directorio temporal (/tmp por lo general) para poder iniciar la instalación.

2. El paso anterior genera un directorio por nombre apache-<numero_de_version> dentro del directorio temporal (/tmp), descienda a este directorio y ejecute el comando: *./configure --prefix=/usr/local/apache2*

3. Este paso configura el código fuente para que Apache 2 sea instalado bajo el directorio /usr/local/apache2.

4. Posteriormente debe ejecutar : *make; make install* Lo anterior compila e instala Apache 2 bajo el directorio /usr/local/apache2

Aunque Apache 2 ya esta instalado bajo el directorio /usr/local/apache2 se recomienda movilizar todo el código fuente de Apache 2 (apache-<numero_de_version> del paso 2) también al directorio /usr/local/apache2, y renombrar el directorio a fuente o source, esto resulta esencial cuando se intenten instalar Módulos en Apache.

También es recomendable modificar la variable ambiental PATH del sistema en /etc/bashrc agregando /usr/local/apache2/bin; esto garantiza que cualquier ejecutable de Apache 2 (apachectl, apxs) este disponible directamente del Shell.

Módulos

Un Módulo en Apache es una manera de agrupar y organizar ciertos funcionamientos para el Servidor, existen una gran cantidad de Módulos para utilizarse con Apache, algunos son: "PHP4", "Virtual Hosting", "Mod_JK (Java)" entre otros, una lista se encuentra en: *http://modules.apache.org*.

Una de las principales razones de emplear módulos en Apache, es que no toda instalación requiere de las mismas funcionalidades, esto es, una instalación que utilice PHP probablemente no requiera de Tomcat (Java), o bien posiblemente no todas las instalaciones requieran de "Virtual Hosting".

Por lo tanto, si fueran incluidas todas las funcionalidades posibles en una versión única de Apache, esto lo haría sumamente pesado en cuanto a requerimientos de Memoria RAM y espacio en Disco Duro, por esto se opta por modularizar e incluir solo lo necesario. Para verificar cuales son los módulos que se encuentran instalados en el Sistema se puede ejecutar el comando httpd -l, este comando despliega algo como:

http_core.c
mod_env.c
mod_log_config.c
..
..

Lo anterior representa los módulos "Default" incluidos al instalar y compilar Apache.

TIPOS DE MÓDULOS.

Los módulos son distribuidos de dos maneras:

- En el código fuente de Apache: Estos están incluidos en el Tar principal de Apache. (Ejemplo: "Virtual Hosting","Rewrite").

- Por 3eros: Se distribuyen en productos utilizados en conjunción con Apache. (Ejemplo: PHP4).

9.2 En Código Fuente

Al instalar y compilar Apache se recomendó trasladar el código fuente a un directorio llamado fuente o source esto se debió precisamente a que varios módulos residen aquí. Dentro de este directorio (fuente o source) reside un archivo llamado config.status el cual contiene información para integrar módulos en Apache. Se recomienda no modificar manualmente este archivo, sino a través de la linea de comandos.

Instalación de Módulos.

A diferencia de Apache 1.x, esta nueva versión ya posee una serie de módulos integrados al momento de realizarse su instalación inicial, entre ellos el módulo para módulos (mod_so).

No obstante, diversos módulos requieren ser activados para ser utilizados en conjunción de Apache.

Ejecución.

Al igual que la versión 1.x de Apache, el comando llamado apachectl facilita el arranque y terminación de Apache.

1. apachectl start : Inicia el Servidor de Páginas
2. apachectl stop : Termina el Servidor Apache
3. apachectl restart : Re-Inicializa el proceso Apache

Al momento de ejecutar cualquier variación de apachectl, se lee el archivo principal de configuración de Apache, httpd.conf, ubicado en el directorio /usr/local/apache2/conf.

Después de la instalación inicial, el archivo httpd.conf contiene valores razonables de ejecución, sin embargo, en ocasiones es necesario modificar ciertos parámetros.

VALIDACIÓN, # Y VALORES "DEFAULT"

Todos los parámetros que se incluyan en httpd.conf serán validados previo arranque de Apache, esto es, al ejecutar apachectl start (o apachectl restart); la única excepción a esto son los renglones que inicien con el signo: #, ya que estos indican un comentario (al igual que otros archivos en sistemas *nix).

USER, GROUP, SERVERNAME, LISTEN

Los parámetros User y Group indican el Usuario y Grupo al cual pertenece el proceso de Apache, estos parámetros básicamente indican que usuario será capaz de inicializar y terminar el Servidor Apache, por lo general se recomienda generar un usuario y grupo especial para esta tarea.

User web - Group web.
El parámetro ServerName indica el nombre del Servidor que administra Apache, en otras palabras el sitio en cuestión; esto puede ser www.miservidor.com, u otro.

Este parámetro no puede ser inventado y dependerá fuertemente de su configuración DNS, sin embargo, si solo esta instalando Apache en un ambiente local ("Workstation","Intranet") se puede realizar sin necesidad de DNS.

Para realizarse en ambiente local es necesario modificar el archivo /etc/hosts para que pueda realizarse la resolución correspondiente, un ejemplo seria: 127.0.0.1 www.miservidor.com

Lo anterior indica que la maquina local (127.0.0.1) también puede ser llamada: www.miservidor.com. Una vez configurado DNS o la resolución local, se puede definir el parámetro ServerName dentro de httpd.conf, el parámetro Listen indica el puerto TCP sobre el cual responderá Apache, este valor en casi todo ambiente "Web" toma el ampliamente conocido Puerto 80.

ServerName www.miservidor.com
Listen 80
DocumentRoot, DirectoryIndex

Los pasos anteriores indican a Apache el nombre del sitio que se administra , sin embargo, aun falta indicar donde se encuentra el contenido de este sitio , esto es, cuando se visite www.miservidor.com que aparecerá en Pantalla ? . Hasta este punto si ejecuta Apache y abre su navegador ("Netscape", "Lynx", "Opera" u otro) e intenta visitar el sitio definido en ServerName debe observar la página inicial de Documentación para Apache, pero como se pueden observar otros documentos ?

El parámetro DocumentRoot indica el directorio local donde reside la información del sitio en cuestión, el valor "default" en httpd.conf es /usr/local/apache2/htdocs, en este directorio

radica la documentación de Apache, desde luego es claro que debe modificar este parámetro hacia el directorio donde reside su propia documentación en HTML.

El parámetro DirectoryIndex indica el Documento que debe ser enviado al accesar un directorio, generalmente toma el valor de index.html, inclusive puede tomar varios valores, esto es especialmente útil cuando se emplean ambientes más elaborados con Perl o Java.

DocumentRoot /usr/local/misitio/
DirecotryIndex index.html, index.htm, index.php, home.html

Lo anterior indica que al intentarse accesar cualquier directorio bajo /usr/local/misitio (siempre y cuando no se indique un archivo especifico), se intentará enviar el archivo index.html, si este no existe, se intenta con index.htm, seguido de index.php, y finalmente si no existiesen ninguno de los anteriores enviar home.html.

CustomLog y ErrorLog - Registros ("Logs")

Apache puede registrar los accesos y errores del sitio administrado , estos registros son los que ofrecen estadísticas sobre el número de visitantes, el origen de cada usuario (Nodos IP), los posibles errores, etc.

Los dos parámetros son CustomLog y ErrorLog los cuales se recomienda sean modificados a un directorio relacionado con DocumentRoot

Analice estos registros con Analog .

SCRIPTALIAS

Este parámetro es requerido para sitios que utilizan "Scripts"(Programas), generalmente escritos en Perl , estos "Scripts"(Programas) son los que permiten la generación de información dinámica en un sitio de Internet.

SCRIPTALIAS /CGI-BIN/ "/USR/LOCAL/APACHE2/CGI-BIN/"

Lo anterior indica que cualquier solicitud bajo el directorio cgi-bin, será atendida por el directorio /usr/local/apache2/cgi-bin/, es dentro de este directorio donde se encontraran los "Scripts" que ejecutará Apache.

Aparentemente parece excesivo este tratamiento, porque no simplemente basarse en el antes definido DocumentRoot ? , la primer razón es Administrativa ya que estos "Scripts"(Programas) tienen mayor probabilidad de generar un error ya que son programas, no documentos estáticos; y la segunda razón es que estos "Scripts" pueden ser compartidos por varios "Sitios", a través "Virtual Hosting".

9.3 Ejemplo de Instalación del servidor web Apache.

Este ejemplo muestra como instalar un servidor web en el servidor "pinero" y que los usuarios publiquen y gestionen sus páginas web. Los archivos de configuración están en /etc/httpd/conf, aquí se encuentran los archivos httpd.conf donde se especifica la configuración general, srm.conf para configurar recursos y access.conf para configurar permisos, como sólo es necesario httpd.conf y resulta más cómodo manejar un archivo que tres, se fusionan con:

 # cat srm.conf >> httpd.conf
 # cat access.conf >> httpd.conf

Configuración

Se quiero tener la siguiente estructura de páginas web, en http:\\www.pinero\index.html estará la página principal y en http:\\www.pinero\~nombre_usuario, estarán las páginas de los usuarios:

/home/httpd/html/index.html
garza/index.html
paco/index.html
lolo/index.html

En /home/httpd/html/ se crea un subdirectorio por cada usuario que va a publicar páginas web, con permisos 755 y como propietario del directorio el usuario. El umask de cada usuario debe ser 022, así los directorios se crearán con permisos 755 y los archivos con 644 y todos podrán ver todas las páginas.

En httpd.conf modificar lo siguiente:
User nobody
Group nobody
ServerAdmin garza@pinero -- email del usuario administrador de apache
ServerName pinero -- nombre del servidor
ServerRoot /etc/httpd -- directorio donde están los archivos de configuración y logs

Por motivos de seguridad es conveniente asignar como usuario y grupo a nobody, ya que si atacasen el sistema, aprovechando cualquier bug de apache, el atacante entraría en el sistema como nobody, que se trata de un usuario con pocos privilegios. Cuando un usuario navega por las páginas del servidor web, lo hace con los privilegios de nobody.

Especificar donde alojar las páginas:
DocumentRoot /home/httpd/html -- lugar donde se van a almacenar las páginas, corresponde a httpd://pinero/index.html
UserDir public_html /home/httpd/html -- páginas de usuarios, corresponde a httpd://pinero/~login_usuario
Configuro la directiva:
<Directory /home/httpd/html>
Options Indexes Includes FollowSymLinks
AllowOverride None
order allow,deny
allow from all
</Directory>

Y con esto ya puede funcionar el servidor web, el resto de la configuración se pude mantener con las opciones (directivas) que por defecto a instalado apache, no obstante es importante revisar que el puerto predeterminado para la web es el 80 (Port 80).

El servidor web puede atender simultáneamente numerosas peticiones, dependiendo del número de peticiones de clientes que tenga el servidor será necesario modificar las directivas MinSpareServer y MaxSpareServers para que el rendimiento del equipo sea óptimo. Revisar también el número de servidores con los que se inicia el servidor: StartServers.

Terminada la configuración, con "httpd -t" se chequea la sintaxis de los archivos de configuración.

¿Cómo publicar páginas web?

Para poder publicar las páginas web en el servidor utilizando la opción de menú de Publicar de netscape u otros navegadores:
- Modificar httpd.conf
<Directory /home/httpd/html>
.....
Script PUT /cgi-bin/put.cgi
</Directory>
<Directory /home/httpd/cgi-bin>
AllowOverride None
Options ExecCGI
AuthType Basic
AuthName "Authorised PUT Publishers"
AuthUserFile /home/httpd/htpasswd-putusers
</Directory>

Copiar el archivo put.cgi (ver http://www.apacheweek.com/features/put) en /home/httpd/cgi-bin/.

Para que un usuario pueda publicar hay que darlo de alta en el archivo de password /home/httpd/htpasswd-putusers con # htpasswd htpasswd-putusers nombre_usuario.

La solución es instalar un servidor ftp y los usuarios con un cliente ftp administrarían su sitio web.

Proxy.

También puede hacer que el servidor apache haga de proxy con caché, para ello quite los comentarios de la directiva ProxyRequests On y si desea cache para acelerar los accesos quitar los comentarios de las directivas:

CacheRoot /var/cache/httpd
CacheSize 5
CacheGcInterval 4
CacheMaxExpire 24
CacheLastModifiedFactor 0.1
CacheDefaultExpire 1
#NoCache a_domain.com another_domain.edu joes.garage_sale.com -- No deseo cache para los dominios especificados.

Script de inicio

La directriz ServerType puede tener dos valores, "standalone" si httpd se va a iniciar mediante un script de inicio o "inetd" si se va a iniciar con inetd, así que modifique httpd.conf para iniciar de un modo u otro. Parece ser que si el servidor web va a tener muchas peticiones es mejor iniciarlo con standalone, pero para una red local es suficiente hacerlo con inetd y además puede ser más seguro.

En el primer caso incluir en /etc/rc.d/rcX.d el siguiente script de inicio, que en algún sistema tiene el nombre S85httpd, ejecutar con el parámetro "start", probarlo.

```
#!/bin/sh
#
# Startup script for the Apache Web Server
#
# chkconfig: 345 85 15
# description: Apache is a World Wide Web server. It is used to serve \
# HTML files and CGI.
# processname: httpd
# pidfile: /var/run/httpd.pid
# config: /etc/httpd/conf/access.conf
# config: /etc/httpd/conf/httpd.conf
# config: /etc/httpd/conf/srm.conf
# Source function library.
. /etc/rc.d/init.d/functions
# See how we were called.
```

```
case "$1" in
start)
echo -n "Starting httpd: "
daemon httpd
echo
touch /var/lock/subsys/httpd
;;
stop)
echo -n "Shutting down http: "
killproc httpd
echo
rm -f /var/lock/subsys/httpd
rm -f /var/run/httpd.pid
;;
status)
status httpd
;;
restart)
$0 stop
$0 start
;;
reload)
echo -n "Reloading httpd: "
killproc httpd -HUP
echo
;;
*)
echo "Usage: $0 {start|stop|restart|reload|status}"
exit 1
esac
exit 0
```

Para iniciarlo con inetd, y aplicarle las reglas de seguridad de tcp_wrapers, añada en el archivo /etc/inetd.conf:

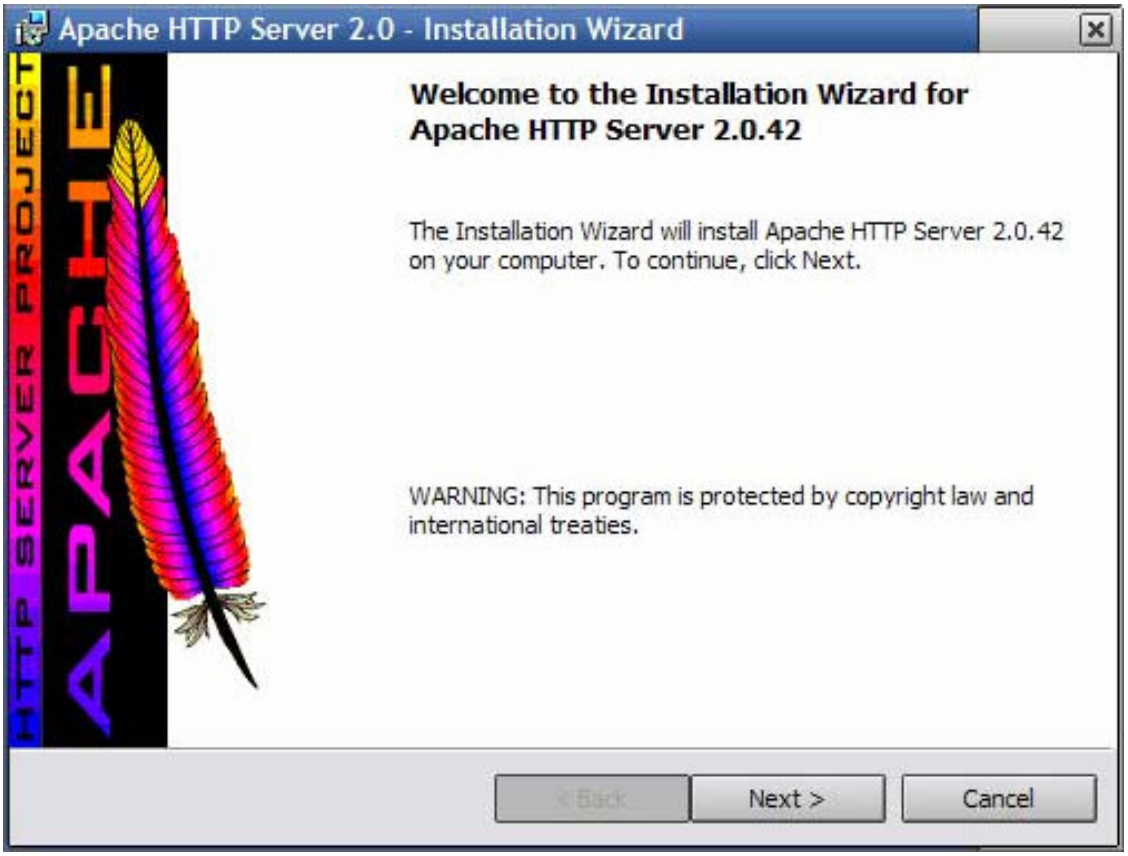
httpd stream tcp nowait root /usr/sbin/tcpd /usr/sbin/httpd -f /etc/httpd/conf/httpd.conf
compruebe que en /etc/services existe la entrada:
httpd 80/tcp httpd
y reiniciar inetd:
 # killall -HUP inetd

9.4 Instalación de Apache 2 en un Sistema Windows.

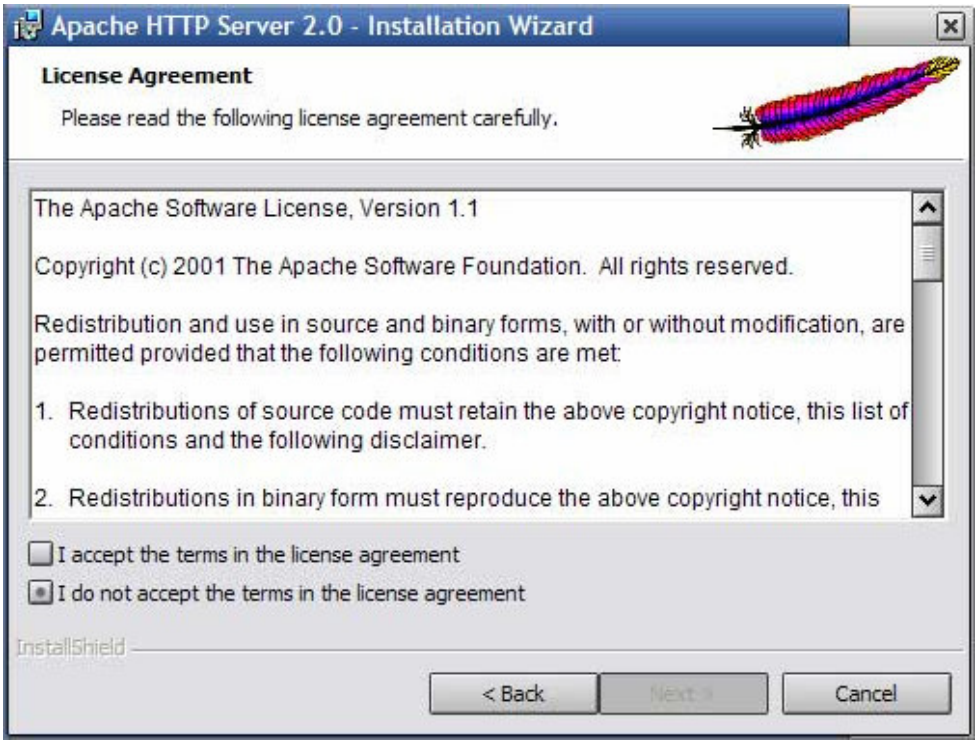
Puede descargar el programa de instalación apache_2.0.43-win32-x86-no_ssl.exe de la siguiente dirección: http://nagoya.apache.org/dist/httpd/binaries/win32/apache_2.0.43-win32-x86-no_ssl.exe

Bien, una vez descargado el programa de instalación de Apache, tiene que es instalarlo; estos son los pasos a seguir:

1º - Una vez bajado doble clic ejecutable y nos aparece la siguiente ventana:

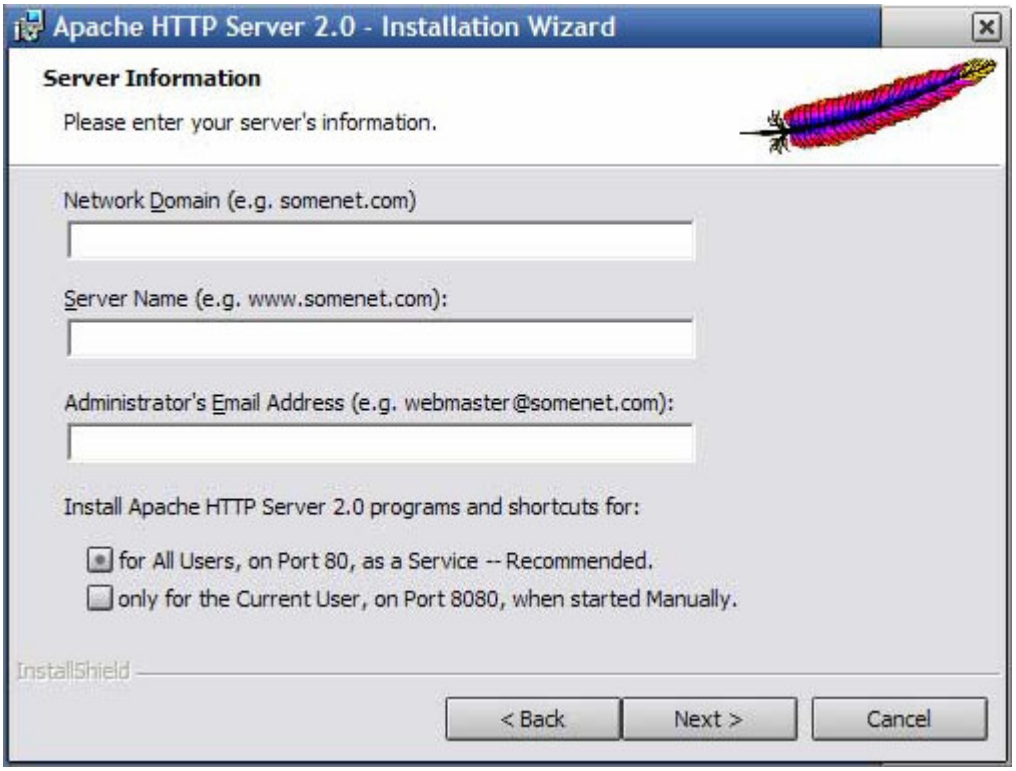


2º - En esta ventana haga clic en el botón next y aparecerá la ventana del contrato:



3º - Después de haber leído la licencia de Apache y haber seleccionado la casilla de aceptación (I accept the terms in the licence agreement), haga clic en el botón next, apareciendo la ventana de información general de Apache:

4º – Luego del leer que es Apache, donde descargar la última versión de Apache y todos los aspectos informativos hacer clic en el botón next, apareciendo la ventana de información del servidor a instalar:



Network Domain:
En esta casilla hay varias opciones; lo más comun, es que quiera instalar el servidor localmente, si este es tu caso en esta casilla pon "localhost" (sin las comillas). - Si por el contrario desea que el resto del mundo pueda ver publique o desarrolle bajo Apache entonces en esta casilla debe poner su dirección IP.

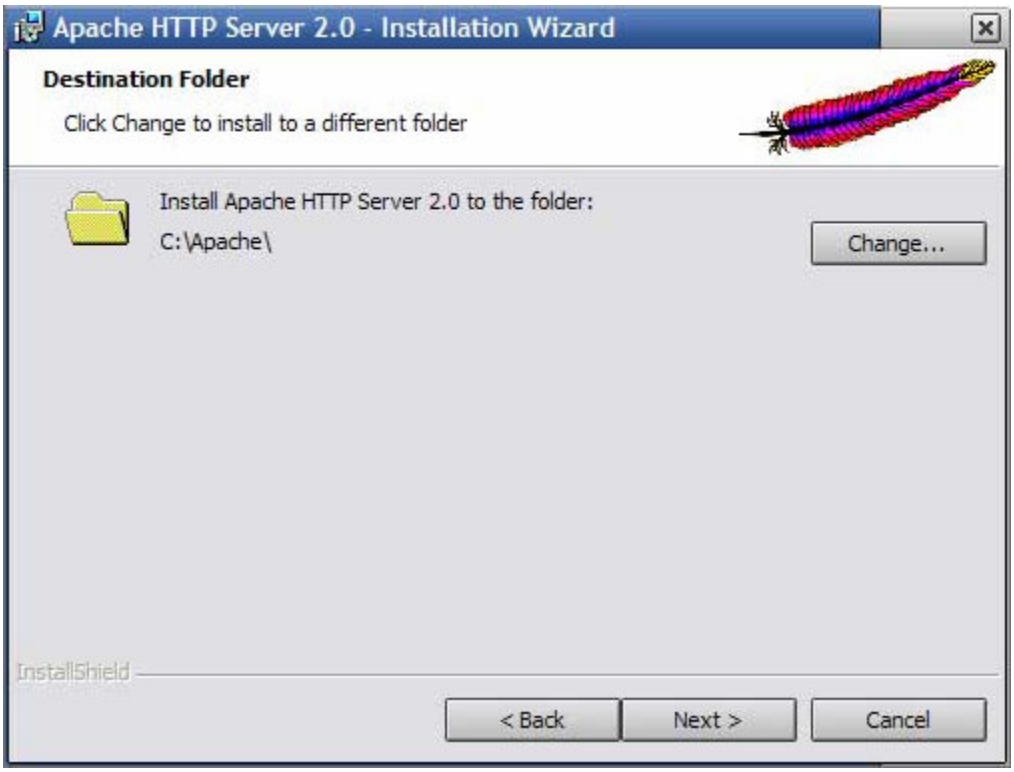
Nota: ¿No sabe cual es su dirección IP? puede saberlo ejecutando, en modo consola, el comando ipconfig.

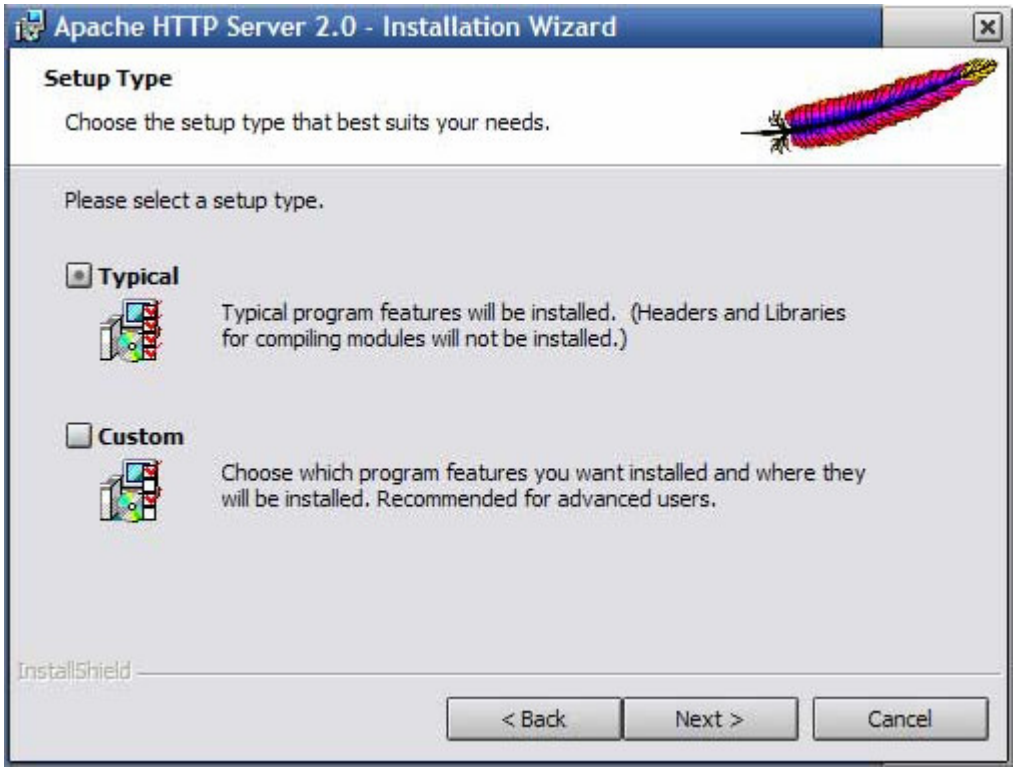
Server Name:
Es el nombre que quiera que tenga el servidor web, por ejemplo My_Server. Administrator´s Email address : tienes que poner la dirección de correo electrónico del administrador del servidor web; por ejemplo: edgardo.romero@udb.edu.sv. Después de haber rellenado estás casillas aparecen dos opciones:

- For all users in port 80, as a service: instala Apache como un servicio de Windows, es decir que Apache se ejecuta al iniciar el ordenador; eligiendo esta opción el servidor se configura para que todos los usuarios tengan acceso a el por medio del puerto 80.

- Only for the current user , on port 8080, when started manually: instala Apache como un programa normal, para ejecutar el servidor lo elige en el menú de inicio y se abre una ventana para indicar que se está ejecutando Apache. El servicio tendrá que ser iniciado manualmente.

Una vez rellenados todos los campos y elegida la opción que desea, haga clic en el botón next; aparecerá la ventana para elegir el tipo de instalación: Típica o Personalizada y haga clic en el botón next.





Aquí da a elegir el directorio donde desea instalar el Apache, por defecto se instala en Archivos de programa, pero lo puede instalar en la carpeta que usted quiera, en este caso se instalara en c:\Apache\ Para cambiar el directorio de instalación haga clic en el botón change y escriba c:\Apache; acepte dando clic en el botón OK y luego en el botón next.

Después de esto saldrá una ventana de confirmación, haga clic en el botón next y comenzará la instalación de Apache.

Cuando acabe la instalación aparecerá otra ventana, simplemente haga clic en el botón finish y ya habrá acabado la instalación del servidor.

Observe la configuración:
Lo primero es ejecutar el servidor apache eligiendo la opción start Apache in console, abriéndose una ventana (negra) indicando que se está ejecutando el servidor Apache.

Después de esto abra el navegador de Internet para ver si realmente el servidor está funcionando.

Una vez abierto, tendrá varias opciones según como haya rellenado el campo network domain:

Si ha puesto su dirección IP, en la barra de dirección de su navegador ponga esto: *http://SuDireccionIp:8080*

Nota: Se pone al final: 8080 para indicar al navegador que el servidor está configurado en el puerto 8080.

Si en el campo network domain ha puesto localhost, para que el servidor no sea visible desde Internet ponga esto en la barra de dirección del navegador: http://localhost:8080 o simplemente http://localhost

Si todo es correcto aparecerá una página similar a la siguiente diciendo que el servidor Apache está configurado con éxito.

Bien, lo siguiente es modificar la configuración de Apache a su gusto. Toda la información del servidor se guarda en el archivo de texto c:\Apache\Apache2\conf\httpd.conf , abra el archivo y busque el siguiente texto:

#
 # DocumentRoot: The directory out of which you will serve your
 # documents. By default, all requests are taken from this directory, but
 # symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "C:/Apache/Apache2/htdocs"

Esta es la carpeta donde va a tener los archivos .php , .html , etc lo mejor es cambiarla, por ejemplo ponga:

DocumentRoot "C:/servidor_web"

Nota: es importante fijarse que la barra es esta "/" no esta "\"

El siguiente texto a buscar es este:

#
 # This should be changed to whatever you set DocumentRoot to.
#
Directory "C:/Apache/Apache2/htdocs"

Aquí tiene que hacer lo mismo antes, sustituir el directorio por el que desee, en este caso:

Directory "C:/servidor_web”

10. APACHE PROJECT AXIS.

Una de las capacidades más notables del estándar SOAP es que permite el envío de mensajes entre plataformas diferentes. Por ejemplo, .NET Framework y Java. Básicamente, SOAP provee un protocolo estandarizado para transmitir datos entre máquinas.

Menos publicado, pero casi tan importante, es el hecho de que la "S" en SOAP es por "Simple" y es que los diseñadores de este protocolo cuidaron que la norma sea fácil de implementar, buscando que aparezcan muchas versiones en diversidad de ambientes.

En particular, en Java la opción más popular quizás es Axis, básicamente porque Axis es parte del proyecto Apache, que es una de las iniciativas de código abierto más exitosas y apreciadas.

Axis es una implementación Open-Source de un "SOAP Engine"; a través de este componente es posible llevar acabo una comunicación mediante web services una de las variaciones más prometedoras del lenguaje XML.

¿Dónde radica "SOAP Engine" como Axis ?

En el caso de "Axis", éste se encuentra diseñado para ser ejecutado dentro de "Java Application Server" o bien un "Servlet Engine" como Tomcat ; aunque hoy en día ya existen otros "SOAP Engines" que pueden ser ejecutados de manera independiente de un "Java Application Server" o "Servlet Engine", al residir un "SOAP Engine" dentro estos, permite que métodos residentes puedan ser publicados como web services y de esta manera ser accesibles de otras plataformas/lenguajes que también soporten "SOAP"

10.1 Servlet Engines" y "Java Application Servers"

Servlet Engines

Quizás el nombre que más salga a relucir con "Servlet Engines" es Tomcat o Jakarta Apache . Tomcat surgió de Sun Microsystem' s cuando desarrollaban un "Servidor de Páginas" que utilizara "Java",y posteriormente cedieron el código fuente a la fundación Apache.

A pesar del nombre Apache-Tomcat; Tomcat no requiere de Apache para su funcionamiento (solo requiere de un JDK ("Java Development Kit") y es aquí donde difiere un poco de las previas implementaciones:

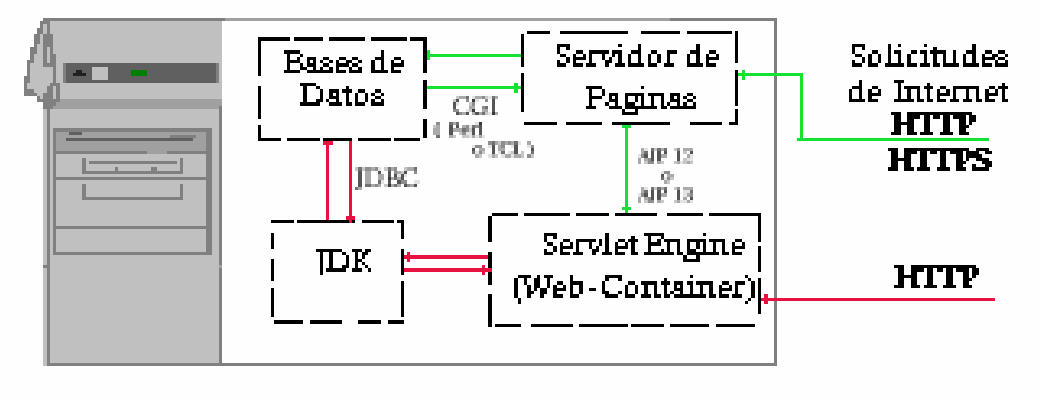


Figura 14. Servlet Engine.

En la figura anterior se demuestra que el Servlet engine (también llamado "Web Container") es capaz de responder a requisiciones de Internet, en efecto actuando como "Servidor de Páginas", sin embargo, aunque esto sea posible la gran mayoría de las implementaciones de Servlet Engines no funcionan tan eficiente como un "Servidor de Páginas", es por esto y otras razones que se opta por utilizar un "Servidor de Páginas" (Apache,Aol,Netscape..) en conjunción con un "Servlet Engine".

Tambien se demuestra la utilización de un "Servidor de Páginas" con un "Servlet Engine" (Web-Container), esta implementación suele utilizarse cuando se requiere utilizar encriptación o se tiene un sitio que contiene documentos puros en HTML , las cuales son dos areas que un "Servidor de Páginas" supera en desempeño comparado con un "Servlet Engine". La comunicación entre el "Servidor de Páginas" y el "Servlet Engine" se lleva acabo mediante el protocolo denominado ajp12 y recientemente con su sucesor ajp13.

¿Que hace el Servlet Engine ?

El "Servlet Engine" ofrece un "Ambiente" donde habitan los JSP y Servlets, es ahí donde se contemplan una gran cantidad de funcionalidades como: threading, manutención de sesiones, conectividad con el "Servidor de Páginas", es por esto al "Servlet Engine" también se le denomina "Web-Container".

Dos "Servlet Engines" (Web-Containers) que están en amplio uso y son utilizados con "Servidores de Páginas" son: Tomcat y ServletExec, donde el primero es open-source y el último es un producto cerrado; otro "Servlet Engine" es Resin (Open-Source) el cual permite: utilizar JavaScript como "Scripting Language" dentro de JSP' s y acceso a XSL una extensión de XML .

Como se observa en el diagrama también se requiere de un JDK ("Java Development Kit") , el cual llevará acabo la ejecución de los programas ("Servlets" y "JSP' s") en Java; como toda otra implementación existen diversas versiones de JDK' s, esto se debe a que cada JDK debe ser diseñado alrededor de un Sistema Operativo (para ser más exactos es el JVM "Java Virtual Machine" el que debe ser diseñado alrededor del Sistema Operativo), algunos JDK' s son: J2SE' s (Java 2 Standard Edition) de Sun y JDK' s de IBM.

Java Application Servers

"Java Application Servers" hoy en día denominados "Application Servers" ofrecen una manera de Integrar y ofrecer las funcionalidades requeridas por la gran mayoría de sistemas empresariales, una de las razones por las cuales el mercado ha sido inundado con estos "Application Servers" es que están diseñados alrededor de J2EE , que es solo un grupo de especificaciones definidas por Sun.

Estos "Application Servers" comunmente llamados Middleware se encuentran compuestos de la siguiente manera:

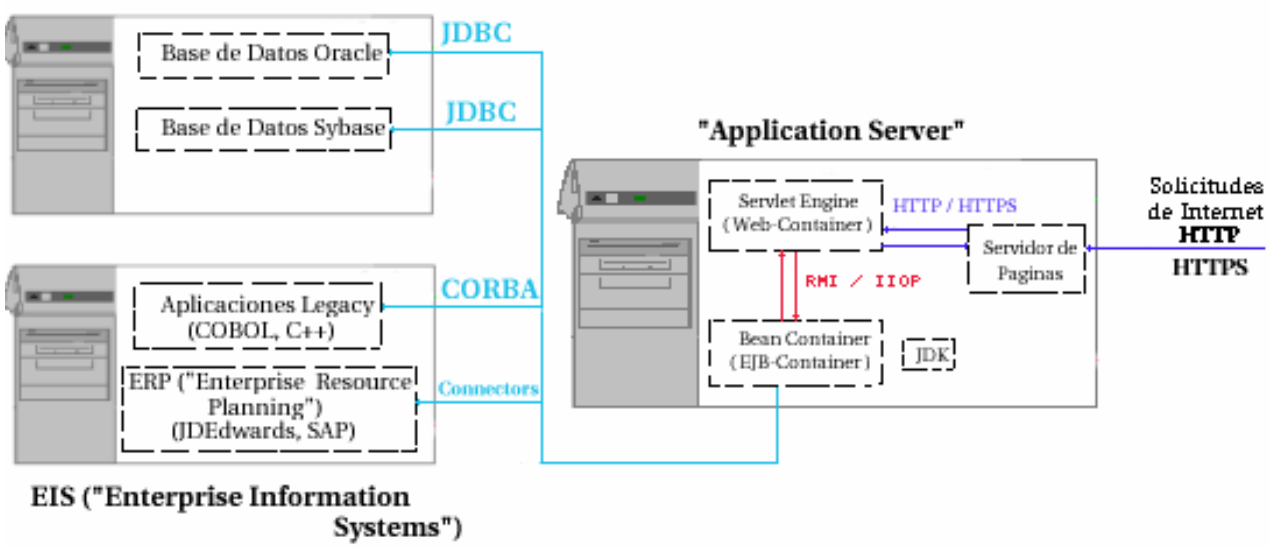


Figura 15. Application Servers.

Como su denominación lo implica ("Middleware") se encuentra en la parte media de una arquitectura de sistema, su flexibilidad reside en la posibilidad de accesar información en sistemas empresariales (EIS) como SAP, JdEdwards, bases de datos o alguna aplicación escrita en COBOL u otro lenguaje.

Dependiendo de la empresa que desarrolle el "Application Server" éste puede contener inclusive hasta un "Servidor de Páginas" o algún otro desarrollo propietario, sin embargo, los dos elementos primordiales (aunque no sean comercializados como tal) son el "Servlet Engine" (Web-Container) y "Enterprise Bean Engine" (Bean-Container).

¿Qué hace el Servlet Engine y Enterprise Bean Engine?

El Servlet Engine (Web-Container) en un "Application Server" realiza las mismas funcionalidades que fueron mencionadas anteriormente.(Ofrecer un ambiente para JSP y Servlets).

El "Enterprise Bean Engine" (Bean-Container) ofrece un "ambiente" donde residen EJB' s ("Enterprise Java Beans") , es mediante "Enterprise Java Beans" que se ejecuta la lógica de negocios sobre la información que reside en los sistemas empresariales ("EIS"). En el "Bean Container" (al igual que en el "Web Container") se contemplan varias funcionalidades: "Pooling" hacia bases de Datos (JDBC),control de transacciones(JTA-JTS),conectividad con ERP(Connectors),aplicaciones legacy(CORBA),entre otras cosas.

La mayor ventaja de este tipo de arquitectura se debe a la separación de funcionalidades y uso de protocolos de red como RMI/CORBA , esto facilita que puedan existir 4 o 5 "Hosts" en diferentes regiones geográficas, cada uno empleando cualquiera de los componentes antes mencionados. Por último, existen diversos "Application Servers" que son denominados "Fully J2EE Compliant" esto indica que cumplen con todas las especificaciones J2EE indicadas por Sun. (Vea J2EE en http://www.osmosislatina.com/java/componentes.htm#j2ee). Algunos "Application Servers" "Fully J2EE Compliant" son:

- WebLogic http://www.weblogic.com/.
- Websphere http://www.ibm.com/software/webservers/appserv.
- JRun http://www.allarie.com/.
- Oracle 9i Application Server http://technet.oracle.com/products/ias.
- iPlanet (Previamente Netscape Enterprise o Kiva) http://www.iplanet.com/

¿ Qué es JAX-RPC ?

JAX-RPC es el API estándar en Java para implementar e invocar operaciones de web services mediante el paradigma de los RPCs, similar al paradigma de CORBA, especifica un mapping de WSDL a Java, permitiendo que las implementaciones de JAX-RPC proporcionen un compilador de WSDL a Java que genere stubs (proxies) y skeletons para invocar e implementar web services, especifica un mapping de Java a WSDL que permite que las implementaciones de JAX-RPC proporcionen un compilador de Java a WSDL que genere el documento WSDL correspondiente a un interfaz Java, la definición del interfaz Java está sujeta a ciertas restricciones, el documento WSDL permite que un cliente (escrito sobre cualquier plataforma) pueda invocar el web service.

Axis es Una implementación de JAX-RPC para contenedores web J2EE, existen dos maneras de implementar un web service:

- Dentro de un contenedor web
- Dentro de un contenedor de EJBs

Incluye:

- Un conjunto de librerías
- Un compilador de Java a WSDL
- Un compilador de WSDL a Java

Definición de bindings

Un binding especifica un protocolo y formato de datos para un tipo de puerto (e.g. SOAP sobre HTTP)

Definición de servicios

Un servicio especifica un conjunto de "puertos" (endpoints), cada puerto está asociado a un binding particular y especifica su dirección de contacto, en JAX-RPC se usa el término "service endpoint" para referirse al puerto de un web service, se usa el término "interfaz del service endpoint" para referirse al interfaz del puerto, Usará indistintamente los términos "service endpoi nt" y "puerto".

Mapping de Java a WSDL

Tipos válidos JAX-RPC: tipos que se pueden emplear en la definición de interfaces remotos:

- Tipos primitivos y sus contrapartidas objetuales
- Clases estándar
- Tipos valor JAX-RPC
- Arrays ([]) de tipos válidos

TIPOS DE DATOS Y SU CONTRAPARTE:	
JAVA	**WSDL**
boolean	xsd: boolean
byte	xsd: byte
short	xsd: short
int	xsd: int
long	xsd: long
float	xsd: floa
double	xsd: double

CLASES ESTÁNDAR	
JAVA	**WSDL**
java.lang.string	xsd: string
java.math.BigInteger	xsd: integer
java.math.BigDecimal	xsd: decimal
java.until.Calendar	xsd: dateTime
java.until.Date	xsd: dateTime

ARRAYS ([ ]) DE TIPOS VÁLIDOS

Se mapean a tipos complejos derivados por restricción de un array SOAP, a excepción de byte[ ] (xsd:base64Binary).

TIPOS VALOR JAX-RPC

En general estas clases deben tener:

- Un constructor público sin argumentos
- Atributos públicos de tipos válidos o usar las convenciones de nombrado de JavaBeans para sus atributos (métodos getXXX y setXXX).

Pueden heredar de otras clases valor, se mapean a tipos WSDL complejos con compositor all o sequence, en caso de herencia, el tipo complejo se define por derivación, es buena práctica que implementen java.io.Serializable (interfaz marker).

Definición de interfaces remotos ("interfaces de service endpoints")

Extienden java.rmi.Remote (interfaz marker), todas las operaciones deben declarar java.rmi.RemoteException, cada interfaz se mapea a un puerto dado que en WSDL no existe herencia entre puertos, si un interfaz deriva de otro, el puerto hijo incluye todas las operaciones del padre, una operación no puede recibir como parámetro o devolver como valor de retorno una referencia a un interfaz remoto, actualmente SOAP no ofrece soporte para ello. Ejemplo: StockQuoteProvider.

EXCEPCIONES

java.rmi.RemoteException se mapea a un fault de SOAP, las excepciones específicas al puerto, es decir, las que extienden directa o indirectamente java.lang.Exception (pero no java.lang.RuntimeException) se mapean a un wsdl:fault. Definen un método getXXX para recuperar el valor de cada propiedad, disponiendo de un constructor que recibe las propiedades como parámetros. Ejemplo: IncorrectTickerSymbolException.

Si la excepción sólo tiene una propiedad, la parte del fault será de tipo simple; en otro caso, será de tipo complejo, la herencia de excepciones se mapea a herencia de tipos complejos

OTROS TIPOS

Si se desea usar otros tipos para los que JAX-RPC no tiene soporte directo (ej.: implementaciones de java.util.Collection), JAX-RPC permite implementar clases serializadoras y deserializadoras serializador es el que convierte el valor de un tipo Java a XML, deserializador es el que convierte el valor de un tipo XML a Java, algunas implementaciones de JAX-RPC proporcionan serializadores/deserializadores para clases estándar usuales. Ejemplo: Axis proporciona clases serializadoras/deserializadoras para algunas de las implementaciones de java.util.Collection.

INTEROPERABILIDAD

El uso de clases serializadoras/deserializadoras puede causar problemas de interoperabilidad, actualmente no hay un formato estándar para transmitir listas, mapas, etc. Para máxima interoperabilidad es mejor restringirse a los tipos directamente soportados, con el uso de arrays ([]) y tipos valor JAX-RPC se pueden representar estructuras complejas, que se mapean de forma estándar a tipos WSDL esta es la técnica usada en todos los ejemplos.

MAPPING DE WSDL A JAVA.

Las reglas del mapping de Java a WSDL a la inversa xsd:dateTime se mapea a java.util.Calendar (y no a java.util.Date).

Los structs XML (con compositor all o sequence) se mapean a una clase Java con métodos getXXX/setXXX para cada campo del struct. Además, necesita saber:

- ¿ Cómo se traducen las enumeraciones ?
- ¿ Cómo se traducen los parámetros out e inout ?

Si parte de las interfaces remotas Java, estas dos preguntas no son relevantes (porque Java no soporta directamente estos dos conceptos), pero sí lo son si parte de la definición WSDL También necesita conocer algunas clases generadas que son específicas al cliente o al servidor, se estudian como parte del modelo de implementación de clientes y servidores

ENUMERACIONES

```
// WSDL
<simpleType name="EyeColor">
        <restriction base="xsd:string">
                <enumeration value="green"/>
                <enumeration value="blue"/>
        </restriction>
</simpleType>

// Java
public class EyeColor {
        public static final String _green = "green";
        public static final String _blue = "blue";
        public static final EyeColor green = new EyeColor(_green);
        public static final EyeColor blue = new EyeColor(_blue);
        protected EyeColor(String value) { ... }
        public String getValue() { ... }
        public static EyeColor fromValue(String value) { ... }
        public boolean equals(Object obj) { ... }
        public int hashCode() { ... }
        // Otros métodos ...
}
```

PARÁMETROS OUT E INOUT

Uso de clases Holder similares a las de CORBA.

Existen clases Holder para los tipos WSDL predefinidos en el paquete javax.xml.rpc.holders. Para los tipos definidos por el programador, el compilador de WSDL genera clases Holder con el formato.

```
final public class <XXX>Holder
        implements javax.xml.rpc.holders.Holder {
        public <XXX> value;
        public <XXX>Holder() { ... }
        public <XXX>Holder(<XXX> value) { ... }
}
```

10.4 Modelo de implementación de servicios.

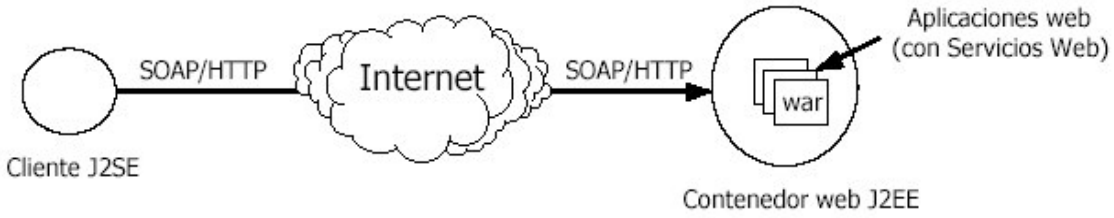


Figura 16. Modelo de implementación de servicios.

Contenedor web J2EE.

Es el Servidor en el que se pueden instalar un conjunto de aplicaciones web J2EE, Cada aplicación web (servlets, páginas JSP, páginas HTML, librerías, clases de la aplicación, etc.) se empaqueta en un archivo war, Una aplicación web puede incluir uno o varios web services, o puede contener sólo web services, se usara Jakarta Tomcat como contenedor web, modelo basado en contenedor de EJBs, El web service se implementa como un Stateless Sesion Bean, cuya interfaz remota es la del web service.

Modelo basado en contenedor web

REQUISITOS DE LA CLASE DE IMPLEMENTACIÓN

Implementa el interfaz remoto, ofrece un constructor público sin argumentos, en Axis, por defecto, el nombre de la clase de implementación es XXXSoapBindingImpl, siendo XXX el nombre del interfaz remoto. Ej.: StockQuoteProviderSoapBindingImpl.

Modelo de ejecución

La implementación de JAX-RPC tiene que incluir un servlet (o varios) que recibe las peticiones SOAP sobre HTTP que envían los clientes, invoca la operación correspondiente sobre el web service, devuelve una respuesta SOAP sobre HTTP con el resultado de la operación.

El servlet que recibe las peticiones puede crear una o varias instancias de la clase de implementación, Todas las instancias se consideran equivalentes ya que no pueden mantener estado específico para el cliente, una instancia puede recibir varias peticiones concurrentemente (en varios threads).

La clase de implementación tiene que ser thread-safe normalmente no será necesario hacer nada especial, dado que la implementación de las operaciones generalmente sólo hace uso de variables locales (pila) o de variables globales (static) de sólo lectura (típicamente caches), si modifica alguna estructura global (un atributo propio o alguna variable global), necesita sincronizar el acceso, sin embargo, en general, eso es mala idea, dado que una aplicación con estas características no funcionará en un entorno en cluster, para lograr escalabilidad y tolerancia a fallos, el servidor de aplicaciones web se puede replicar en varias máquinas, en estos casos, es mejor usar una base de datos para las estructuras globales que sean de lectura/escritura.

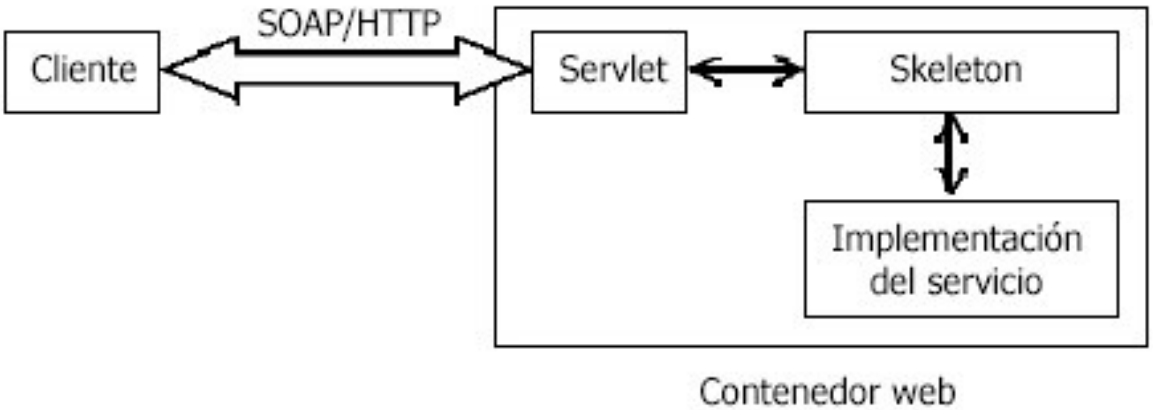


Figura 17. Modelo de Ejecución.

10.5 Ciclo de vida.

La clase de implementación puede implementar opcionalmente el interfaz (del paquete javax.xml.rpc.server).

```
public interface ServiceLifecycle
{
        void init(Object context)
        throws javax.xml.rpc.ServiceException;
        void destroy();
}
```

Permite realizar tareas de inicialización y destrucción, cada vez que el servlet crea una instancia de la clase de implementación, tiene que invocar a init en el caso de un contenedor de aplicaciones web, el contexto pasado es de tipo

javax.xml.rpc.server.ServletEndpointContext, y proporciona métodos para acceder a aspectos tales como la sesión, el ServletContext, etc.

Cada vez que el servlet decide destruir una instancia de la clase de implementación, tiene que invocar a destroy.

Empaquetamiento de aplicaciones web.

En J2EE, las aplicaciones web se empaquetan en archivos war, en particular, una aplicación web puede incluir web services, jar cvf aplicacionWeb.war directorio, opciones similares al comando Unix tar.

Ant incluye la tarea interna war, estructura de un archivo war:

- ♣ Directorio WEB-INF/classes
  - archivos .class que conforman la aplicación web, agrupados en directorios según su estructura en paquetes
  - ¡Sin archivos fuente!
- ♣ Directorio WEB-INF/lib
  - archivos jar de librerías que usa la aplicación.
  - ¡Sin archivos fuente!
- ♣ WEB-INF/web.xml
  - Configuración estándar de la aplicación web
- ♣ Directorio raíz y subdirectorios
  - Vista de la aplicación (Ej.: archivos HTML, páginas JSP, imágenes, etc.).
  - Visible a los navegadores. (Lo que hay debajo de WEB-INF sólo es visible a los servlets y páginas JSP de la aplicación).
  - Un archivo war se puede instalar (deployment) en cualquier servidor de aplicaciones web conforme a J2EE

WEB-INF/LIB

Incluye las librerías (archivos .jar) necesarias de Apache Axis.

WEB-INF/CLASSES

Contiene las clases del paquete es.udc.fbellas.corbaws.stockquote.wsdl, Definición del web service web e implementación, en el ejemplo, se han incluido algunas que no son necesarias para la implementación del servicio (Ej.: el stub).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
        PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
        "http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">
```

```
<web-app>
<display-name>Apache-Axis</display-name>

<servlet>
        <servlet-name>AxisServlet</servlet-name>
        <display-name>Apache-Axis Servlet</display-name>
        <servlet-class>
                org.apache.axis.transport.http.AxisServlet
        </servlet-class>
</servlet>
<servlet>
        <servlet-name>AdminServlet</servlet-name>
        <display-name>Axis Admin Servlet</display-name>
        <servlet-class>
                org.apache.axis.transport.http.AdminServlet
        </servlet-class>
                <load-on-startup>100</load-on-startup>
</servlet>
<servlet-mapping>
        <servlet-name>AxisServlet</servlet-name>

        <url-pattern>/servlet/AxisServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
        <servlet-name>AxisServlet</servlet-name>
        <url-pattern>/services/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
        <servlet-name>AdminServlet</servlet-name>
        <url-pattern>/servlet/AdminServlet</url-pattern>
</servlet-mapping>
</web-app>
```

Se declaran los servlets AxisServlet y AdminServlet formando parte de las librerías de Axis (WEB-INF/lib), AxisServlet el servidor de aplicaciones web le pasará todas las peticiones dirigidas a las URLs http://.../NombreAplicacionWeb/services/*

```
<servlet-mapping>
        <servlet-name>AxisServlet</servlet-name>
        <url-pattern>/services/*</url-pattern>
</servlet-mapping>
```

En este caso, asumiendo que tenga instalada la aplicación web con el nombre StockQuote, el cliente usará la URL http://.../StockQuote/services/StockQuoteProvider para acceder al puerto StockQuoteProvider (es la URL que aparece en el documento WSDL).

El servlet invocará la operación correspondiente sobre el service endpoint al que va dirigida la petición, y finalmente enviará una respuesta SOAP con el resultado de la operación.

ADMINSERVLET

Este permite realizar tareas de administración sobre los web services instalados (Ej.: activarlos, desactivarlos, etc.). Axis proporciona una aplicación standalone para comunicarse con este servlet

TCP MONITOR

Axis incluye una herramienta que permite monitorizar las peticiones y respuestas SOAP que envían clientes y servidores, Actúando como un "túnel" y recibiendo las peticiones del cliente, las muestra en pantalla y las redirige al service endpoint, al recibir las respuestas del service endpoint, las muestra en pantalla y se las envía al cliente StockQuoteClient.sh http://localhost:8000/StockQuote/services/StockQuoteProvider IBM SUN MIC.

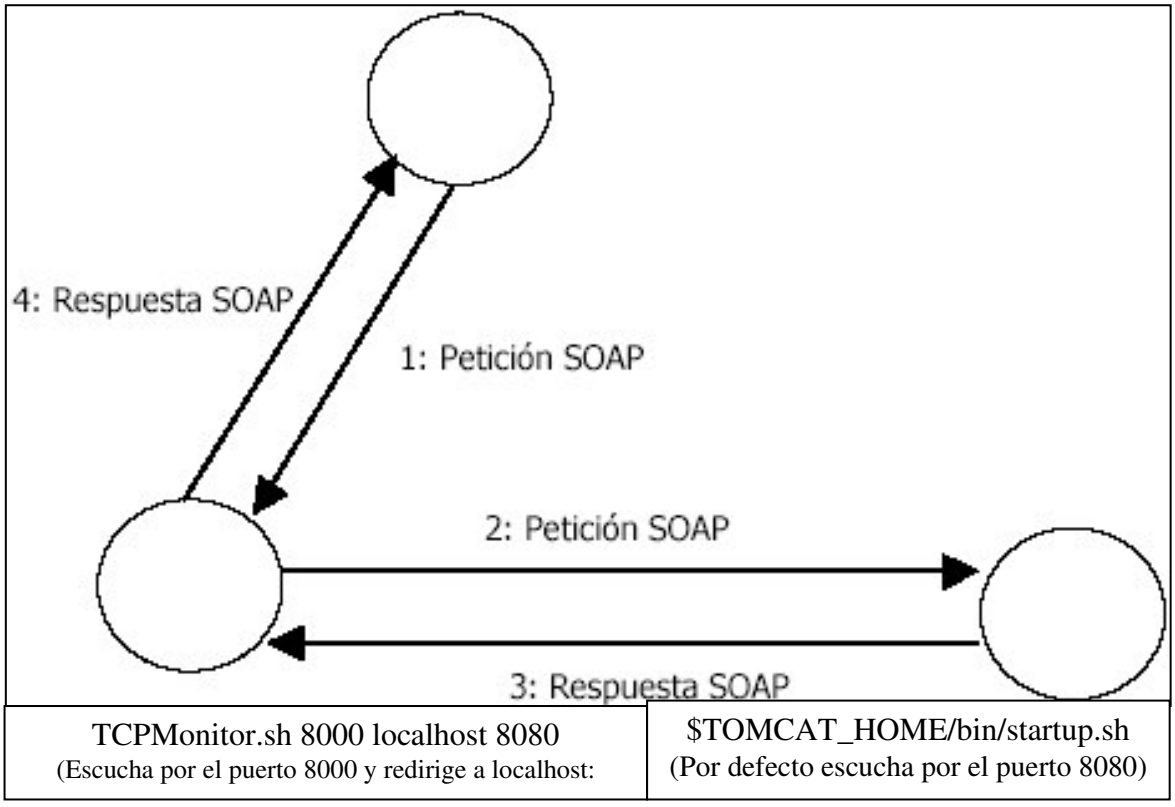


Figura 18. Proceso SOAP

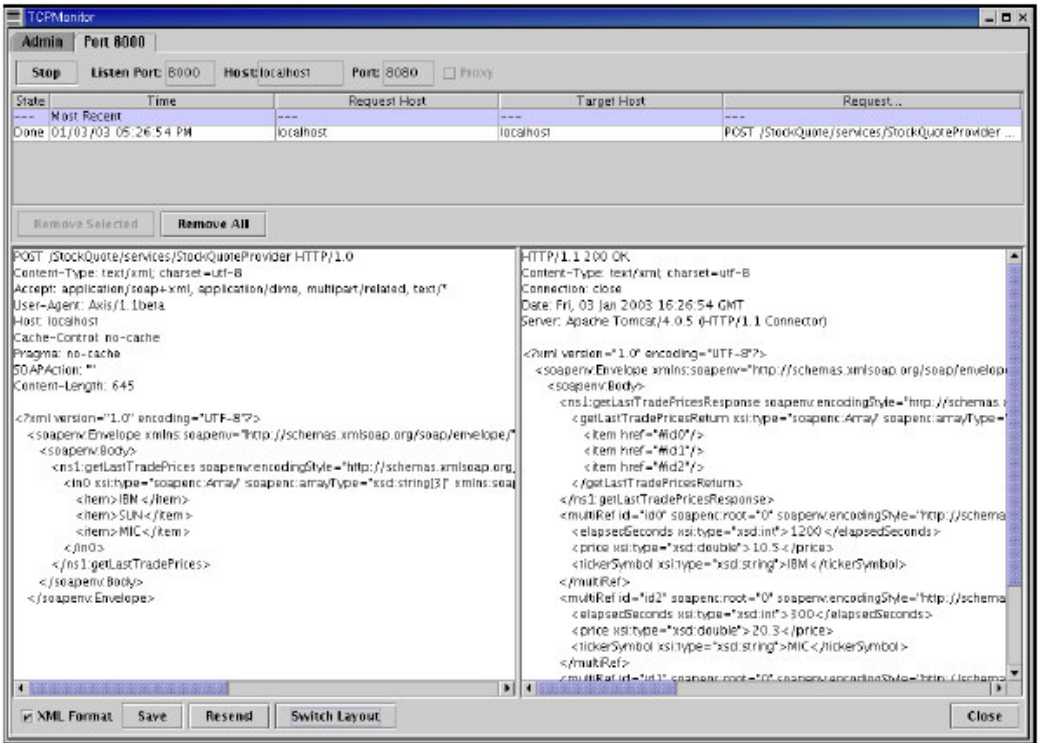


Figura 19. TCP Monitor

El stub del cliente envía una petición SOAP para invocar la operación getLastTradePrices sobre el service endpoint StockQuoteProvider, el parámetro de la operación (String[ ]) se ha serializado a XML, AxisServlet recibe la petición SOAP y determina que el cliente desea invocar la operación getLastTradePrices sobre StockQuoteProvider, deserializa el parámetro de la operación invocando la operación y serializa el valor de retorno (TradePrice[]) de la operación luego este envía una respuesta SOAP, el stub recibe la respuesta SOAP y deserializa el valor de retorno.

10.6 Instalación y Configuración

Instalación de Axis.

1. Descargar la versión binaria de Axis en: http://xml.apache.org/axis.
2. Descomprimir el archivo Tar de Axis un directorio temporal (/tmp/), esto genera un directorio llamado axis-<numero_de_version>.
3. Dentro de este directorio existe otro sub-directorio llamado lib, dentro del cual residen las distintas librerías (archivos JAR) necesarias para ejecutar Axis.

Configuración de Tomcat

Para utilizar Axis dentro de Tomcat es necesario diseñar un WAR ("Web-Archive") donde residirán las Clases/Métodos que serán ejecutados como "Web-Services". La estructura del WAR ("Web-Archive") para Axis es la siguiente:

```
+-usr/local/tomcat/webapps+--\
                        |
/-------------------------/
|
\ +-axis-+
        |
        |-*.jws (Clases/Métodos con Web-Services Nativos)
        |
        +-WEB-INF+
                |
                +-web.xml
                |
                +-classes--+
                |          |
                |          +- (Clases/Métodos para ser Web-Services)
                |
                +-lib--+
                       |
                       +-axis-ant.jar
                       +-axis.jar
                       +-commons-discovery.jar     Librerías
                       +-commons-logging.jar       de la Distribución Axis
                       +-jaxrpc.jar                /tmp/axis-numero_de_versión>/lib/
                       +-log4j-1.2.4.jar
                       +-saaj.jar
                       +-wsdl4j.jar
```

Nótese que la estructura del archivo WAR es colocada bajo un directorio llamado axis dentro de la instalación de Tomcat, esto permitirá el acceso a "Web-Services" bajo el URL: www.servidorprueba.com:8080/axis.

Archivo web.xml del WAR ("Web-Archive") Axis es el siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
   PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
   "http://java.sun.com/j2ee/dtds/web-app_2.3.dtd">

<web-app>

<!-- Definición de Servlets -->

 <servlet>
  <servlet-name>AxisServlet</servlet-name>
  <servlet-class>
     org.apache.axis.transport.http.AxisServlet
  </servlet-class>
 </servlet>

 <servlet>
  <servlet-name>AdminServlet</servlet-name>
  <servlet-class>
     org.apache.axis.transport.http.AdminServlet
  </servlet-class>
  <load-on-startup>100</load-on-startup>
 </servlet>
```

```
<!-- Mapeo de Servlets -->

  <servlet-mapping>
    <servlet-name>AxisServlet</servlet-name>
    <url-pattern>*.jws</url-pattern>
  </servlet-mapping>

  <servlet-mapping>
    <servlet-name>AxisServlet</servlet-name>
    <url-pattern>/services/*</url-pattern>
  </servlet-mapping>

  <servlet-mapping>
    <servlet-name>AxisServlet</servlet-name>
    <url-pattern>/servlet/AxisServlet</url-pattern>
  </servlet-mapping>

  <servlet-mapping>
    <servlet-name>AdminServlet</servlet-name>
    <url-pattern>/servlet/AdministracionAxis</url-pattern>
  </servlet-mapping>

</web-app>
```

- Las primeras declaraciones hasta antes del elemento <web-app> son comunes a cualquier otro archivo web.xml.

- Seguido del elemento <web-app> se definen dos Servlets para Axis, el primero de estos llamado AxisServlet representa la Clase org.apache.axis.transport.http.AxisServlet la cual representa el elemento principal en requisiciones; el otro Servlet representado por la Clase org.apache.axis.transport.http.AdminServlet representa un servicio administrativo para Axis, el cual es definido a través del nombre AdminServlet.

- En la siguiente sección son mapeados distintos URL' s para ser redireccionados a los distintos Servlets definidos anteriormente:

  - Los URL' s terminados en *.jws así como todos aquellos URL' s definidos bajo el directorio services y el URL /servlet/AxisServlet serán atendidos por el Servlet llamado AxisServlet.

  - El URL /servlet/AdministracionAxis será atendido por el Servlet Administrativo AdminServlet.

Una vez definido este archivo web.xml es posible iniciar con el proceso de diseño de "Web-Services" a través de Axis; en Axis existen dos posibilidades para definir "Web-Services":

- A través de Clases .jws, lo cual implica un diseño desde "0" (cero).
- Y la más importante, permitiendo acceso a Clases ya existentes para que sean accesibles como un "Web-Service".

Servicio web

Un primer programa no puede estar completo sin un "Hola Mundo" en algún lugar dentro de él, este es un programa ejemplo en código fuente Java:

```
public class HelloWorldService
{
    public String HelloWorld(String data)
    {
        return "Hola Mundo! enviaste la cadena '  " + data + "'  .";
    }
}
```

Después hay que registrar este código fuente en Axis. Hay dos maneras básicas de registrar web services , una forma fácil y rápida que tiene algunas restricciones, y una forma larga y minuciosa que es más flexible:

- **Rápida** - coloque el archivo fuente Java en el directorio raíz de Axis y cambie la extensión del archivo a *.jws. Eso es - bastante fácil, lo malo de éste método es que no puede especificar paquetes, y no puede usar archivos de clases en binario - necesita el código fuente. En la documentación sugieren usar la forma minuciosa para servicios en producción, pero indican que más características para el registro rápido estaban en las tareas.
- **Minuciosa -** un poco difícil pero no por mucho tiempo. Tiene que compilar su clase, crear un archivo de configuración llamado "Descriptor de registro de web services"[Web Service Deployment Descriptor] (WSDD), y usar una herramienta para someter ese archivo WSDD al servidor Axis.

Luego de haber registrado en Axis el código fuente Java, hay que probar que el web service fue instalado navegando a la dirección del servicio:
http://debianbox:8080/axis/HelloWorldService.jws.

Si recibe una página HTML diciendo que hay un web service instalado en ese lugar ha tenido éxito! vaya un paso más allá tratando de llamar el método HelloWorld: http://debianbox:8080/axis/HelloWorldService.jws?method=HelloWorld&data=Hola+mi+nombre+es+Edgardo. Y reciba el XML de la llamada al método. La llamada se completo exitosamente!

10.7 Instalación de TomCat y Ejecución.

Pasos Previos

Instalación del JDK (J2SE)

La base para operar cualquier producto que utiliza "Java" es el "JDK" de la plataforma correspondiente, puede encontrar instrucciones para plataformas Linux así como Windows, en las siguientes direcciones:

- JDK para Linux : http://javabasico.osmosislatina.com/java_linux.jsp
- JDK para Windows : http://javabasico.osmosislatina.com/java_windows.jsp.

Los pasos anteriores son los suficientes para el "JDK" en lo que concierne a la instalación de Tomcat 4.x, posiblemente después tenga que trabajar con la variable ambiental CLASSPATH.

INSTALACIÓN DE TOMCAT

1. Descargar la versión binaria de Tomcat en: http://jakarta.apache.org/tomcat. (La versión de Código Fuente (src) solo es necesaria si quiere experimentar y/o Instalar Apache con Tomcat).

2. Descomprimir el archivo Tar de Tomcat en /usr/local/, esto genera un directorio llamado jakarta-tomcat-<numero_de_version>, para dar mayor uniformidad se recomienda cambiar el nombre de este directorio a tomcat.

3. Posteriormente se debe definir una variable ambiental la cual le indicará al sistema la ubicación de Tomcat, esta variable se llama CATALINA_HOME la cual debe ser agregada a /etc/bashrc, si no esta familiarizado con ambientes *nix, esto significa agregar la linea: export CATALINA_HOME=/usr/local/tomcat; para instalaciones Windows esta variable ambiental puede ser definida de la misma manera que CLASSPATH, descrita en las instrucciones del JDK.

CONFIGURACIÓN LOCAL

Solo para ilustrar esta instalación inicial modifique el archivo /etc/hosts agregando una línea como la siguiente: **127.0.0.1 www.servidorprueba.com**

EJECUCIÓN Y PRUEBAS

Una vez efectuados los pasos anteriores es posible realizar pruebas iniciales sobre Tomcat, para esto ejecute lo siguiente:

1. Vaya al directorio bin de Tomcat (/usr/local/tomcat/bin) y ejecute el archivo startup.sh:

   [root@servidor1 bin]# ./catalina.sh run
   Using CATALINA_BASE: /usr/local/tomcat
   Using CATALINA_HOME: /usr/local/tomcat
   Using CATALINA_TMPDIR: /usr/local/tomcat/temp
   Using JAVA_HOME: /usr/local/jdk

   Debe observar algo similar al desplegado anterior; la pantalla permanecerá inhabilitada, este congelamiento en pantalla se debe a que aquí será enviado todo error detectado por "Tomcat"

2. Aunque pueda descongelarse la pantalla mediante Ctrl-C y seguir ejecutando comandos en pantalla, Tomcat no es cerrado/clausurado hasta que se ejecute shutdown.sh, inclusive si intenta ejecutar ./startup.sh consecutivamente, sin ejecutar shutdown.sh se genera un error.

3. Asegurándose que Tomcat este activo, para cualquier navegador ("Netscape","Galeon","Opera" u otro) del sistema intente visitar la dirección que fue definida en /etc/hosts, en este caso www.servidorprueba.com agregando el fragmento: 8080 al final: www.servidorprueba.com:8080.

   Al introducir la dirección anterior en un navegador, se le esta indicando que solicite la página principal de www.servidorprueba.com en el puerto TCP 8080; www.servidorprueba.com esta en la maquina local (127.0.0.1) como fue definido en /etc/hosts o por DNS y el puerto 8080 es en el que precisamente esta Tomcat (en la secuencia de startup.sh se indica).

4. En su navegador debe aparecer la página principal de documentación de Tomcat y los diversos Links hacia los ejemplos de Tomcat.

10.8 Configuración

server.xml

server.xml es el archivo principal de configuración para Tomcat, al igual que otros archivos de configuración para productos empleados en servidor puede contener una gran variedad de parámetros.

VALIDACIÓN, <!-- --> Y VALORES "DEFAULT"

El archivo server.xml es un archivo en XML, el cual de no contener una estructura conforme a XML, se indicará al arranque de Tomcat; dicho archivo se encuentra en el directorio /usr/local/tomcat/conf donde /usr/local/tomcat es el directorio definido en CATALINA_HOME.

Como cualquier otro documento en XML todo contenido entre <!-- --> es considerado un comentario, y por lo tanto cualquier parámetro que se encuentre entre estos caracteres no es utilizado por "Tomcat"; aquellos parámetros que no sean definidos dentro de server.xml son asignados un valor "Default" por Tomcat.

<Server> - </Server>

<Server> es el elemento principal del archivo server.xml y todas las demás secciones deben encontrarse entre estos nodos; el atributo port indica el puerto TCP donde se espera la señal de cierre (shutdown) de Tomcat, el cual rara vez es modificado.

<Listener>

A través de los elementos <Listener> se configuran las extensiones JMX ("Java Management Extensions") que serán utilizadas por Tomcat, dichos elementos toman dos atributos : className que indica la Clase diseñada para escuchar sobre eventos JMX y debug para especificar el nivel de "debug" generado al tiempo de ejecución.

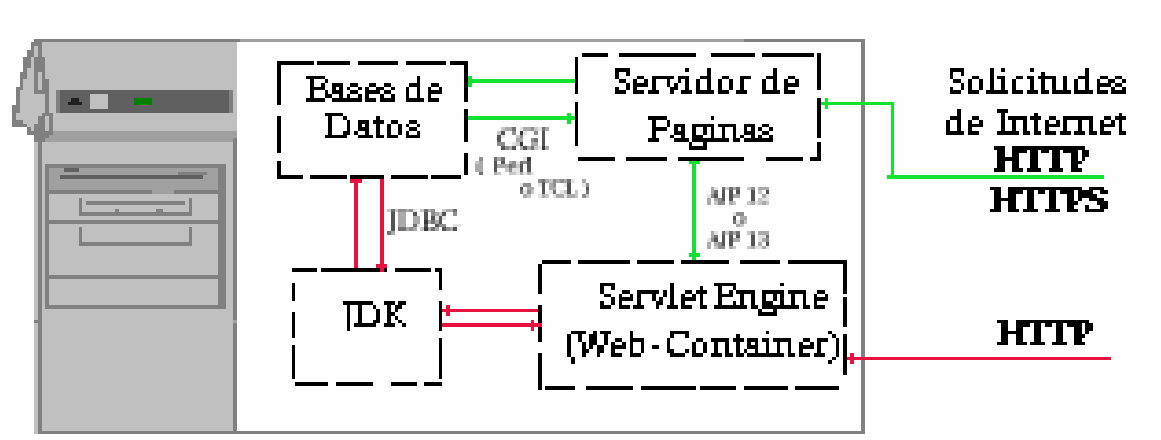
<GlobalNamingResources> , <Resource> y <ResourceParams>

Anidado dentro de los elementos <GlobalNamingResources> es posible definir recursos JNDI para ser utilizados globalmente en Tomcat. Lo anterior evita que estos recursos tengan que ser declarados a nivel de WAR de manera individual.

A través de <Resource> es como define el tipo de recurso JNDI que será utilizado y mediante <ResourceParams> se especifican los parámetros específicos que tomará el recurso en dicha instancia de Tomcat.

<Service> - </Service>

Esta parámetro permite configurar Tomcat para diferentes modalidades de ejecución, en el archivo server.xml "Default" se definen dos modalidades a través del atributo name; la definición asignada name="Catalina" es empleada para ejecutar Tomcat individualmente.



<Connector>

El elemento Connector representa las conexiones (Puertos TCP) que serán abiertas por Tomcat al arranque, a su vez dentro de cada elemento Connector se definen diversos atributos los cuales dan más detalles acerca de la conexión. El elemento Connector más importante es aquel que define la clase: HttpConnector.

```
<Connector port="8080"
        maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
        enableLookups="false" redirectPort="8443" acceptCount="100"
        debug="0" connectionTimeout="20000"
        disableUploadTimeout="true" />
```

La declaración anterior indica que Tomcat esta dispuesto a dar respuesta a requisiciones que provengan en el puerto 8080 del "Host" donde esta instalado Tomcat.

Para ambientes de producción en los cuales toda requisición será atendida por Tomcat el parámetro port de este elemento debe ser modificado al valor de 80, este puerto es el ampliamente conocido puerto TCP 80 con el que intenta comunicarse cualquier Navegador("Netscape","Explorer","Opera" u otro) en Internet.

Otras declaraciones para Connectors son aquellas para utilizar encriptación (HTTPS) y los conectores AJP; la configuración de HTTPS, sin embargo, los conectores AJP son aquellos utilizados para la comunicación entre Apache y Tomcat y son los siguientes:

```
<Connector port="8009"
        enableLookups="false" redirectPort="8443" debug="0"
        protocol="AJP/1.3" />
```

Esta declaración indica que el puerto 8009 estará recibiendo conexiones bajo ajp13, lo anterior es solo de interés si utiliza Apache en conjunción con Tomcat.

<Engine> - </Engine>

Los elementos <Engine> los cuales deben encontrarse anidados dentro de <Service> representan el mecanismo que atenderá toda solicitud arribada Tomcat, esto es, toda solicitud recibida por las definiciones Connectors será procesada por <Engine>, los atributos de este elemento son los siguientes:

```
<Engine name="Catalina" defaultHost="localhost" debug="0">
```

defaulthost representa el nombre del servidor Tomcat mientras debug indica el nivel de "debug" generado al tiempo de ejecución.

LOGGER

Los elementos Logger le indican a Tomcat hacia donde deben ser enviados los registros "Logs":

```
<Logger className="org.apache.catalina.logger.FileLogger"
        directory="logs"  prefix="localhost_log." suffix=".txt"
        timestamp="true"/>
```

Lo anterior indica que los registros de Tomcat deben ser enviados al archivo localhost_log.txt; la ubicación de este registro ("log") puede ser modificada al nivel de los elementos Host los cuales permiten definir Virtual Hosts. Analice estos registros con Analog.

<Host> - </Host>

Los elementos Host permiten definir varios Hosts "Virtuales" para Tomcat, esto es, a través del elemento <Engine> se define un sitio (localhost) para atender solicitudes, a través de Host es posible definir diversos sitios "Virtuales", su configuración es la siguiente:

<Host name="desarrollo.servidorprueba.com" debug="0" appBase="webapps" unpackWARs="true">

Lo anterior indica que el sitio desarrollo.servidorprueba.com contiene sus aplicaciones (Archivos WARS) bajo el directorio $CATALINA_HOME/webapps, donde $CATALINA_HOME es /usr/local/tomcat; el atributo unpackWARs le indica a Tomcat que debe descomprimir los archivos WAR' s al arranque.

Como ya fue mencionado, dentro de estos elementos <Host> es posible utilizar <Logger> para generar registros ("logs") por cada sitio virtual.

CONTEXT

Context es un elemento utilizado para indicar la ubicación de las aplicaciones ejecutadas en Tomcat , en su configuración "Default" estas aplicaciones se encuentran dentro del directorio webapps bajo el directorio raíz de Tomcat (/usr/local/tomcat).

Una aplicación en Tomcat o cualquier Servlet Engine (Web-Container) es un conjunto de "JSP' s (Java Server Pages)" y/o "Servlets" agrupados con ciertos parámetros de arranque y seguridad, este conjunto de archivos / aplicación en todo Servlet Engine es conocido como un WAR (Web-Archive).

Toda aplicación en Tomcat se encuentra agrupada en WARS ("Web-Archives"), la estructura de un WAR es definida por Sun (creador de Java) la cual debe ser implementada en cualquier producto de "Servlet Engine"(Web-Container).

WARS EN TOMCAT

Cuando recién se instala "Tomcat" éste ya contiene varios WARS, los cuales contienen ejemplos y documentación, estos WARS se encuentran en el directorio /usr/local/tomcat/webapps, donde /usr/local/tomcat es el directorio raíz de Tomcat (TOMCAT_HOME) definido al instalar Tomcat

Si se desciende a este directorio se observará que además de los archivos *.war existe además un directorio con el mismo nombre del WAR, esto es, si existen WAR' s por nombre ROOT.war y examples.war también existen directorios llamados ROOT y examples.

Estos directorios son los que precisamente contienen la estructura del archivo WAR, esto es, el archivo WAR en sí no es legible sino que tiene que ser expandido/descomprimido para ser leído.

ESTRUCTURA DE UN ARCHIVO WAR

La estructura de un Archivo WAR es la siguiente:

- **/ *.html *.jsp *.css:** Este directorio base contiene los elementos que comúnmente son utilizados en un sitio, Documentos en HTML, JSP's, CSS ("Cascading Style Sheets") y otros elementos.

- **/WEB-INF/web.xml:** Contiene elementos de seguridad de la aplicación así como detalles sobre los Servlets que serán utilizados dentro de la misma.

- **/WEB-INF/classes/ :** Contiene las clases Java adicionales a las del JDK que serán empleadas en los JSP' s y Servlets

- **/WEB-INF/lib/:** Contiene los JAR' s que serán utilizados por su aplicación.

Este tipo de estructura permite portabilidad a las diversas aplicaciones que son desarrolladas en Java, esto es, si se genera una aplicación para Foros o transacciones financieras estas se pueden contener en un archivo WAR, concentrando cualquier clase Java dentro de /WEB-INF/classes, parámetros específicos en /WEB-INF/web.xml, archivos JAR' s en /WEB-INF/lib/ y los diversos JSP' s, CSS y documentos HTML bajo el directorio raíz del archivo WAR.
Ejecución de WARS en Tomcat

Los siguientes pasos describen como interactúa "Tomcat" con WARS, tome en cuenta que este proceso muy probablemente difiera de otros "Servlet Engines (Web-Containers)" como ServletExec y los diversos "Java Application Servers" en el mercado.

Cuando se lleva acabo la ejecución de Tomcat éste inspecciona y automáticamente expande cualquier archivo WAR que se encuentra bajo el directorio webapps, si observa a detalle los registros ("Logs") de Tomcat notará algunas líneas como las siguientes:

2004-07-13 02:55:13 - ContextManager: Adding context Ctx(/examples)
2004-07-13 02:55:13 - ContextManager: Adding context Ctx(/admin)
2004-07-13 02:55:13 - ContextManager: Adding context Ctx( ) <--- WAR ROOT
2004-07-13 02:55:13 - ContextManager: Adding context Ctx(/test)

Estas líneas indican que han sido expandidos 4 WARS con nombres examples, admin, test y uno llamado ROOT, esta expansión genera un directorio llamado de la misma manera que el archivo WAR.

Una vez creados estos directorios (expandidos de WARS) Tomcat toma la información solicitada al sitio de estos directorios, lo anterior significa que si su sitio se llama www.apservidor.com y un usuario intenta visitar su página principal le aparecerá el archivo que se encuentra en el directorio ROOT, si se intenta visitar www.apservidor.com/admin aparecerá el archivo inicial del directorio admin y así sucesivamente.

Lo anterior generalmente ofrece el comportamiento deseado, siendo que se expande el WAR de acuerdo a su nombre, si se incluye un WAR llamado foros este será accesible del directorio llamado foros del sitio (www.apservidor.com/foros), sin embargo, existen ocasiones en las que es conveniente permitir acceso a cierto WAR bajo otro directorio diferente a su nombre.

Si tiene un WAR llamado registro.war es posible que desee sea utilizado/accesible en varios lugares de un sitio y no necesariamente bajo www.apservidor.com/registro, esta variación se logra modificando el parámetro Context de server.xml.

Creación de WARS en Tomcat

Finalmente cabe mencionar las maneras en que pueden ser creados los archivos WARS, existen varias, a continuación se mencionan las más comunes:

- Vía "IDE": A través de un IDE ("Integrated Development Environment") es posible generar estos archivos WAR' s, con este tipo de herramienta la creación depende fuertemente del tipo de IDE utilizado.

- Mediante Ant: "Ant" es una herramienta Open-Source, la cual facilita la construcción de aplicaciones en Java, "Ant" no es considerado un IDE (inclusive en ocasiones se utiliza en conjunción con un IDE), sin embargo, facilita la creación de aplicaciones Java; si conoce un poco de Unix / Linux "Ant" es considerado el make de Java.

Finalmente recuerde que la creación de WAR' S no es obligatoria sino solo una manera de transferir y modularizar aplicaciones, si coloca la estructura de un WAR bajo el directorio webapps en Tomcat esto funcionará de la misma manera que si hubiera sido descomprimido/expandido el archivo WAR.

Web-Services" Nativos (*.wsd)

Esta metodología para diseñar "Web-Services" en Axis es una de las maneras más sencillas para ejemplificar el uso de SOAP.

CÓDIGO FUENTE CALCULADORA.JWS

```
public class Calculadora {
  public int sumar(int x, int y)
  {
    return x + y;
  }


  public int restar(int x, int y)
  {
    return x - y;
  }
}
```

La Clase Java anterior define dos métodos los cuales suman y restan respectivamente dos valores proporcionados como datos de entrada, nótese que esta Clase es definida con la terminación .jws (Java Web Service) y no con la clásica terminación .java.

Para ejecutar esta Clase dentro de Axis debe ser colocada bajo el WAR Axis ("Web-Archive") definido anteriormente, la clase debe ser colocada como Código Fuente y no debe ser compilada; recuerde que debido al contenido del archivo web.xml en este WAR ("Web-Archive"), todo archivo terminado en .jws será procesado por Axis.

11. SOAP

11.1 Visión General

SOAP es fundamentalmente un paradigma de intercambio de mensajes en un sólo sentido, sin estado, pero las aplicaciones pueden crear patrones de interacción más complejos (por ejemplo, petición/respuesta, petición/respuestas múltiples, etc.) combinando tales intercambios de un solo sentido con características proporcionadas por el protocolo utilizado y/o información específica de la aplicación en cuestión. SOAP no interfiere en la semántica de cualesquiera datos específicos de aplicación que comunica, ni tampoco en asuntos tales como en enrutamiento de mensajes SOAP, transferencia de datos fiables, cortafuegos que atraviesa, etc. No obstante, SOAP proporciona el marco de trabajo por el que la información de aplicaciones específicas puede comunicarse de forma extensible. También, SOAP proporciona una descripción completa de las acciones que debe realizar un nodo SOAP al recibir un mensaje SOAP.

11.2 Intercambio de Mensajes SOAP

SOAP es una infraestructura de mensajería sencilla para la transferencia de información especificada en la forma de un conjunto de Información XML entre un remitente SOAP inicial y un destinatario SOAP final. Los escenarios más interesantes implican habitualmente el intercambio de múltiples mensajes entre estos dos nodos. El intercambio más simple de esta forma es el de un patrón petición-respuesta. Algunos de los primeros usos de [SOAP 1.1] enfatizaban el uso de este patrón como el medio para transportar llamadas a procedimientos remotos (RPC), pero es importante hacer notar que no todos los intercambios petición-respuesta SOAP pueden o necesitan ser modelados como RPC. Este último es utilizado cuando existe la necesidad de modelar cierto comportamiento programático, en el que los mensajes intercambiados conformen con una descripción predefinida de la llamada remota y de retorno.

Un conjunto de escenarios de uso mucho mayor que el cubierto por el patrón petición-respuesta puede ser modelado simplemente como contenido basado en XML intercambiado en mensajes SOAP para formar una "conversación" de ida y vuelta, en la que la semántica esta al nivel de las aplicaciones que envían y reciben los mensajes

11.3 Escenarios de Error

SOAP proporciona un modelo para la gestión de situaciones en las que surgen errores durante el proceso de un mensaje SOAP. SOAP distingue entre las condiciones que resultan en un error, y la habilidad de señalar el error al remitente del mensaje erróneo o a otro nodo. La habilidad de señalar el error depende del mecanismo de transferencia utilizado, y un aspecto de la especificación de enlace de SOAP a un protocolo es especificar como se señalan loe errores, si es que se señalan. El resto de esta sección asume que existe un

mecanismo de transferencia para señalar los errores que se producen al procesar los mensajes recibidos, y se concentra en la estructura del mensaje SOAP de error.

El elemento SOAP env: Body tiene otro papel diferente en el que es el lugar en el cual se aloja la información de error propiamente dicha.

El modelo de error SOAP requiere que todos los errores específicos de SOAP y específicos de la aplicación sean comunicados utilizando un único elemento diferenciador, env: Fault [env: Error], transportado dentro del elemento env: Body. El elemento env: Fault contiene dos subelementos obligatorios, env: Code [env: Codigo] y env: Reason [env: Razon], y (opcionalmente) información específica de la aplicación en el sub-elemento env: Detail [env: Detalle]. Otro sub-elemento opcional, el env: Node [env: Nodo], identifica a través de un URI el nodo SOAP que generó el error, su ausencia implica que fue el destinatario final del mensaje el que lo hizo. Aún existe otro subelemento opcional más, env:Role [env:Papel], que identifica el papel que juega el nodo que generó el error.

El subelemento env:Code de env:Fault está en sí mismo formado por un subelemento obligatorio env:Value [env:Valor], cuyo contenido está especificado en la especificación SOAP así como el subelemento opcional env:Subcode [env:Subcodigo].

11.4 Modelo de Procesamiento SOAP

Una vez establecidos varios aspectos sintácticos de un mensaje SOAP así como algunos patrones básicos de intercambio de mensajes, el modelo de procesamiento SOAP describe las acciones tomadas por un nodo SOAP el recibir un mensaje SOAP.

Mensaje SOAP mostrando una variedad de bloques de encabezado

El modelo de procesamiento de SOAP describe las acciones (lógicas) tomadas por un nodo SOAP al recibir un mensaje SOAP. Existe un requisito para el nodo que analice las partes de un mensaje que son específicas de SOAP, concretamente esos elementos del espacio de nombres SOAP "env". Tales elementos son el mismo sobre, el encabezado y el cuerpo. Un primer paso es, por supuesto, la comprobación global de que el mensaje SOAP es sintácticamente correcto. Esto es, que es conforme al conjunto de información SOAP XML, sujeto a las restricciones de uso de ciertas construcciones XML - Instrucciones de Procesamiento y Definiciones de Tipo de Documento.

11.5 Uso de Vínculos con Varios Protocolos

Los mensajes SOAP pueden intercambiarse utilizando una variedad de protocolos, incluyendo las capas de protocolos de otras aplicaciones. La especificación sobre como deben transmitirse los mensajes SOAP de un nodo a otro utilizando el protocolo de transmisión, se denomina Vínculo SOAP. SOAP define un mensaje SOAP en la forma de

un [XML Infoset], en términos de información de atributos y elementos de un documento "abstracto" denominado env:Envelope.
Cualquier env:Envelope SOAP representado en forma de conjunto de información se concretará a través de un vínculo con un protocolo, cuya tarea, entre otras cosas, es proporcionar una representación serializada del conjunto de información que puede enviarse al siguiente nodo SOAP en el camino del mensaje de un modo tal que el conjunto de información pueda ser reconstruido sin pérdida de información. En ejemplos típicos de mensajes SOAP y, ciertamente en todos los ejemplos de este documento de fundamentos, la serialización que se muestra es la de un documento [XML 1.0] bien formado. No obstante, puede haber otros vínculos con protocolos - por ejemplo un vínculo de protocolo entre dos nodos SOAP sobre un interfaz de ancho de banda limitado - donde pudiera elegirse una serialización alternativa, comprimida del mismo conjunto de información. Otro vínculo, elegido con un propósito diferente, puede proporcionar una serialización que sea una estructura encriptada representando el mismo conjunto de información.

Además de proporcionar una comprensión concreta de un conjunto de información SOAP entre nodos SOAP adyacentes a lo largo del camino de un mensaje SOAP, con un protocolo proporciona los mecanismos que dan soporte a las características que necesita una aplicación SOAP. Una característica es una especificación de una cierta funcionalidad proporcionada por un vínculo. Una descripción de característica se identifica mediante un URI, de forma que todas las aplicaciones que la referencien se aseguren la misma semántica. Por ejemplo, un escenario de uso típico podría requerir muchos intercambios petición-respuesta concurrente entre nodos SOAP adyacentes, en cuyo caso la característica que se requiere es la habilidad de correlacionar una respuesta con una petición. Otros ejemplos incluyen una característica de "canal encriptado", o una característica de "canal de entrega fiable", o una característica de patrón de intercambio de mensajes SOAP particular.

Una especificación de vínculo SOAP describe, entre otras cosas, aquellas (si existen) características que proporciona. Algunas características pueden ser proporcionadas de forma nativa por el protocolo que lo soporta. Si la característica no está disponible a través del vínculo, podría ser implementada dentro del sobre SOAP, utilizando bloques de encabezado SOAP. La especificación de una característica implementada utilizando bloques de encabezado SOAP se denomina módulo SOAP.

Por ejemplo, si los intercambios de mensajes SOAP fuesen transportados directamente sobre un protocolo de paquetes como UDP, obviamente la característica de correlación del mensaje debería ser proporcionado en cualquier otra parte, ya sea directamente por la aplicación o más probablemente como parte de los conjuntos de información SOAP que se intercambian. En este último caso, la unión con el vínculo de la característica de correlación de mensajes está determinada específicamente dentro del sobre SOAP, p.e., como un bloque de encabezado SOAP, definido en un módulo de "Correlación Petición-Respuesta" identificado por un URI. No obstante, si los conjuntos de información SOAP se fuesen intercambiados utilizando un protocolo que en sí mismo fuera petición/respuesta, la aplicación podría "heredar" implícitamente esta característica proporcionada por el vínculo, y no sería necesario dar más soporte a nivel SOAP o al de la aplicación. (De hecho, el vínculo HTTP para SOAP saca provecho de esta característica de HTTP.)

Sin embargo, un mensaje SOAP puede viajar dando varios saltos entre un remitente y el destinatario último, donde cada salto puede utilizar un vínculo con el protocolo diferente. En otras palabras, una característica (p.e., correlación de mensajes, fiabilidad, etc.) del vínculo con el protocolo de uno de los saltos, podría no tenerla otro vínculo de los que componen el camino del mensaje. SOAP en sí mismo no proporciona ningún mecanismo para ocultar las diferencias de características proporcionadas por diferentes protocolos. No obstante, cualquier característica que necesite una aplicación particular, que pudiera no estar disponible en la infraestructura de soporte, a lo largo del camino del mensaje anticipado, puede ser compensada al ser transportada como parte del conjunto de información del mensaje SOAP, p.e., como un bloque de encabezado SOAP especificado en algún módulo.

Por tanto, parece que existe un número de cuestiones a las que debe enfrentarse un diseñador de aplicaciones para cumplir la semántica de una aplicación particular, incluyendo como sacar provecho de las características nativas de los protocolos de apoyo disponibles en el entorno elegido. SOAP proporciona una infraestructura general para describir la manera en la que aplicaciones basadas en SOAP pueden elegir la utilización de características proporcionadas por el vínculo con el protocolo en el que se apoyan para intercambiar mensajes SOAP.

Entre otras cosas, una especificación de vínculo debe definir una característica particular, el patrón(es) de intercambio de mensajes que soporta. SOAP define dos patrones de intercambio, un patrón de intercambio de mensajes Petición-Respuesta SOAP donde un mensaje SOAP es intercambiado en cada dirección entre dos nodos SOAP adyacentes, y un patrón de intercambio de mensajes SOAP de Respuesta que consiste en un mensaje no-SOAP que actúa como una petición seguida de un mensaje SOAP incluido como parte de la respuesta.

SOAP también ofrece al diseñador de aplicaciones una característica general denominada la característica de método Web SOAP que permite a las aplicaciones control total sobre la elección del, así llamado, "método Web" - uno de GET, POST, PUT, DELETE cuya semántica se define en las especificaciones [HTTP 1.1] - que pueden ser utilizadas encima del vínculo. Esta característica está definida para asegurar que las aplicaciones que utilicen SOAP puedan hacerlo de forma que sea compatible con los principios arquitectónicos de la World Wide Web. (Muy brevemente, la simplicidad y escalabilidad de la Web se debe en gran parte al hecho de que existan algunos métodos genéricos (GET, POST, PUT, DELETE) que se pueden utilizar para interactuar con cualquier recurso disponible en la Web a través de un URI.) La característica de método Web SOAP es soportada por el vínculo SOAP HTTP, aunque, en principio, está disponible para todos los vínculos de SOAP con protocolos.
SOAP especifica un vínculo de protocolo estandarizado utilizando la infraestructura de SOAP, en concreto como se utiliza SOAP junto con HTTP como el protocolo de apoyo. SOAP Versión 1.2 se restringe a sí mismo a la definición de un vínculo HTTP al que sólo permite el uso del método POST en conjunción con el patrón de intercambio de mensajes Petición-Respuesta. Otras especificaciones futuras podrían definir vínculos SOAP con HTTP o con otros medios de transporte que hagan uso de otros métodos Web (p.e., PUT, DELETE).

Las siguientes secciones muestran ejemplos de dos vínculos con protocolos diferentes, concretamente con [HTTP 1.1] y correo electrónico.

11.6 El Vínculo SOAP HTTP

HTTP tiene un modelo bien conocido y un patrón de intercambio de mensajes. El cliente identifica al servidor mediante un URI, se conecta a él utilizando la red TCP/IP, envía un mensaje de petición HTTP y recibe un mensaje de respuesta HTTP sobre la misma conexión TCP. HTTP relaciona implícitamente su mensaje de petición con su mensaje de respuesta; por tanto, una aplicación que utilice este vínculo puede elegir inferir una correlación entre el mensaje SOAP enviado en el cuerpo de un mensaje de petición HTTP y el mensaje SOAP devuelto en una respuesta HTTP. De forma similar, HTTP identifica el punto de destino en el servidor mediante un URI, el Request-URI [URI-Solicitado], que también puede servir como el identificador de un nodo SOAP en el servidor.

HTTP permite múltiples intermediarios entre el cliente inicial y el servidor de origen identificado por el Request-URI, en cuyo caso el modelo petición/respuesta es una serie de esos pares. Obsérvese, no obstante, que los intermediarios SOAP son diferentes de los intermediarios HTTP.

El vínculo con HTTP de SOAP, hace uso de una característica de método Web SOAP para permitir a las aplicaciones elegir el, así llamado, método Web - restringiéndolo a un GET o un POST - para utilizar encima del intercambio de mensajes HTTP. Además, hace uso de dos patrones de intercambio de mensajes que ofrecen a las aplicaciones dos maneras de intercambiar mensajes SOAP vía HTTP: 1) el uso del método HTTP POST para transportar mensajes SOAP en los cuerpos de los mensajes de peticiones y respuestas HTTP, y 2) el uso del método HTTP GET en una petición HTTP para devolver un mensaje SOAP en el cuerpo de una respuesta HTTP. El primer patrón de uso es la instancia de HTTP-específica de una característica del vínculo llamada patrón de intercambio de mensajes petición-respuesta SOAP, mientras que el segundo utiliza una característica denominada patrón de intercambio de mensajes de respuesta SOAP.

El propósito de facilitar estos dos tipos de usos es el de acomodar los dos paradigmas de interacción que está bien establecidos en la World Wide Web. El primer tipo de interacción permite el uso de datos en el cuerpo de un HTTP POST para crear o modificar el estado de un recurso identificado por el URI al que se dirige la petición HTTP. El segundo tipo de patrón de interacción ofrece la posibilidad de utilizar una petición HTTP GET para obtener una representación de un recurso sin alterar su estado en modo alguno.

En el primer caso, el aspecto de interés específico SOAP es que el cuerpo de la petición HTTP es un mensaje SOAP que tiene que ser procesado (por el modelo de procesamiento SOAP) como una parte del procesamiento específico de la aplicación requerido para ajustarse a la semántica de POST. En el segundo caso, el uso típico que se prevé es el caso en el que la representación del recurso que se pide no sea devuelto como HTML, ni como un documento genérico XML, sino como un mensaje SOAP. Esto es, el encabezado del tipo de contenido en el mensaje de respuesta lo identifica como del tipo "application/soap+xml". Presumiblemente, habrá editores de recursos en la Web que determinen que tales recursos se harán disponibles y serán obtenidos mejor en la forma de mensajes SOAP. Obsérvese, no obstante, que los recursos pueden, en general, hacerse disponibles en múltiples representaciones, y la representación preferida o deseada y la aplicación que realiza la petición indica la representación preferida o deseada utilizando el encabezado HTTP Accept.

Un aspecto más del vínculo SOAP HTTP es la cuestión de como una aplicación determina cual de esos dos tipos de patrones de mensaje utilizar. SOAP sirve de guía en las circunstancias en las que las aplicaciones pueden utilizar uno de los dos patrones de intercambio de mensajes. (Es consejo - aunque muy fuerte - como el descrito en la forma de un " SHOULD" ["DEBERÍA"] en las especificaciones más que un requisito absoluto identificado por la palabra "MUST" ["DEBE"], interpretándose esas palabras según se encuentran definidas en IETF [RFC 2119].) El patrón de intercambio de respuesta SOAP con el método HTTP GET es utilizado cuando una aplicación asegura que el intercambio de mensajes es con el propósito de obtención de información, en el que el recurso de información permanece "intocable" como resultado de la interacción. A tales interacciones se les refiere como seguras y potentes en la especificación HTTP. Como el uso de HTTP SOAP GET no permite un mensaje SOAP en la petición, las aplicaciones que necesiten características en la interacción de ida que sólo puedan ser soportadas por una expresión específica al vínculo dentro del conjunto de información SOAP (p.e., como bloques de encabezado SOAP), no pueden hacer, obviamente, uso de este patrón de intercambio de mensajes. Observe que el vínculo HTTP POST está disponible para ser utilizado en todos los casos.

Las siguientes subsecciones proporcionan ejemplos del uso de estos dos patrones de intercambio de mensajes para el vínculo HTTP.

11.7 Uso de SOAP HTTP GET

El uso del vínculo HTTP con el patrón de intercambio de mensajes de Respuesta SOAP se encuentra restringido al método HTTP GET. Esto significa que la respuesta a una petición HTTP GET desde un nodo SOAP solicitante es un mensaje SOAP en la respuesta HTTP.

El encabezado HTTP Accept se utiliza par indicar la representación preferida del recurso que se solicita, que en este ejemplo es de tipo "application/soap+xml" para ser consumido por una máquina cliente, en ves de tipo "text/html" para ser mostrado por un navegador cliente y consumido por un humano.

11.8 Uso de SOAP HTTP POST

El uso del vínculo HTTP con el patrón de intercambio de mensajes Petición-Respuesta SOAP está restringido al método HTTP POST. Obsérvese que el uso de este patrón de intercambio de mensajes en el vínculo SOAP HTTP se encuentra disponible para todas las aplicaciones, ya estén envueltas en el intercambio de datos generales en XML, o RPCs (como en los siguientes ejemplos) encapsulados en mensajes SOAP.

Cuando se ubican mensajes SOAP en cuerpos HTTP, el encabezado HTTP Content-type debe tener el valor "application/soap+xml". (El parámetro opcional charset [juego de caracteres], que puede tomar los valores "utf-8" o "utf-16", se muestra en este ejemplo, pero sin que las reglas independientes de juegos de caracteres de [XML 1.0] se apliquen al cuerpo de la petición HTTP.)

11.9 Uso de SOAP Compatible con la Arquitectura Web

Uno de los conceptos esenciales de la World Wide Web es aquel del URI como identificador de recursos. Los servicios SOAP que utilizan el vínculo HTTP y desea interoperar con otro software Web deberían utilizar URIs para indicar todos los recursos importantes en su servicio. Por ejemplo, un uso muy importante - verdaderamente predominante - de la World Wide Web es la pura obtención de información, en la que la representación de un recurso disponible, identificado por un URI, se obtiene utilizando una petición HTTP GET sin afectar al recurso en modo alguno. (Esto se llama un método seguro e impotente en terminología HTTP.) El punto clave es que el editor del recurso hace disponible su URI, que los consumidores pueden "GET" ["OBTENER"].

Hay muchos casos en los que los mensajes SOAP se diseñan para usos que son puramente de obtención de información, como aquellas en las que se solicita el estado de algún recurso (u objeto, en términos de programación), opuestamente a usos en los que se realiza manipulación del recurso. En tales casos, el uso de un cuerpo SOAP para transportar la petición del estado, con un elemento del cuerpo representando al objeto en cuestión, es visto como contrario al espíritu de la Web porque el recurso no está identificado por el Request-URI del HTTP GET. (En algunas implementaciones SOAP/RPC, el Request-URI de HTTP no suele ser el identificador del recurso en sí, sino de alguna entidad intermedia que tiene que evaluar el recurso SOAP para identificar el recurso.)

11.10 SOAP Sobre Correo Electrónico

Los desarrolladores de aplicaciones pueden utilizar la infraestructura de correo electrónico de Internet para transmitir mensajes SOAP ya sean como mensajes de correo electrónico de texto o como adjuntos. Los ejemplos que se muestran a continuación muestran un modo de transmitir mensajes SOAP, y deben ser tomados como el modo estándar de hacerlo. Las

especificaciones SOAP Versión 1.2 no especifican tal vínculo. Sin embargo, existe una Nota W3C no-normativa [SOAP Email Binding] que describe un vínculo de SOAP con el correo electrónico, su propósito principal es comenzar a demostrar la aplicación de la Infraestructura general de Vínculos con el Protocolo SOAP descrita en [SOAP Part 1].

Aunque el correo electrónico es un intercambio de mensajes en un solo sentido, y no se da ninguna garantía de entrega, infraestructuras como la de la especificación Simple Mail Transport Protocol (SMTP) [Protocolo para el Transporte de Correo Simple] [SMTP] ofrecen un mecanismo de notificación de entrega que, en el caso de SMTP, se denominan Delivery Status Notification (DSN) [Notificación de Estado de Entrega] y Message Disposition Notification (MDN) [Notificación de Disposición de Mensaje]. Estas notificaciones toman la forma de mensajes de correo electrónico enviados a la dirección de correo electrónico especificada en el encabezamiento del mensaje de correo. Las aplicaciones, así como los usuarios finales del correo, pueden utilizar estos mecanismos para proporcionar el estado de una transmisión de correo electrónico, pero estos, si se existiesen, serían notificaciones al nivel SMTP.

El desarrollador de aplicaciones debe comprender completamente la capacidad y limitaciones de estas notificaciones de entrega o el riesgo de asumir que haya existido una entrega del mensaje con éxito cuando podría no haberse producido.

Los mensajes de estado de entrega SMTP son separados del procesamiento del mensaje en la capa SOAP. Las respuestas SOAP resultantes a los datos SOAP serán devueltas a través de un mensaje de correo electrónico nuevo que podría tener o no un enlace con el mensaje de la petición original al nivel SMTP. El uso del encabezado In-reply-to: [En-respuesta-a] según [RFC 2822] puede conseguir una correlación al nivel SMTP, pero no implica necesariamente una correlación al nivel SOAP.

11.11 Vínculo SOAP HTTP

En el vínculo SOAP 1.2 HTTP, el encabezado HTTP SOAPAction definido en SOAP 1.1 ha sido eliminado, y se ha solicitado un nuevo código de estado HTTP 427 a IANA para indicar (a discreción del servidor HTTP del origen) que su presencia sea requerida por la aplicación del servidor. Los contenidos del encabezado HTTP original SOAPAction se expresan ahora como un valor de un parámetro (opcional) "action" del tipo "application/soap+xml" que se señala en el vínculo HTTP.

En el vínculo SOAP 1.2 HTTP, el encabezado Content-type debería ser "application/soap+xml" en vez de "text/xml" como ocurría en SOAP 1.1. El registro en IETF de este nuevo tipo de medio está pendiente [SOAP MediaType].

SOAP 1.2 permite descripciones de granularidad más fina en el uso de varios códigos de estado HTTP 2xx, 3xx, 4xx.

El soporte para la infraestructura de extensiones HTTP ha sido eliminado en SOAP 1.2.

SOAP 1.2 proporciona un patrón de intercambio de mensajes adicional que puede ser utilizado como parte del vínculo HTTP que permita el uso de HTTP GET para recuperaciones de información segura.

RPC
SOAP 1.2 proporciona un elemento accesorio rpc:result para RPCs.
SOAP 1.2 proporciona varios códigos de error adicionales en el espacio de nombres RPC.
SOAP 1.2 ofrece ayuda en una aproximación para definir, de un modo Web-amigable, RPCs en las que el propósito del procedimiento es la pura obtención "segura" de información.

SOAP codificaciones
Se ha formulado un modelo abstracto de datos basado en un grafo dirigido etiquetado para SOAP 1.2. Las codificaciones SOAP 1.2 son dependientes de este modelo de datos. Las convenciones SOAP RPC son dependientes de este modelo de datos, pero no tienen dependencias con la codificación SOAP. El soporte de las convenciones de codificación SOAP 1.2 y de las de SOAP RPC 1.2 es opcional.

La sintaxis para la serialización de un vector/matriz ha cambiado en SOAP 1.2 de la de SOAP 1.1.

El soporte proporcionado en SOAP 1.1 para vectores/matrices transmitidas parcialmente y para matrices casi vacías no está disponible en SOAP 1.2.

SOAP 1.2 permite la serialización en línea (embebida) de valores multiref.

El atributo href en SOAP 1.1 (del tipo xs:anyURI) se llama enc:ref en SOAP 1.2 y es de tipo IDREF.
En SOAP 1.2, los accesorios de tipos compuestos omitidos son iguales que NILs.
SOAP 1.2 proporciona varios subcódigos de error para indicar errores de codificación.
Los tipos en los nodos son opcionales en SOAP 1.2.
SOAP 1.2 ha eliminado los valores genéricos compuestos del Modelo de Datos SOAP.
SOAP 1.2 ha añadido un atributo opcional enc:nodeType a los elementos codificados utilizando codificación SOAP que identifica su estructura (es decir, a un valor simple, una estructura de datos o un vector o matriz).
SOAP Parte 1 Apéndice A proporciona reglas de gestión de versiones para un nodo SOAP que pueda soportar la transición de versión de [SOAP 1.1] a SOAP Versión 1.2. En concreto, define un bloque de encabezado env:Upgrade que puede ser utilizado por un nodo SOAP 1.2 al recibir un mensaje [SOAP 1.1], para enviar un mensaje de error SOAP al remitente indicando la versión de SOAP que soporta.

11.12 Referencias

Esta versión:
http://www.w3.org/TR/2003/REC-soap12-part0-20030624/
Última versión:
http://www.w3.org/TR/soap12-part0/
Versión anterior:
http://www.w3.org/TR/2003/PR-soap12-part0-20030507/
Editor:
Nilo Mitra (Ericsson)

[SOAP Parte 1] Recomendación W3C "SOAP 1.2 Parte 1: Infraestructura de Mensajes", Martin Gudgin, Marc Hadley, Jean-Jacques Moreau, Henrik Frystyk Nielsen, 24 de Junio 2003 (Ver http://www.w3.org/TR/2003/REC-soap12-part1-20030624/.)
[SOAP Parte 2] Recomendación W3C "SOAP 1.2 Parte 2: Adjuntos", Martin Gudgin, Marc Hadley, Jean-Jacques Moreau, Henrik Frystyk Nielsen, 24 de Junio 2003 (Ver http://www.w3.org/TR/2003/REC-soap12-part2-20030624.)
[RFC 2396] IETF "RFC 2396: Identificadores de Recursos Uniformes (URI): Sintaxis Genérica", T. Berners-Lee, R. Fielding, L. Masinter, Agosto de 1998. (Ver http://www.ietf.org/rfc/rfc2396.txt.)
[HTTP 1.1] IETF "RFC 2616: Protocolo de Transferencia de Hipertexto -- HTTP/1.1", R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, T. Berners-Lee, Enero de 1997. (Ver http://www.ietf.org/rfc/rfc2616.txt.)
[XML 1.0] Recomendación W3C "Lenguaje de Marcas Extensible (XML) 1.0 (Segunda Edición)", Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, 6 de Octubre de 2000. (Ver http://www.w3.org/TR/2000/REC-xml-20001006.
[Espacios de Nombres en XML] Recomendación W3C "Espacios de Nombres en XML", Tim Bray, Dave Hollander, Andrew Layman, 14 de Enero de 1999. (Ver http://www.w3.org/TR/1999/REC-xml-names-19990114/.)
[Esquema XML Parte 1] Recomendación W3C "Esquema XML Parte 1: Estructuras", Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, 2 de Mayo de 2001. (See http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/)
[Esquema XML Parte 2] Recomendación W3C "Esquema XML Parte 2: Tipos de Datos", Paul V. Biron, Ashok Malhotra, 2 de Mayo 2001. (See http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/)
[SMTP] SMTP se define en una serie de RFCs:
RFC 2822 Formato de Mensaje en Internet. (Ver http://www.ietf.org/rfc/rfc2822.txt.)
RFC 2045 Extensiones Multipropósito al Correo de Internet (MIME) Parte Uno: Formato de los Cuerpos de Mensajes en Internet. (Ver http://www.ietf.org/rfc/rfc2045.txt.)
RFC 2046 Extensiones Multipropósito al Correo de Internet (MIME) Parte Dos: Tipos de Medios. (Ver http://www.ietf.org/rfc/rfc2046.txt.)
RFC 1894 Un Formato de Mensaje Extensible para Notificaciones de Estado de Entrega. (Ver http://www.ietf.org/rfc/rfc1894.txt.)
RFC 2852 Extensión al Servicio de Entrega por SMTP. (Ver http://www.ietf.org/rfc/rfc2852.txt.)

RFC 2298 Un Formato de Mensaje Extensible para Notificaciones de Disposición. (See http://www.ietf.org/rfc/rfc2298.txt.)

[RDF] Recomendación W3C "Especifiación del Modelo y la Sintaxis de la Infraestructura de Descripción de Recursos (RDF)", O. Lassila y R. Swick, Editores. World Wide Web Consortium. 22 de Febrero de 1999. (Ver http://www.w3.org/TR/REC-rdf-syntax.)

[SOAP 1.1] Nota del W3C "Protocolo Simple de Acceso a Objetos (SOAP) 1.1", Don Box et al., 8 de Mayo, 2000 (Ver http://www.w3.org/TR/2000/NOTE-SOAP-20000508/)

[XML Infoset] Recomendación W3C "XML Information Set", John Cowan, Richard Tobin, 24 de Octubre de 2001. (Ver http://www.w3.org/TR/xml-infoset/)

[SOAP MediaType] Borrador IETF "El tipo de medio ' application/soap+xml' ", M. Baker, M. Nottingham, "draft-baker-soap-media-reg-03.txt", 29 de Mayo de 2003. (Ver http://www.ietf.org/internet-drafts/draft-baker-soap-media-reg-03.txt, obsérvese que este Borrador expira en Noviembre de 2003)

[SOAP Email Binding] Nota W3C "SOAP Versión 1.2 Vínculo con el Correo Electrónico", Highland Mary Mountain et al, borrador de 13 de Junio de 2002. (Ver http://www.w3.org/TR/2002/NOTE-soap12-email-20020626.)

[XML Base] Recomendación W3C "XML Base", Jonathan Marsh, 27 de Junio de 2001. (Ver http://www.w3.org/TR/2001/REC-xmlbase-20010627/)

[RFC 2119] IETF "RFC 2119: Palabras clave para utilizar en RFCs indicando los Niveles de Requisito", S. Bradner, Marzo de 1997. (Ver http://www.ietf.org/rfc/rfc2119.txt.)

12. XML.

XML, o Lenguaje de Marcado Extensible (Extensible Markup Language), es un lenguaje de marcado que se puede usar para crear etiquetas propias. Fue creado por el Consorcio para la World Wide Web (W3C) para superar las limitaciones de HTML, el Lenguaje de Marcado de Hipertexto que es la base para todas las páginas Web. Aunque SGML ha sido usado durante décadas en la industria para publicación, su complejidad ha intimidado a mucha gente que, en otro caso, podría haberlo usado. XML fue diseñado pensando en la Web.

¿Porqué se necesita XML?

HTML es el lenguaje de marcado con más éxito del momento. Puede ver las etiquetas HMTL más simples en casi cualquier dispositivo, desde palmtops hasta los mainframes, e incluso puede convertir HTML en voz o en otros formatos con las herramientas adecuadas. Dado el éxito de HTML, ¿porqué W3C creó XML?. Para responder a esta pregunta observe el siguiente documento:

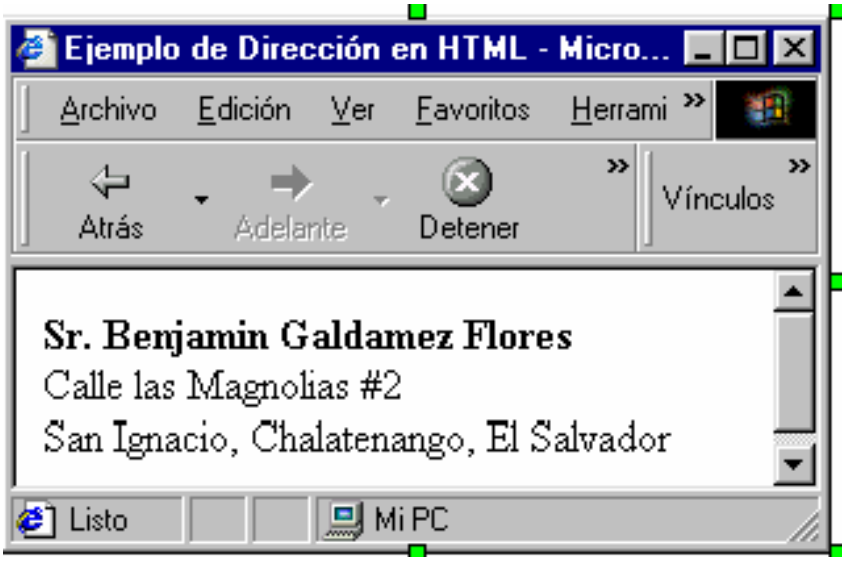
* Sr. Benjamin Galdamez Flores*
*
*
Calle las Magnolias #2
*
*
**San Ignacio, Chalatenango, El Salvador **

El problema con HTML es que fue diseñado pensando en los humanos. Incluso sin ver el documento anterior en un navegador, se puede imaginar que se trata de la dirección postal de alguien. Como humanos, tenemos la inteligencia para comprender el significado e intención de muchos documentos. Una máquina, desafortunadamente no puede hacerlo. Mientras que las etiquetas de este documento le dicen al navegador cómo debe mostrar la información, no le dicen lo que la información es. Nosotros los humanos sabemos que es una dirección, pero la máquina no.

12.1 Representar HTML

Para representar HTML, el navegador únicamente sigue las instrucciones en el documento HTML. La etiqueta de párrafo le dice al navegador que empiece la presentación en una nueva línea, típicamente con una línea en blanco antes, mientras que las dos etiquetas de ruptura le dicen al navegador que avance hasta la siguiente línea sin dejar líneas en blanco en medio.

Aunque el navegador da un bonito formato al documento, la máquina aún no sabe que se trata de una dirección postal.



12.2 Procesamiento de HTML

Para continuar la discusión con el ejemplo del documento HTML, considere la tarea de extraer el nombre de la calle de la dirección. Aquí se presenta un algoritmo (intencionadamente frágil) para encontrar la marca HTML para el nombre de calle: Si se encuentra un párrafo con dos etiquetas <br>, el nombre de la calle es la segunda palabra después de la primera etiqueta de ruptura. Aunque este algoritmo funciona con el ejemplo, existen en el mundo una gran cantidad de direcciones perfectamente válidas para las cuales no puede funcionar. Incluso si se pudiera escribir un algoritmo que encontrase el nombre de calle para cualquier dirección escrita en HTML, existe un gran número de párrafos con dos etiquetas de ruptura cuyo contenido no es una dirección postal. La escritura de un algoritmo que busque en cualquier párrafo HTML y encuentre cualquier nombre de calle que contenga sería extremadamente difícil, sino imposible.

Un documento XML de ejemplo.

Con XML, se puede asignar algún significado a las etiquetas en el documento. Más importante aún, también resulta fácil para una maquina el procesar la información. Se puede extraer el nombre de calle de un documento simplemente localizando el contenido rodeado por las etiquetas <calle> y </calle>, técnicamente conocido como el elemento <calle>.

```
<direccion>
        <nombre>
                <titulo>Sr.</titulo>
                <nombre>
                        Benjamin
                </nombre>
                <apellidos>
```

```
                Galdamez Flores
            </apellidos>
        </nombre>

        <calle>
            Calle las Magnolias
        </calle>
        <casa>
            #2
        </casa>
        <ciudad>
            San Ignacio
        </ciudad>
        <departamento>
            Chalatenango
        </departamento>
        <pais>
            El Salvador
        </pais>
</direccion>
```

12.3 Etiquetas, elementos y atributos.

Existen tres términos comúnmente usados para describir las partes de un documento XML: etiquetas, elementos y atributos. Aquí está un documento de ejemplo que ilustra estos términos:

```
<direccion>
        <nombre>
            <titulo>Sr.</titulo>
            <nombre>
                Benjamin
            </nombre>
            <apellidos>
                Galdamez Flores
            </apellidos>
        </nombre>
        <calle>
            Calle las Magnolias
        </calle>
        <casa>
            #2
        </casa>
        <ciudad departamento='Chalatenango'>San Ignacio</ciudad>
        <pais>
```

```
            El Salvador
        </pais>
</direccion>
```

Una etiqueta es un texto entre el símbolo menor que (<) y el símbolo mayor que (>). Existen etiquetas de inicio (como <nombre>) y etiquetas de fin (como </nombre>). Un elemento consta de la etiqueta de inicio, la etiqueta de fin y de todo aquello que este entre ambas. En el ejemplo anterior, el elemento <nombre> contiene tres elementos hijos: <titulo>, <nombre>, y <apellidos>.

Un atributo es un par nombre-valor dentro de la etiqueta de inicio de un elemento. En este ejemplo, departamento es un atributo del elemento <ciudad>, en ejemplos anteriores <departamento> era un elemento.

Cómo está cambiando XML la Web

Los desarrolladores pueden usar XML para crear documentos con datos que se auto-describen, estos documentos se están usando para aumentar el rendimiento de la Web. Se muestran algunas áreas a continuación:

XML simplifica el intercambio de datos. Debido a que diferentes organizaciones (en incluso diferentes partes dentro de una misma organización) raramente utilizan un único conjunto de herramientas, esto puede suponer una cantidad de trabajo significativa para comunicar las aplicaciones. Usando XML cada grupo crea una única utilidad que transforma sus formatos de datos internos en XML y viceversa. Lo mejor de todo, hay una gran probabilidad de que sus suministradores de software, ya proporcionen herramientas que transformen sus registros de base de datos(o directorios LDAP, o ordenes de compra, etc.) desde y hacia XML.

XML permite un código más ligero. Debido a que los documentos XML pueden ser estructurados para identificar cada pieza importante de información (así como las relaciones entre dichas piezas), es posible escribir código que procese estos documentos XML sin intervención humana. El hecho de que los proveedores de software hayan gastado cantidades masivas de tiempo y dinero construyendo herramientas de desarrollo. XML significa que escribir ese código es un proceso relativamente simple.

XML permite búsquedas más rápidas. Aunque los motores de búsqueda han mejorado sustancialmente durante los años, es todavía bastante común encontrar resultados erróneos en una búsqueda. Si se están buscando páginas HTML de alguien llamado "Chip", se pueden encontrar páginas sobre chips de chocolate, chips de ordenador, chips de madera y montones de otros enlaces inútiles. Buscando en documentos XML por elementos <nombre> que contengan el texto Chip debería darnos un conjunto de resultados mucho mejor.

12.4 Reglas de los documentos XML.

La especificación XML requiere un parser o una pieza de código que intenta leer un documento e interpretar su contenido para rechazar los documentos XML que no sigan las reglas básicas. Muchos analizadores HTML aceptarán marcados inconclusos o inconsistentes haciendo aproximaciones sobre lo que el escritor del documento intentaba. Para evitar el desorden de estructuras encontrado en la mayoría de los documentos HTML, los creadores de XML decidieron reforzar la estructura del documento desde el principio.

DOCUMENTOS INVÁLIDOS, VÁLIDOS Y BIEN FORMADOS

Hay tres tipos de documentos XML:

- Documentos inválidos no siguen las reglas de sintaxis definidas por la especificación XML. Si un desarrollador tiene reglas definidas de lo que ese documento puede contener en una DTD o Esquema, y el documento no las sigue, ese documento es inválido.

- Documentos válidos siguen tanto las reglas de sintaxis XML como las reglas definidas en su propio DTD o Esquema.

- Documento bien formado sigue las reglas de sintaxis XML, pero no tiene un Esquema o DTD.

EL ELEMENTO RAÍZ

Un documento XML debe estar contenido en un elemento único. Este elemento único es llamado el elemento raíz y contiene todo el texto y cualquier otro elemento en el documento. En el ejemplo siguiente, el documento XML está contenido en un elemento único, el elemento <greeting>. Nótese que el documento tiene un comentario el cual, aunque está fuera del elemento raíz, es perfectamente legal.

```
<?xml version="1.0"?>
<!-Un documento valido-->
<saludo>
        Hola Mundo!
</saludo>
```

Un documento que no contiene un único elemento raíz:

```
<?xml version="1.0"?>
<!-Un documento no valido -->
<saludo>
        Hola Mundo!
```

```
</saludo>

<saludo>
Hola, el Mundo!
</saludo>
```

Es necesario un parser XML para rechazar el documento, independientemente de la información que pueda contener.

Los elementos XML no pueden solaparse. El marcado ilegal:

```
<!-marcado XML ilegal -->
<p>
<b>Yo <i>amo de verdad
</b> XML.
</i>
</p>
```

Si comienza un elemento <i> dentro de un elemento <b> también deberá terminarlo dentro. Si se quiere que el texto XML aparezca en cursiva, hay que añadir un segundo elemento <i> para corregir el marcado:

```
<!-marcado XML legal -->
<p>
<b>Yo <i>amo de verdad</i></b>
<i>XML.</i>
</p>
```

Un parser XML aceptará solamente este marcado; Los parser HTML de la mayoría de los navegadores aceptarán ambos.

LAS ETIQUETAS DE FIN SON OBLIGATORIAS

No puede olvidarse de ninguna etiqueta de fin. En siguiente ejemplo, el marcado no es legal debido a que no hay etiquetas de fin de párrafo (</p>). Aunque esto es aceptable en HTML (y, en algunos casos, en SGML), un parser XML lo rechazaría.

```
<!-Marcado XML NO LEGAL -->
<p>Yada yada yada...
<p>Yada yada yada...
<p>...
```

Si un elemento carece de contenido se le llama elemento vacío; los elementos HTML para ruptura (<br>) e imagen (<img>) son dos ejemplos. En los elementos vacíos en XML pueden ponerse lo barra de cierre en la etiqueta de inicio. Los dos elementos de ruptura y los dos elementos de imagen a continuación significan la misma cosa para un parser XML:

```
<!- Dos rupturas equivalentes -->
<br></br>
<br/>
<!- Dos elementos de imagen equivalentes -->
<img src="../img/c.gif"></img>
<img src="../img/c.gif" />
```

Los elementos son sensibles a mayúsculas.

Los elementos XML son sensibles a mayúsculas. En HTML, <h1> y <H1> son lo mismo; en XML no. Si intenta terminar un elemento <h1> con una etiqueta </H1>, obtendrá un error. En el ejemplo siguiente, el primero presenta un encabezado ilegal, mientras que el último es correcto.

```
<!- marcado XML ILEGAL -->
<h1>Los elementos son sensibles a mayúsculas</H1>

<!- marcado XML LEGAL -->
<h1> Los elementos son sensibles a mayúsculas</h1>
```

LOS ATRIBUTOS DEBEN TENER VALORES ENTRECOMILLAS

Existen un par de reglas para los atributos en los documentos XML:

- Los atributos deben tener valores
- Esos valores deben estar entrecomillas.

Compare los dos ejemplos que vienen a continuación. El marcado del primero es legal en HTML, pero no en XML. Para obtener el equivalente en XML, se le debe dar un valor al atributo y debe colocar entrecomillas.

```
<!- marcado XML ILEGAL -->
<ol compact>

<!- marcado XML LEGAL -->
<ol compact="yes">
```

Puede usar comillas simples o dobles siempre y cuando sea consecuente. Si el valor del atributo contiene un single o un double puede usarse el otro tipo de comillas para rodear el valor (como en nombre='Doug' s car'), o usar las entidades " para comillas dobles y ' para comillas simples. Una entidad es un símbolo, como ", que el parser XML reemplazará con otro texto como ".

12.5 Declaraciones XML

Muchos documentos XML comienzan con una declaración XML que proporciona al parser información básica sobre el documento. Una declaración XML está recomendada, pero no es obligatoria. Si existe una, debe ser lo primero que aparezca en el documento.

La declaración puede contener hasta tres pares nombre-valor (mucha gente les llama atributos, aunque técnicamente no lo son). versión es la versión de XML usada; actualmente este valor debe ser 1.0. encoding es el conjunto de caracteres usado en el documento. El conjunto de caracteres ISO-8859-1 referenciado en esta declaración incluye todos los caracteres usados en la mayoría de los lenguajes de Europa Occidental. Si no se especifica encoding el parser XML asume que los caracteres pertenecen al conjunto UTF-8, un estándar Unicode que soporta virtualmente cada carácter e ideograma de cualquier lenguaje del mundo.

<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>

Finalmente, standalone, que solo puede ser yes o no, define si este documento puede ser procesado sin leer otros archivos. Por ejemplo, si el documento XML no referencia ningún otro archivo, puede especificarse standalone="yes". Si el documento XML referencia otros archivos que describen lo que el documento puede contener, se debe especificar standalone="no". Debido a que standalone="no" es la opción por defecto, raramente veremos standalone en las declaraciones XML.

OTRAS COSAS EN LOS DOCUMENTOS XML

Hay algunas otras cosas que se pueden encontrar en un documento XML:

- **Comentarios:** los comentarios pueden aparecer en cualquier lugar en el documento; incluso antes o después del elemento raíz. Un comentario comienza con <!- y termina con -->. Un comentario no puede contener un guión doble (--), excepto al final; con esta excepción, un comentario puede contener cualquier cosa. Aún más importante, cualquier marca dentro de un comentario será ignorada; si quiere eliminar una gran sección de un documento XML simplemente introdúzcala en un comentario. Ejemplo que contiene un comentario.

  <!-Esto es una PI para Cocoon: -->
  <?cocoon-process type="sql"?>

- **Instrucciones de procesamiento:** Una instrucción de procesamiento es una marca que se interpretará como un segmento de código. En el ejemplo previo hay una instrucción de procesamiento (algunas veces llamadas PI) para Cocoon, un marco de procesado para XML de Apache Software Foundation. Cuando Cocoon está procesando un documento XML busca instrucciones de procesamiento que comiencen con cocoon-process, entonces procesa el documento XML de acuerdo a ellas. En este ejemplo, el atributo type="sql" le dice a Cocoon que el documento XML contiene una sentencia SQL.

<!- He aquí una entidad: -->
<!ENTITY dw "developerWorks">

- **Entidades:** El ejemplo previo define una entidad para el documento. Dondequiera que el procesador XML encuentre la cadena &dw;, la reemplazará con la cadena developerWorks. La especificación XML también define cinco entidades que pueden usarse en lugar de varios caracteres especiales. Las entidades son:

< para el símbolo menor que (<)
> para el símbolo mayor que (>)
" para las comillas dobles (“)
' para la comilla simple o apostrofe (‘)
& para el ampersand (&).

Namespaces

El poder de XML proviene de su flexibilidad, del hecho de que millones de otras personas puedan definir propias etiquetas para describir datos. En el ejemplo del documento XML con el nombre y dirección de una persona, ese documento incluía el elemento <titulo> para el título o tratamiento de cortesía, una elección perfectamente razonable como nombre de un elemento. Si se crea una librería online, podría elegir el crear un elemento <titulo> para almacenar el título de un libro.

Si crea una compañía hipotecaria online, podría elegir crear un elemento <titulo> Para el título de una propiedad. Todas son elecciones razonables, pero todas ellas crean elementos con el mismo nombre. ¿Cómo saber si, dado un elemento <titulo> se refiere a una persona, un libro o una propiedad? Con los namespaces. Para usar un namespace, debe definir un prefijo namespace y mapearlo a una cadena particular. Así es cómo se deberían definir prefijos namespace para nuestros tres elementos <titulo>:

<?xml version="1.0"?>
<resumen_usuario
xmlns:direccion="http://www.xyz.com/direcciones/"
xmlns:libros="http://www.zyx.com/libros/"
xmlns:hipoteca="http://www.yyz.com/hipotecas/"
>

... <direccion:nombre><titulo>Mrs.</titulo> ... </direccion:nombre> ...
... <libros:titulo>El Señor de los Anillos</libros:titulo> ...
... <hipoteca:titulo>NC2948-388-1983</hipoteca:titulo> ...

La definición de un namespace para un elemento particular significa que todos los elementos hijos pertenecen al mismo namespace. El primer elemento <titulo> pertenece al namespace direccion debido a que su elemento padre <direccion:Nombre>, ya pertenecía a ese namespace.

El punto final: La cadena de una definición de namespace es solo una cadena. Si, esa cadena parece una URL, pero no lo es. Podría definir xmlns:direccion="mike" y funcionaría exactamente igual, Lo único que importa acerca de la cadena del namespace es que sea única; por esto la mayor parte de las definiciones de namespace parecen URLs. Los parser XML no van a la dirección http://www.zyx.com/libros/ para buscar una DTD o un Esquema, simplemente usan esos textos como cadenas. Es confuso, pero así es como funcionan los namespaces.

Definición del contenido de un documento.

Existen dos maneras para representar datos:

- Un método es usar un **Document Type Definition o DTD**. Un DTD define los elementos que pueden aparecer en un documento XML, el orden en el cual pueden aparecer, cómo pueden estar anidados y otros detalles básicos de la estructura del documento XML. Los DTD son parte de la especificación original de XML y son muy similares a los DTDs de SGML.

- El otro método es usar un **Esquema XML (XML Schema).** Un esquema puede definir todas las estructuras de documento que pudieran definirse con DTD y además, puede definir tipos de datos y reglas mucho más complicadas de las que pueden hacerse con DTD. El W3C desarrollo la especificación de Esquemas XML un par de años después que la especificación original XML.

12.6 Interfaces de programación para XML

Estas interfaces proporcionan a los desarrolladores una forma consistente de trabajar con los documentos XML. Hay muchos APIs disponibles; Hay cuatro de la más populares y útiles: el Document Object Model (DOM), el Simple API for XML (SAX), JDOM, y el Java API for XML Parsing(JAXP) .

Document Object Model

El Document Object Model, comúnmente llamado DOM, define un conjunto de interfaces a la versión interpretada por parser de un documento XML. El parser lee el documento

completo y construye un árbol en la memoria, de forma que el código puede entonces utilizar las interfaces DOM para manipular ese árbol. Es posible moverse a través del árbol para ver lo que el documento original contenía, se pueden borrar secciones del árbol, reordenarlo, añadir nuevas ramas, etc. DOM fue creado por el W3C, y es una Recomendación Oficial del consorcio.

Limitaciones de DOM

DOM proporciona un amplio conjunto de funciones que pueden usarse para interpretar y manipular un documento XML, pero esas funciones tienen un precio. Tan pronto como el DOM para XML original empezó a desarrollarse, mucha gente en la lista de correos de XML-DEV empezó a preocuparse acerca de lo siguiente:

- DOM construye un árbol en la memoria con el documento entero. Si el documento es muy grande ser requiere una cantidad significativa de memoria.

- DOM crea objetos que los representan todo en el documento original, incluyendo elementos, texto, atributos y espacios en blanco. Si solo se tiene interés en una pequeña parte del documento original, es extremadamente costoso crear todos esos objetos que nunca se usarán.

- Un parser DOM tiene que leer el documento entero antes de que el código pueda tomar el control. Para documentos muy grandes esto puede causar un retraso significativo. Estas son algunas de las debilidades derivadas del diseño del Document Object Model; independientemente de estas, el API DOM es un parser muy útil para interpretar documentos XML.

Simple API for XML

Como alternativa a las carencias de DOM, los participantes de XML-DEV (liderados por David Megginson) crearon la interfaz SAX. SAX tiene diversas características que solucionan las limitaciones de DOM.

- Un parser SAX envía eventos al código. El parser le dice cuando ha encontrado el comienzo y/o fin del documento, elemento, texto, etc. El programador decide que eventos son importantes y que tipo de estructura de datos desea crear para almacenarlos. Si no salva
- explícitamente los datos de un evento, se perderán.

- Un parser SAX no crea ningún objeto en absoluto, simplemente envía eventos hacía su aplicación. Si desea crear objetos basados en esos eventos, es cosa suya.

- Un parser SAX empieza a enviar eventos tan pronto como se inicia. Su código recibirá un evento cuando el parser encuentre el inicio del documento, cuando encuentre el inicio de un elemento, texto, etc. Su aplicación comenzará a generar resultados inmediatamente, sin tener que esperar hasta que el documento entero haya sido parseado. Aún mejor, si solo está buscando algunas cosas en el documento, su código puede elevar una excepción sólo si ha encontrado lo que estaba buscando. Una excepción detiene el parser SAX y su código entonces puede hacer cualquier cosa que necesite hacer con los datos encontrados.

Limitaciones de SAX

Los parsers SAX tienen limitaciones que pueden causar preocupación:

- Los eventos SAX no tienen estado. Cuando un parser SAX encuentra texto en un documento XML, envía un evento a su código. Este evento solo le indica el texto encontrado; no le dice que elemento contenía ese texto. Si quiere saberlo, el programador tendrá que escribir el código para la gestión de estado usted mismo.

- Los eventos SAX no son permanentes. Si su aplicación necesita una estructura de datos que modele el documento XML, debe escribir ese código usted mismo. Si necesita acceder a los datos de un evento SAX y no los ha almacenado en su código, tiene que parsear el documento de nuevo.

- SAX no está controlado por una organización centralizada. Aunque esto no ha causado problemas hasta hoy, algunos desarrolladores se sentirían más cómodos si SAX estuviese controlado por una organización como la W3C.

JDOM

Frustrados por la dificultad en realizar ciertas tareas con los modelos DOM y SAX, Jasón Hunter y Brett McLaughlin crearon el paquete JDOM. JDOM es un proyecto open-source de tecnología basada en Java que intenta seguir la regla 80/20: Desarrollar lo que el 80% de los usuarios necesitan con el 20% de las funciones en DOM y SAX. JDOM trabaja con parsers SAX y DOM, así que está implementado como un conjunto relativamente pequeño de clases Java.

La principal ventaja de JDOM es que reduce enormemente la cantidad de código que se debe escribir. Aunque este tutorial introductorio no discute tópicos sobre programación con profundidad, el tamaño de las aplicaciones JDOM es, por lo normal, una tercera parte que las de las aplicaciones DOM y aproximadamente la mitad que el de las aplicaciones SAX. (Los puristas de DOM, por supuesto, aseguran que el uso de DOM es una buena disciplina que producirá beneficios a largo plazo.) JDOM no lo hace todo, pero para la mayor parte del parseo que se quiera hacer, es probablemente la elección correcta.

El API Java para parseo de XML.

Aunque DOM, SAX y JDOM proporcionan interfaces estándar para las tareas más comunes, todavía hay varias cosas que no hacen. Por ejemplo, el proceso de creación de un objeto DOMParser en Java difiere de un parser DOM a otro. Para solucionar este problema, Sun ha publicado JAXP, el API Java para parseo de XML. Este API proporciona interfaces comunes para el procesado de documentos XML usando DOM, SAX y XSLT.

JAXP proporciona interfaces tales como DocumentBuilderFactory y DocumentBuilder, los cuales proporcionan a su vez una interfaz estándar a diferentes parsers. Existen también métodos que le permiten controlar si el parser subyacente utiliza namespaces y si usa DTDs o esquemas para validar los documentos XML.

¿Qué interfaz es la correcta?

Para determinar que interfaz de programación es la correcta, se necesita comprender la intención con que se diseñaron y necesita comprender que es lo que la aplicación tiene que hacer con los documentos XML que va a procesar. Considere las siguientes preguntas como una ayuda para encontrar la aproximación adecuada.

- ¿Su aplicación estará escrita en Java? JAXP trabaja con DOM, SAX y JDOM; si esta se escribirá el código en Java, debería usar JAXP para aislar el código de los detalles de implementación de los diversos parsers.

- ¿Cómo será distribuida la aplicación? Si la aplicación va a ser distribuida cómo un applet Java, y se quiere minimizar la cantidad de código descargado, hay que tener en mente que los parser SAX son más pequeños que los parser DOM. También se debe considerar que el uso de JDOM requiere una cantidad de código menor que los parser SAX o DOM.

- ¿Una vez que se ha interpretado el documento usando parser, es necesario acceder a esos datos varias veces? Si se necesita volver sobre la versión parseada del documento XML, DOM es probablemente la elección correcta. Cuando un evento SAX se ha disparado es responsabilidad del desarrollador, almacenarlo si lo va a necesitar más tarde. Si se necesita acceder a un evento que no se salvo, hay que parsear el archivo de nuevo. DOM almacena todos los datos automáticamente.

- ¿Se Necesitan solo unas pocas cosas del documento XML? Si solo se va a necesitar unas pocas cosas del documento XML, SAX es probablemente la elección correcta. SAX no crea objetos para todo lo que contiene el documento fuente; se puede decidir que es lo importante. Con SAX se puede comprobar cada evento par determinar si es relevante para sus necesidades y procesarlo entonces adecuadamente. Aún mejor, una vez que ha encontrado lo que andaba buscando, su código puede elevar una excepción para parar definitivamente el parser SAX.

- ¿Trabajará en una máquina con muy poca memoria? Si es así, SAX es la mejor elección, independientemente de todos los otros factores que pueda considerar. Hay que tener en cuenta que existen APIs XML para otros lenguajes; Las comunidades Perl y Phyton en particular tienen muy buenas herramientas XML.

12.9 Estándares XML

Existe una gran variedad de estándares en el universo XML. Además del estándar base XML, otros estándares definen esquemas, hojas de estilo, enlaces, web services, seguridad y otros importantes items. A continuación se muestran los estándares más populares, y le indica referencias para encontrar otros estándares.

LA ESPECIFICACIÓN XML

Esta especificación, localizada en w3.org/tr/rec-xml, define las reglas básicas para los documentos XML. Todas las reglas para documentos XML definidas anteriormente están definidas aquí. Además del estándar básico XML, la especificación de Namespaces es otra parte importante de XML. Puede encontrar el estándar para los namespaces en: w3.org/TR/REC-xml-nombres/.

ESQUEMA XML

El lenguaje de Esquemas XML está definido en tres partes:

- Un primario, localizado en w3.org/TR/xmlschema-0, que proporciona una introducción a los documentos de esquema XML y al para qué están diseñados.
- Un estándar para los documentos de estructura, localizados en w3.org/TR/xmlschema-1, que ilustra cómo se define la estructura de los documentos XML.
- Un estándar para los tipos de datos, localizada en w3.org/TR/xmlschema-2, que define algunos tipos de datos comunes y las reglas para crear otros nuevos.

XSL, XSLT, Y XPATH

El Lenguaje Extensible de Hojas de Estilo, XSL (Extensible Stylesheet Language), define un conjunto de elementos (llamados elementos de formato) que describen como los datos deben ser formateados. Por claridad, este estándar aún se designa como XSL-FO para distinguirlo de XSLT.

Aunque inicialmente fue diseñado para generar documentos para impresión de gran calidad, también se pueden usar los objetos de formato para generar archivos de audio desde XML. El estándar XSL-FO se encuentra en w3.org/TR/xsl/.

El Lenguaje Extensible de Hojas de Estilo para Transformaciones, XSLT (extensible Stylesheet Language for Transformations), es un vocabulario XML que describe cómo convertir un documento XML en algo más. El estándar esta en w3.org/TR/xslt XPath, el lenguaje de rutas XML es una sintaxis que describe localizaciones en los documentos XML. XPath se usa en las hojas de estilo XSLT para describir qué partes de un documento XML se quiere transformar. XPath también es usado in otros estándares XML, siendo este el porqué es un estándar separado de XSLT. XPath está definido en w3.org/TR/xpath.

DOM

El Document Object Model define cómo se convierte un documento XML en una estructura de árbol en memoria. DOM está definido en un varias especificaciones en la W3C:

El núcleo (core) DOM define el DOM mismo, la estructura de árbol y los tipos de nodos y

excepciones que su código puede encontrar cuando se mueve a través del árbol. La especificación completa se encuentra en w3.org/TR/DOM-Level-2-Core/.

Eventos, define los eventos que pueden sucederle al árbol como esos eventos son procesados. Esta especificación es un intento de reconciliar las diferencias en los modelos de objetos soportados por Netscape e Internet Explorer desde la versión 4 de estos navegadores. Esta especificación esta en w3.org/TR/DOM-Level-2-Events/.

Estilo define cómo las hojas de estilo XSLT y las hojas de estilo CSS pueden ser accedidas por un programa. Esta especificación esta en w3.org/TR/DOM-Level-2-Style/. Traversaciones y Rangos, define interfaces que permiten a los programas traversar el árbol o definir un rango de nodos en el árbol. Puede encontrar la especificación completa en w3.org/TR/DOM-Level-2-Traversal-Range/ .

Vistas, define una interfaz AbstractView para el documento mismo. Puede encontrar más información en w3.org/TR/DOM-Level-2-Views/.

SAX, JDOM, Y JAXP

El Simple API for XML define los eventos e interfaces usados para interactuar con un programa parser XML que cumpla con SAX. Puede encontrar la especificación completa de SAX en www.saxproject.org.

El proyecto JDOM fue creado por Jason Hunter y Brett McLaughlin y se encuentra en jdom.org/. En el sitio de JDOM puede encontrar código, programas de ejemplo, y otras herramientas para empezar.

Un punto significativo acerca de SAX y JDOM es que ambos han salido de la comunidad de desarrolladores de XML, no de un cuerpo de estándares. Su amplia aceptación es un tributo a la participación de los desarrolladores XML en el mundo entero. Puede encontrar todo lo que hay que saber acerca de JAX en java.sun.com/xml/jaxp/.

ENLAZADO Y REFERENCIA

Hay dos estándares para el enlazado y referencia en el mundo XML: XLink y XPointer:

- XLink, el lenguaje XML de enlazado, define muchas maneras de enlazar diferentes recursos juntos. Puede hacer enlaces normales punto-a-punto (como con el elemento XML <a>) o enlaces extendidos, que pueden incluir enlaces multipunto, enlaces a través de terceros y reglas que definen lo que significa seguir un enlace determinado. El estándar XLink se encuentra en w3.org/TR/XLink/.

- XPointer¸ el lenguaje de apuntadores XML, usa XPath como una manera de referenciar otros recursos. También incluye algunas extensiones a XPath. Puede encontrar su especificación en www.w3.org/TR/xptr/.

SEGURIDAD

Hay dos estándares significativos que determinan la seguridad de los documentos XML. Uno es el estándar XML Digital Signature o Firma Digital XML (w3.org/TR/xmldsig-core/), que define una estructura de documento XML para las firmas digitales. Puede crear una firma digital XML para cada tipo de datos, tanto si es un documento XML, un archivo HTML, texto plano, datos binarios, etc. La firma digital es usada para verificar que un archivo concreto no ha sido modificado después de que fuese firmado. Si el dato que ha firmado es un documento XML, puede incrustar el documento XML en el archivo de firma, lo que hace que el procesado de los datos y de la firma sea muy simple.

Los otros estándares determinan el encriptado de documentos XML. Aunque es bueno que los documentos XML sean escritos de tal manera que un humano pueda leerlos y entenderlos, esto puede significar un problema si el documento cae en las manos equivocadas. El estándar XML Encryption o Encriptación XML (w3.org/TR/xmlenc-core/) define como partes de un documento XML pueden ser encriptadas.

Usando estos estándares juntos, se pueden usar documentos XML con confianza. Podría firmar digitalmente un importante documento XML, generando una firma que incluya el mismo documento. Al realizar la encriptación del documento (usando clave privada y clave pública) y enviárselo a un destinatario. Cuando este lo reciba, puede desencriptarlo con su clave privada y clave pública, lo que le permitirá saber que he sido la única persona que le ha enviado el documento. Una vez desencriptado el documento, la firma digital es usada para asegurarse de que no ha sido modificado en ninguna forma.

OTROS ESTÁNDARES

Existe un gran número de otros estándares XML que no se han detallado. Además de los estándares de aplicación general como Scalable Vector Graphics (www.w3.org/TR/SVG/) Vector Graphics y SMIL, el Syncronized Multimedia Integration Language (www.w3.org/TR/smil20 /), hay muchos estándares específicos en la industria. Por ejemplo, el Consorcio HR-XML ha definido una gran cantidad de estándares XML para Recursos Humanos; puede encontrarlos en html.org Finalmente, para una buena fuente de estándares XML puede visitar el Repositorio XML en xml.org/xml/registry.jsp. Este sitio proporciona cientos de estándares para una gran variedad de industrias.

CONSEJOS Y RECURSOS

- Decidir que datos se desea convertir en XML. Típicamente estos son los datos que se necesitan mover entre sistemas o que deben ser transformados en varios formatos.

- Comprobar si ya existe algún estándar XML. Si esta usando datos muy comunes cómo ordenes de compra, registros médicos, o reservas en almacén, tiene muchas posibilidades de que alguien ya haya definido un estándar XML para esos datos.

- Comprobar si las herramientas actuales soportan XML. Si está usando una versión reciente de un paquete de base de datos, hoja de cálculo o algún otro tipo de herramienta para la administración o manejo de datos, es probable que estas herramientas (o alguna actualización) pueden usar XML como formato para la entrada o salida de datos.

- Aprender cómo construir aplicaciones basadas en XML. Es necesario comprender como están almacenados los datos actualmente, como necesitan ser transformados y cómo integrar sus esfuerzos de desarrollo XML con la aplicaciones existentes. La columna de Benoît Marchall's Working XML es un estupendo lugar donde empezar; puede encontrar un listado actualizado de todas sus columnas en http://www-106.ibm.com/developerworks/xml/library/x-wxxmcol/.

- Únase al grupo de estándares apropiado. Considere unirse a grupos como el World-Wide Web Consortium (W3C) , así como a grupos de industrias específicas cómo HRXML.org.

- Ser un miembro de esos grupos ayudará a estar informado sobre lo que ocurren en la industria, y le da la oportunidad de modelar el futuro de los estándares XML.

- Evite soluciones propietarias. Es importante que use sólo tecnología basada en estándares en sus esfuerzos de desarrollo; resista los señuelos de los vendedores que le ofrecen "mejoras". Una de las ventajas de XML es que puede tener control completo sobre los datos. Una vez que ha sido tomados como rehenes por un formato de datos propietario, se padece una tremenda perdida del control.

RECURSOS

Se presentan algunos recursos para ayudar a empezar:

The dW XML zone es un buen lugar para ver recursos XML.
Ver www106.ibm.com/developerworks/xml para todo lo que quiera saber sobre XML.
XML tools: developerWorks tiene "Fill your XML toolbox" con artículos que describen herramientas de programación XML para varios lenguajes:

- C/C++: Ver el artículo de Rick Parrish en www106.ibm.com/developerworks/library/xctlbx. html (developerWorks, Septiembre 2001).

- Java: Ver el artículo de Doug Tidwell en www-106.ibm.com/developerworks/library/java-xml-toolkit/index.html (developerWorks, Mayo 2000).

- Perl: Ver el artículo de Tony Darugar en www-106.ibm.com/developerworks/library/perl-xml-toolkit/index.html (developerWorks, Junio 2001).

- PHP: Ver el artículo de Craig Knudsen en www-106.ibm.com/developerworks/library/php-xml-toolkit.html (developerWorks, Junio 2000).

Además de estos artículos puede ver el examen que David Mertz hace a las herramientas XML para Phyton en su artículo "Charming Python: Revisiting XML tools for Python" en www-106.ibm.com/developerworks/library/l-pxml.html .

Tutoriales XML: sobre XML están disponibles en developerWorks; puede ver http://www-105.ibm.com/developerworks/education.nsf/dw/xml-onlinecourse-bytitle para la lista más actualizada.

IBM' s jStart team: El equipo de jStart trabaja a muy bajo precio para ayudar a los clientes a construir soluciones usando las nuevas tecnologías (web services XML, por ejemplo). A cambio, esos clientes le permiten a IBM publicar sus proyectos cómo casos de estudio. Para más información, ver ibm.com/software/jstart.

Estándares XML: Se presenta una lista por orden alfabético de los estándares mencionados.

- DOM, el Document Object Model:
- Core Specification: w3.org/TR/DOM-Level-2-Core/
- Events Specification: w3.org/TR/DOM-Level-2-Events/
- Style Specification: w3.org/TR/DOM-Level-2-Style/
- Traversal and Range Specification:w3.org/TR/DOM-Level-2-Traversal-Range/.
- Views Specification: w3.org/TR/DOM-Level-2-Views/
- HR-XML.org, Human Resources XML Consortium: hr-xml.org
- JAXP, Java API for XML: java.sun.com/xml/jaxp/
- SAX, Simple API for XML: saxproject.org/
- SMIL, Synchronized Multimedia Integration Language: www.w3.org/TR/smil20/
- SOAP, Simple Object Access Protocol, w3.org/TR/SOAP/
- SVG, Scalable Vector Graphics: www.w3.org/TR/SVG/
- UDDI, Universal Description, Discovery, and Integration Protocol: uddi.org
- WSDL, Web Services Description Language: w3.org/TR/wsdl

- XLink, XML Linking Language: w3.org/TR/XLink/
- XML, el estándar que lo comenzó todo: w3.org/TR/REC-xml
- XML Digital Signatures: w3.org/TR/xmldsig-core/
- XML Encryption: w3.org/TR/xmlenc-core/
- XML Namespaces: w3.org/TR/REC-xml-nombres/
- XML Repository de DTDs y esquemas: xml.org/xml/registry.jsp
- XML Schema:
- Part 0 - Primer: w3.org/TR/xmlschema-0
- Part 1 - Document structures: w3.org/TR/xmlschema-1
- Part 2 - Datatypes: w3.org/TR/xmlschema-2
- XPath, el lenguaje XML Path: w3.org/TR/xpath
- XPointer, el lenguaje de punteros para XML: www.w3.org/TR/xptr/
- XSL-FO, Extensible Stylesheet Language for Formatting Objects: w3.org/TR/xsl/
- XSLT, Extensible Stylesheet Language: w3.org/TR/xslt (sin barra de cierre)
- Para saber más sobre JDOM, puede leer los siguientes artículos de developerWorks:
- Simplify XML programming with JDOM at
- www-106.ibm.com/developerworks/java/library/j-jdom/ (developerWorks, Mayo 2001)
- Converting from DOM en
- www-106.ibm.com/developerworks/java/library/x-tipcdm.html (developerWorks, Abril 2001)
- Converting from SAX en
- www-106.ibm.com/developerworks/java/library/x-tipcsx.html (developerWorks, Abril 2001)
- Using JDOM and XSLT en www-106.ibm.com/developerworks/java/library/x-tipjdom.html (developerWorks, Marzo 2001)

13. MICROSOFT .NET FRAMEWORK

Microsoft .NET Framework es una plataforma para desarrollar, implementar y ejecutar los servicios y aplicaciones Web XML. Proporciona un ambiente de productividad, basado en estándares y multilingüe para integrar las inversiones existentes con las aplicaciones y servicios de la siguiente generación, así como agilidad para resolver los retos de implementación y operación de las aplicaciones a escala de Internet.

Para la información más reciente sobre .NET Framework, visite:
http://msdn.microsoft.com/net.

.NET Framework es el resultado de dos proyectos. La meta del primer proyecto era mejorar el desarrollo de Microsoft Windows®, tratando específicamente de mejorar el Modelo de objeto de componente (COM) de Microsoft. El segundo proyecto buscaba crear una plataforma para integrar el software como un servicio. Estos dos proyectos se unieron hace más de tres años. El producto terminado mejora de manera importante la productividad del programador, facilidad de implementación y la ejecución confiable de las aplicaciones además introduce un concepto totalmente nuevo para la computación: los web services XML, aplicaciones acopladas libremente y componentes diseñados para la computación heterogénea actual al comunicarse utilizando protocolos y estándares de Internet, como SOAP y XML.

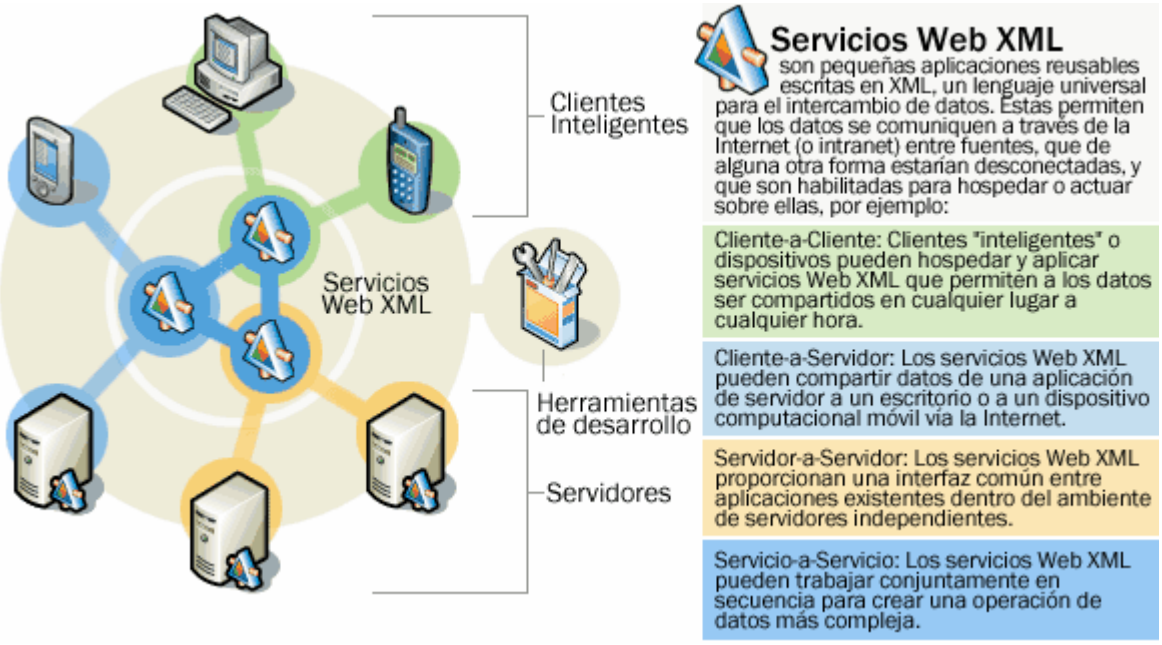


Figura 20. Servicios Web XML.

Microsoft .NET es un conjunto de tecnologías de software de Microsoft para conectar el mundo de información, gente, sistemas y dispositivos. Permite un nivel sin precedente de integración de software a través del uso de web services XML: pequeños, discretos, bloques de aplicaciones construidos que se conectan con cada uno, así como a otras aplicaciones grandes vía Internet.

13.1 Componentes del software conectado de Microsoft .NET.

NET está infundido en los productos que conforman la plataforma Microsoft, proporcionando la habilidad para construir, hospedar e implementar de una manera rápida y confiable, y utilizar soluciones seguras y conectadas usando los web services XML. La plataforma Microsoft provee una suite de herramientas de desarrollo, aplicaciones cliente, web services XML, y servidores necesarios para intervenir en este mundo conectado.

El software de aplicaciones para clientes "inteligentes" (móviles) y sistemas operativos permite a las PCs y otros dispositivos inteligentes interactuar con los servicios Web XML, accesando a la información en cualquier lugar y en cualquier momento. Para mas información consulte: Software del cliente inteligente y .NET http://www.microsoft.com/latam/net/products/client.asp. Y Dispositivos inteligentes y .NET http://www.microsoft.com/latam/net/products/devices.asp

Microsoft y otros están desarrollando un conjunto principal de web services XML, desde autenticación hasta calendarización que puede ser combinado con otros web services XML o usado directamente con aplicaciones del cliente inteligente. Microsoft MapPoint .NET, un web service XML que permite integrar mapas de alta calidad, indicaciones viales, y otras locaciones inteligentes en sus aplicaciones, procesos de negocio, y sitios Web, es un ejemplo de uno de estos servicios. Si desea obtener mas información puede consultar: ¿Qué son los web services XML? http://www.microsoft.com/latam/net/basics/xmlservices.asp y Microsoft MapPoint .NET http://www.microsoft.com/mappoint/net/

Microsoft provee una infraestructura en servidores incluyendo la familia de servidores Microsoft Windows® 2000 y los Servidores .NET Enterprise para implementar, manejar e instrumentar los web services XML. Para mas detalles consulte: Servidores Microsoft y .NET http://www.microsoft.com/latam/net/products/servers.asp

Microsoft Visual Studio® .NET y el Microsoft .NET Framework brindan solución para que los desarrolladores puedan construir, implementar y ejecutar los web services XML. Para mas información consulte: Herramientas para desarrolladores y .NET y http://www.microsoft.com/latam/net/products/tools.asp

Software de cliente inteligente y .NET

Tanto si utiliza un equipo de escritorio como un dispositivo inalámbrico en el exterior, los usuarios deben poder obtener acceso a la información necesaria y utilizarla cuando la necesiten. Esta experiencia informática libre de problemas se está haciendo realidad con el uso, por parte de los programadores, de la plataforma Microsoft® .NET para crear aplicaciones con una gran variedad de características que se integran con Windows e Internet. Microsoft .NET ofrece la experiencia por medio del software de cliente inteligente.

Una nueva generación de aplicaciones conectadas (clientes inteligentes) posibilitará el funcionamiento de equipos y dispositivos móviles e incrustados y permitirá una experiencia de usuario más personalizada en la plataforma .NET.

Microsoft Windows® XP, Windows XP Embedded y el futuro Windows CE .NET, Microsoft está creando esta generación de clientes de software que permiten la conexión en cualquier lugar y en cualquier momento.

El entorno de ejecución seguro y administrado de .NET Framework permite a los programadores crear aplicaciones para el software de cliente en una variedad de dispositivos inteligentes.

Software de cliente inteligente para PC

MICROSOFT WINDOWS XP

Windows XP es el nuevo sistema operativo de Windows para el hogar y la empresa. Por su integración con el uso de XML que las versiones anteriores de Windows, Windows XP ofrece un motor inteligente, conectado y seguro que permite las experiencias en la plataforma .NET tanto para usuarios de equipos portátiles como de escritorio.

El PC seguirá siendo el núcleo de muchas de las experiencias .NET, desde la modificación de fotografías digitales a la creación de hojas de cálculo complejas. El soporte mejorado para equipos móviles integrado en Windows XP facilita el paso de una red a otra y la conexión remota, con lo que el equipo se convierte en un centro de actividad de la plataforma .NET.

MICROSOFT WINDOWS CE .NET

Windows CE .NET, disponible en la versión Beta 2 en la actualidad, es el nombre del sucesor de Windows CE 3.0. Creado para ofrecer un sistema operativo estable en tiempo real para dispositivos de pequeño tamaño, como los asistentes digitales personales de mano, Windows CE .NET permitirá a los programadores comercializar dispositivos inteligentes conectados que utilicen la tecnología inalámbrica y multimedia más avanzada. Windows CE .NET actuará de plataforma operativa para muchos dispositivos inteligentes y ampliará así el alcance de la plataforma .NET.

Microsoft Windows XP EmbeddedT

Windows XP Embedded, la versión de componentes del sistema operativo de escritorio Windows, ofrece una plataforma para una gama de dispositivos de cliente inteligentes como los dispositivos multimedia interactivos, terminales de servicios y clientes ligeros. Windows XP Embedded permite a los programadores y a los fabricantes de dispositivos sacar partido a las numerosas características nuevas de Windows XP y comercializar dispositivos inteligentes con una gran variedad de características basados en microprocesadores x86.

Dispositivos inteligentes y .NET

Entre los dispositivos inteligentes figuran los PC, equipos portátiles, estaciones de trabajo, teléfonos, PC de mano, Tablet PC, consolas de juegos Microsoft® XBox™ y equipos que por ahora sólo imaginamos. Estos dispositivos son "inteligentes" porque son capaces de obtener acceso a web services XML y le permiten interactuar con sus datos independientemente de la ubicación, el tipo y el número de dispositivos que utilice.

Los dispositivos inteligentes permiten obtener acceso a su información en el formato adecuado en cualquier momento y lugar. Los dispositivos inteligentes utilizan los web services XML y ofrecen experiencias en .NET. Todos los dispositivos inteligentes optimizan la presentación y recopilación de la información, desde la conversión de texto a voz al reconocimiento de la escritura manual.

Dispositivos inteligentes basados en tecnologías de Microsoft Pocket PC.

Los dispositivos basados en Pocket PC, comercializados desde el otoño del 2001, ofrecen una interfaz nueva y elegante, características nuevas para usuarios empresariales y compañías, y formas nuevas de comunicarse con amigos y compañeros de trabajo.

Plataforma Microsoft Smartphone.
La plataforma Microsoft Smartphone amplía las funciones de los teléfonos móviles, permite la ejecución de varias aplicaciones en línea y sin conexión para mantenerle conectado. Microsoft y Sendo presentaron un prototipo de la plataforma Microsoft Smartphone en febrero del 2001.

Microsoft Xbox
Xbox, la consola de juegos de nueva generación de Microsoft, se comercializó en América del norte en noviembre de 2001 (la comercialización en Europa estaba prevista para marzo de 2001).

Tablet PC
Tablet PC, incorpora las funciones de escritura manual y voz en un dispositivo de PC móvil y versátil, se comercializa desde el segundo semestre de 2002. Tablet PC ejecuta una versión especializada del sistema operativo Microsoft Windows® XP llamada Windows XP Tablet PC.

Ventajas de los dispositivos inteligentes
Ofrecen una experiencia más personal y sencilla porque los dispositivos inteligentes conocen sus preferencias. Utilizan su identidad, información de perfil y datos de .NET para personalizar la experiencia.

Permiten interactuar en línea, sin conexión y pasar de un modo a otro sin problemas porque los dispositivos inteligentes conocen la red y responden a las limitaciones del ancho de banda

13.2 Los Servidores de Microsoft y .NET

Con Microsoft® .NET, los negocios pueden cambiar los procesos clave en web services XML. Ya sea el itinerario de la oficina de un doctor, el catálogo para un vendedor de libros, una calculadora de intercambio de promedios de divisas a nivel mundial, - los web services XML creados y alojados por una compañía o individuo pueden ser usados por millones de personas en varias combinaciones, para generar experiencias computacionales altamente personales e inteligentes.

Este modelo distribuido de cómputo incrementa las demandas en las infraestructuras de servidores:

- Servidores seguros y escalables, que integren XML, proporcionan la base para alojar e implementar software y servicios .NET.

- Microsoft Windows® 2000 Server, la próxima familia de Windows Server 2003, y Microsoft .NET Enterprise Servers son los servidores Microsoft que cumplen con esta función.

Infraestructura de Servidor

Microsoft .NET Enterprise Servers, la familia de Windows 2000 Server y la familia de Windows Server 2003, proporcionan una solución para alojar e implementar web services, con seguridad integrada, su soporte de XML y su capacidad para escalar rápidamente y enfrentar demandas en aumento.

Servidores .NET Enterprise

La familia de .NET Enterprise Server acelera la integración de los sistemas, aplicaciones y socios usando web services, a través del soporte que tienen los servidores para XML. Este soporte permite que las empresas desarrollen en sistemas heredados en lugar de desbaratarlos y reemplazarlos. Por ejemplo, Microsoft Host Integration Server proporciona un acceso sencillo a mainframes y Microsoft BizTalk™ ofrece conversiones automáticas de formatos de datos existentes a XML. La familia de .NET Enterprise Server incluye:

- Microsoft Application Center 2000 para implementar y administrar aplicaciones Web altamente disponibles y escalables.
- Microsoft BizTalk Server 2000 para desarrollar procesos de negocios en XML a través de aplicaciones y organizaciones.
- Microsoft Commerce Server 2000 para desarrollar de comercio electrónico.
- Microsoft Content Management Server 2001 para administrar el contenido de sitios dinámicos Web de negocios electrónicos.
- Microsoft Exchange Server 2000 para habilitar la mensajería y colaboración a cualquier hora y en cualquier lugar.
- Microsoft Host Integration Server 2000 para pasar datos y aplicaciones a sistemas heredados de mainframe.
- Microsoft Internet Security and Acceleration Server 2000 para una conectividad de Internet segura y rápida.
- Microsoft Mobile Information 2001 Server para habilitar el soporte de aplicaciones por dispositivos móviles como teléfonos celulares.
- Microsoft SharePoint™ Portal Server 2001 para encontrar, compartir y publicar información del negocio.
- Microsoft SQL Server™ 2000 para almacenar, recuperar y analizar datos estructurados de XML.

13.3 Las Herramientas del Desarrollador y .NET

Con Microsoft® Visual Studio® .NET y .NET Framework, los desarrolladores pueden crear web services e integrarlos a otras aplicaciones. La mayoría de los desarrolladores pueden aprovechar habilidades existentes, debido a que el tiempo de ejecución del lenguaje común del .NET Framework le permite desarrollar web services usando cualquier lenguaje de programación moderno.

Microsoft Visual Studio .NET y Microsoft .NET Framework proveen una solución para que los desarrolladores puedan construir, implementar y ejecutar web services. Estas herramientas incrementan el rendimiento, la confiabilidad y la seguridad de los web services.

Microsoft Visual Studio .NET

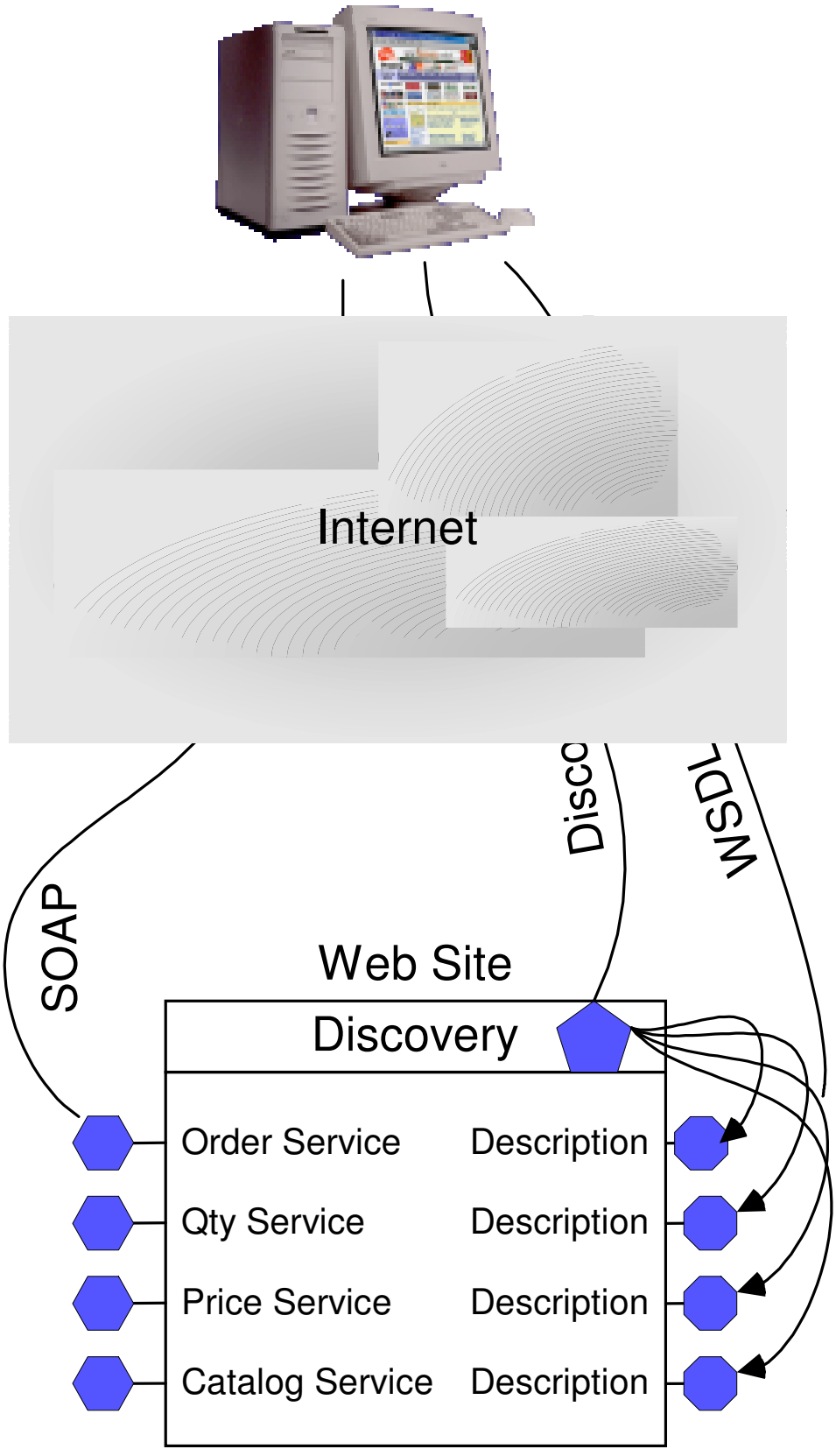
Para crear web services los desarrolladores pueden usar una variedad de ambientes de programación. Microsoft Visual Studio .NET representa un buen ambiente de desarrollo para software y servicios .NET.

Visual Studio .NET se adelanta a los lenguajes de programación de alta productividad: Microsoft Visual Basic®, que incluye nuevas funcionalidades de programación orientada a objetos; Microsoft Visual C++®, que se adelanta al desarrollo de Windows® y le permite desarrollar aplicaciones .NET; y C#, que brinda RAD a el desarrollador de C y C++.

Programa en el Lenguaje Apropiado para la Tarea

Visual Studio .NET proporciona un ambiente unificado de desarrollo. Desarrollado en .NET Framework, Visual Studio proporciona soporte para trabajar con web services XML creados en lenguajes de programación modernos.

Las aplicaciones y web services XML creados en un lenguaje pueden ser programados y depurados en cualquier otro lenguaje soportado por Visual Studio .NET. Esto mejora enormemente la capacidad de usar web services XML existentes para desarrollar nuevas soluciones.


Internet
SOAP
Disco
WSDL
Web Site
Discovery
Order Service
Description
Qty Service
Description
Price Service
Description
Catalog Service
Description

Transformar Aplicaciones en Web Services XML

Visual Studio .NET crea automáticamente la interfaz necesaria de XML y SOAP indispensable para cambiar una aplicación en un web service XML. Los desarrolladores se pueden concentrar en el desarrollo de la aplicación, y no en la estructura interna del web service XML.

Reusar Web Services XML

El desarrollo con web services XML es similar al desarrollo con componentes. Visual Studio .NET le da a los desarrolladores la facilidad de importar web services XML o usar web services XML alojados remotamente y programarlos como lo harían con un elemento COM hoy en día, ahorrando tiempo y dándole a los desarrolladores la oportunidad de concentrarse en funcionalidades básicas.

Microsoft .NET Framework y Microsoft .NET Compact Framework

NET Framework y el dispositivo enfocado a .NET Compact Framework, son ambientes de ejecución de aplicaciones, basados en estándares y multi-lenguajes que manejan tareas esenciales de plumbing y facilitan el desarrollo. El ambiente de ejecución de la aplicación administra la memoria, trata problemas de versiones y mejora la confiabilidad, escalabilidad y seguridad de las aplicaciones. Los componentes incluyen tiempo de ejecución de lenguaje común, un grupo de librerías de clases para desarrollar web services XML y ASP.NET.

El tiempo de ejecución de lenguaje común es el motor en .NET Framework que proporciona un ambiente de ejecución administrado y seguro, y es diseñado para dar soporte a los desarrolladores que usan muchos lenguajes de programación al crear sus aplicaciones. Tiene un sistema unificado y habilita la herencia de lenguaje cruzado y la depuración. Al usar .NET Framework, los desarrolladores tienen el medio más rápido y más productivo para desarrollar aplicaciones que realmente son servicios y aplicaciones Web XML de la tercera generación.

Los beneficios de .NET Framework y Visual Studio .NET son:

- Crear rápida y fácilmente web services XML que escalen y se integren fácilmente.
- Mejorar la confiabilidad, escalabilidad y seguridad de aplicaciones al usar el ambiente de ejecución de la aplicación de .NET Framework.
- Promover el talento del desarrollador a través de .NET Framework y soporte de Visual Studio .NET para los lenguajes de programación más modernos.
- Habilitar el uso de habilidades existentes para crear soluciones para un amplio rango de dispositivos.

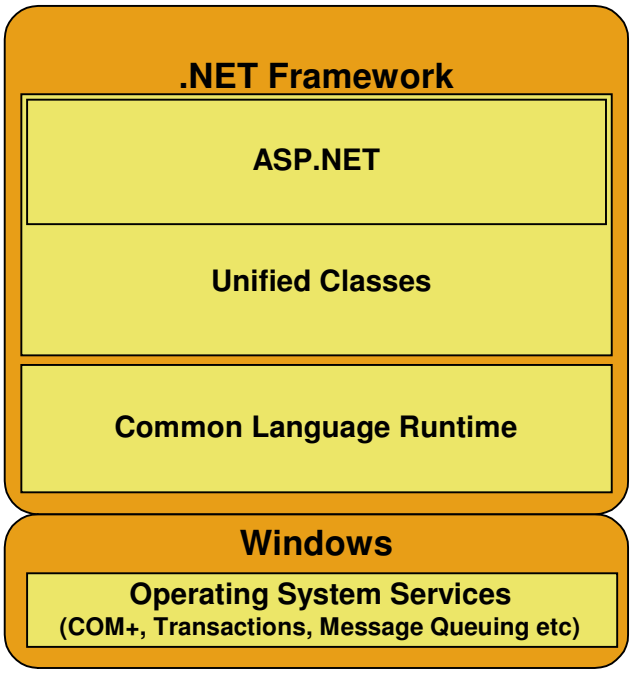


Figura 21. NET Framework en tres partes.

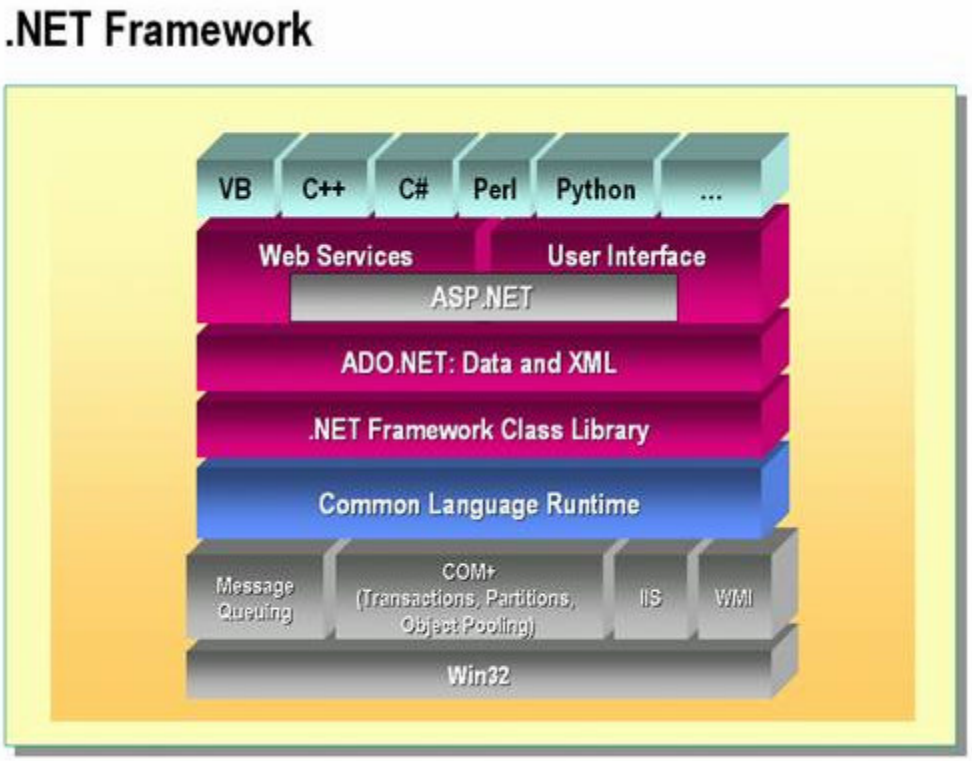


Figura 22. .NET Framework

La meta de Microsoft .NET Framework es facilitar el desarrollo de los servicios y aplicaciones Web XML, pero también tiene un efecto importante en cualquier tipo de aplicación, desde aplicaciones cliente, hasta varios tipos de aplicaciones distribuidas.

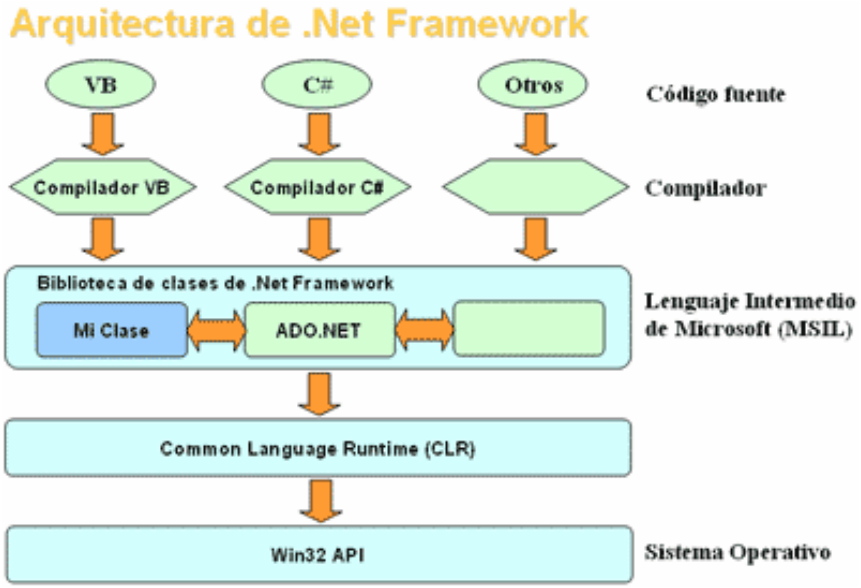


Figura 23.Arquitectura de .Net Framework.

.NET Framework consiste en tres partes principales: el tiempo de ejecución de lenguaje común, un conjunto jerárquico de bibliotecas de clase unificada y una versión basada en componentes de Microsoft Active Server Pages, llamado Microsoft ASP.NET.

El tiempo de ejecución de lenguaje común se desarrolla sobre los servicios del sistema operativo. Es responsable de ejecutar realmente la aplicación, asegurando que se cumplan todas las dependencias de la misma, administrando la memoria, manejando la seguridad, integrando el lenguaje, etc. El tiempo de ejecución proporciona varios servicios que ayudan a simplificar el desarrollo de códigos y la implementación de aplicaciones, lo cual mejora la confiabilidad de la aplicación.

Sin embargo, el desarrollador no interactúa realmente con el tiempo de ejecución. Los desarrolladores utilizan un conjunto unificado de clases desarrolladas sobre el tiempo de ejecución. Estas clases se pueden utilizar desde cualquier lenguaje de programación.

Como parte de estas bibliotecas de clase, .NET Framework incluye un modelo de programación de aplicaciones Web llamado ASP.NET, el cual proporciona componentes y servicios de un nivel mayor que tienen el objetivo específico de desarrollar servicios y aplicaciones Web XML.

13.4 De COM al CLR

Uno de los principios fundamentales de la tecnología de componentes es la definición de contratos que se llevan a cabo entre programas. El modelo COM fue un intento de Microsoft de formalizar estos contratos bajo una tecnología. Estos contratos en términos de diseño eran expresados como definiciones de tipo (comúnmente como Interfaces, o librerías de tipos). Esto fue un avance a las definiciones basándose en métodos o funciones como

puntos de entrada a un programa (a que método llamar). COM trajo la carga dinámica del código en memoria y un sistema basado en tipos de una forma consistente.

COM puede ser visto de dos formas: como un modelo de programación y como una plataforma tecnológica. En el segundo punto COM no ha sido tan sólido como en el modelo de programación que soporta. Por ello se necesita una plataforma tecnológica sólida para lograr que COM sea más que solo una idea o disciplina de programación.

Casi todos los problemas de la plataforma COM tienen que ver con la naturaleza de los contratos entre los componentes (interfaces, librerías de tipo...). En un mundo ideal, los contratos entre los componentes pueden ser expresados en términos de la semántica y las condiciones que el cliente y el componente asumen que existen. Pero aún falta definir una forma de expresar la semántica que pruebe ser comercialmente (o técnicamente) viable para desarrollos grandes y a gran escala.

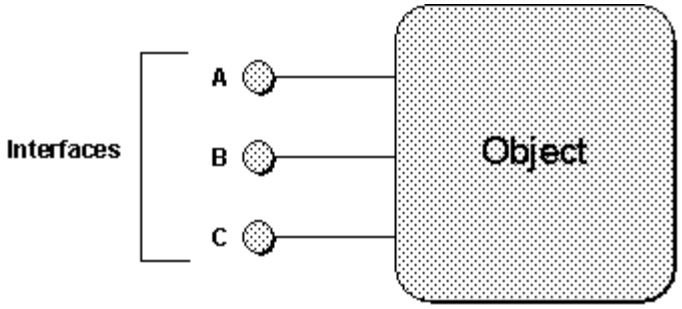


Figura 24. Interfaces

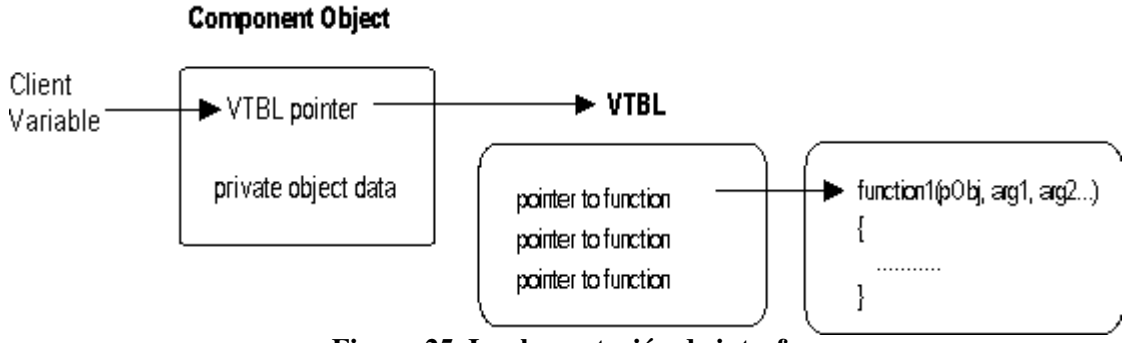


Figura 25. Implementación de interfaces

COM expresa los contratos entre los componentes en términos de tipos. Sin embargo, un contrato de un componente en COM tiene dos problemas claves que hacen que los contratos COM no sean óptimos para expresar la semántica. Uno estos problemas está relacionado a la descripción de un contrato COM. El otro problema está relacionado al contrato mismo.

El primer problema que trata de cómo se describen los contratos en COM. La especificación COM retrocede al evitar imponer un formato de intercambio para las definiciones de contrato. Esto significa que sí uno se basa totalmente en la especificación COM, no hay una forma estándar de describir un contrato; más aún la especificación COM

asume que las definiciones de tipo de un contrato debería ser comunicadas vía otra técnica que estaba fuera del alcance de COM. Por supuesto esto es solamente viable en el mundo de las especificaciones. Para hacer que COM sea una tecnología útil, se necesita una solución concreta, de otra forma es imposible desarrollar compiladores, herramientas, y la infraestructura que las soporte.

Microsoft definió y soportó dos formatos de intercambio para las descripciones de un contrato COM. Esto es realmente un problema porque habían construcciones que podían ser expresados en un formato y que no tenían un significado o representación en el otro formato. Peor era que, las construcciones soportadas por un formato no eran un subconjunto de las construcciones soportadas por el otro. Esto hacía imposible ver alguno de los dos formatos como "completo" o "normativo" para las descripciones de contrato.

Un argumento fue hecho para definir un tercer formato basado en la unión de las construcciones soportadas por ambos formatos. Sin embargo, había dos problemas más críticos con la forma en la que se definían los contratos. Uno era que no se describían las dependencias de un componente. La falta de dependencia provocaba o hacía difícil determinar que DLLs se necesitarían para instalar una aplicación basada en COM. Esto hacía además imposible determinar en forma exacta las versiones que se necesitaban de un componente, lo cual provocaba que el diagnóstico de los problemas sobre las versiones sea extremadamente difícil.

El segundo mayor problema para un formato de descripción de un contrato COM fue su falta de extensibilidad. A comienzos de 1990, el equipo de trabajo sobre el Microsoft Transaction Server (MTS) estuvo trabajando en un nuevo modelo de programación basado en las ideas que ahora se conocen como Programación orientada a aspectos (AOP). AOP toma aspectos del código que no son específicos a un dominio y las saca del código fuente.

Los sistemas basados en AOP usaban mecanismos alternativos para declarar estos aspectos para lograr que el programador sea explícito al definir estos aspectos y no implícito.

El equipo de MTS quería permitir a los desarrolladores expresar sus requerimientos de concurrencia, transacciones, y seguridad como aspectos (atributos) más que llamadas a funciones API.

Debido al amplio uso de COM, los desarrolladores de MTS tenían argumentos de que las descripciones de que los contratos COM fueran el mecanismo para expresar estos aspectos. Los desarrolladores que usan MTS anotan las definiciones de clase, interfaces, y métodos con los atributos que informan al MTS Executive de los requerimientos del código. Para hacer estos atributos útiles, el MTS Executive agrega interceptores basado en los aspectos de la clase a ser cargada. El MTS Interceptor (llamado un context wrapper) hace lo que sea necesario para asegurarse de que los requerimientos y presunciones de los desarrolladores en el código se cumplan al momento de realizar la llamada al método.

Interface Definition Language (IDL), es un formato basado en texto que rara vez se distribuía con el componente. Más aún este formato solo lo usaban los programadores de C++. El segundo formato, las librerías de tipo (TLB), era muy rudimentario, con errores y no extensible. Lo otro era que los programadores de Visual Basic no tenían forma de influir sobre esta librería de tipos, este es generado por la herramienta cuando se compila el componente. Haciendo imposible que pudieran especificar estos atributos durante la etapa de desarrollo, Visual Basic 5.0 agregó el soporte para uno de los atributos de MTS (Transacción), otros atributos no podían ser expresados y más aún estos atributos eran configurados desde el MTS. Finalmente los equipos de COM y de MTS abandonaron los formatos TLB e IDL y fueron en la búsqueda de un nuevo formato, extensible, claro y más accesible. Este nuevo contrato es el enfoque de la nueva plataforma.

Tiempo de ejecución de lenguaje común (CLR Comon Language Runtime).

Para resolver el problema con los contratos COM y sus definiciones, los equipos de MTS y COM en Microsoft, desarrollaron una nueva plataforma de componentes llamada COM3. Luego de que se escogió el nombre, se descubrió que COM3 no era un nombre de directorio legal bajo ciertas plataformas Microsoft, de tal forma que el nombre cambió a Component Object Runtime (COR). Otros nombres que se usaron durante el ciclo de desarrollo incluían COM+ Runtime, Universal Runtime (URT), y luego finalmente, justo antes de su primer beta público, la tecnología se renombró a Common Language Runtime (CLR).

Como parte de la iniciativa .NET Microsoft ha enviado partes de la plataforma a varias organizaciones encargadas de estándares. En particular, Microsoft ha enviado el Common Language Infraestructure (CLI) al ECMA (European Computer Manufacturer' s Association). El CLI incluye el common type system y el common intermediate language (CIL) usado por el CLR. Sin embargo, el CLR mismo no es parte de lo que se ha enviado a ECMA. El CLR es una implementación del CLI que es controlado exclusivamente por Microsoft.

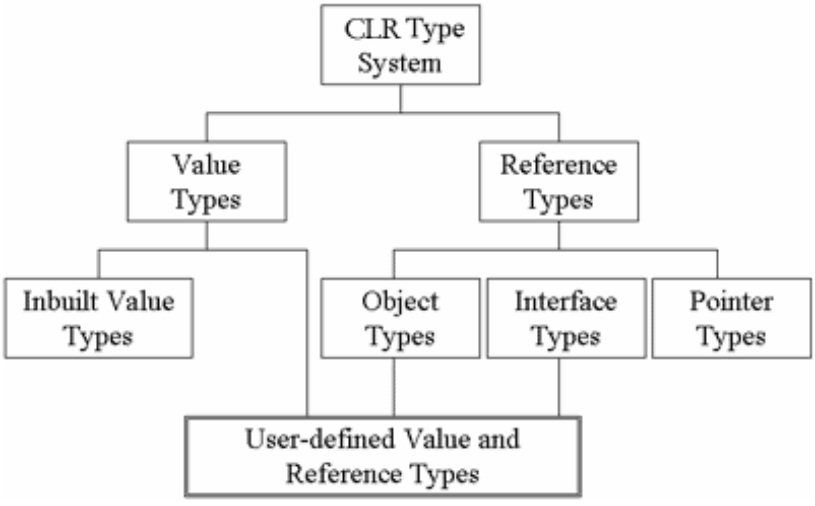


Figura 26. Common Type System.

La funcionalidad del sistema Microsoft .NET se basa en las capacidades del Common Language Infraestructure. A diferencia de algunos sistemas basados en máquinas virtuales. El CLI fue diseñado desde el principio para soportar una variedad de lenguajes de programación. Se espera también que ECMA haga que el CLI este disponible en varias plataformas y no solo Windows. Esta combinación de capacidades con múltiples lenguajes y una implementación multiplataforma hacen que el CLI sea un objetivo importante para los compiladores futuros. Las ideas de máquinas virtuales, lenguajes intermedios y lenguajes con ejecución independientes de la plataforma ha fascinado a los investigadores por largo tiempo. Ejemplos bien conocidos son JVM, Transmeta Crusoe, Vmware.

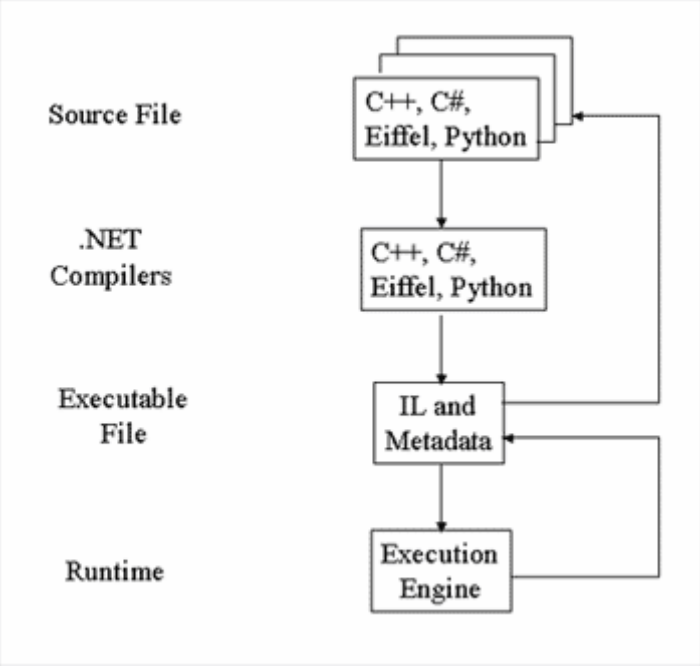


Figura 27. Soporte Multilenguaje de .NET

A diferencia de COM, el CLR se crea desde el principio con un formato completamente especificado para describir, los contratos de los componentes. Este formato es llamado en forma genérica "metadata".

El CLR provee facilidades para leer y escribir la metadata programáticamente sin el conocimiento del formato de archivo usado, esto quiere decir que el programador puede expresar los atributos desde el código, es una programación basada en atributos, por ejemplo los atributos de transacción de un componente puede ser expresados explícitamente y el CLR puede verificar de que se provea el ambiente para su ejecución. CLR metadata es claro y extensible vía atributos personalizados, los cuales a su vez están basados en tipos.

CLR metadata contiene la información sobre la dependencia de los componentes y la versión, lo cual presenta nuevas técnicas para manejar las versiones en los componentes.

Finalmente la presencia de la metadata es obligatoria, lo cual hace que la infraestructura y las herramientas basadas en el CLR sean más fáciles de usar y de construir que en ambientes donde la metadata es opcional, finalmente aquí hay un solo formato y una sola forma de representar los contratos.

La segunda forma en que los contratos en el CLR difieren de los contratos en COM es en la naturaleza misma de los contratos. En COM, un contrato de un componente implica una representación precisa en memoria de un vtable por ejemplo, y cualquier otra estructura de datos que son pasadas como métodos de los parámetros, los cuales ya tienen una expresión fija en términos de direcciones de memoria.

Los contratos en el CLR describen la estructura lógica de los tipos. Los contratos en el CLR específicamente no describen la representación en memoria. El CLR pospone las decisiones acerca de estas consideraciones de representación en memoria hasta que el tipo (o componente) se carguen por primera vez en tiempo de ejecución (Just in time).

Otra facilidad del CLR es que soporta la programación generativa que es llamada CodeDOM. El CodeDOM permite que los programas en C#/VB .NET/JavaScript sean construidos en memoria como un modelo de objetos fuertemente basada en tipos. El CodeDOM permite a los programas generativos que emiten código fuente posponer la decisión sobre el lenguaje de salida hasta el último momento.

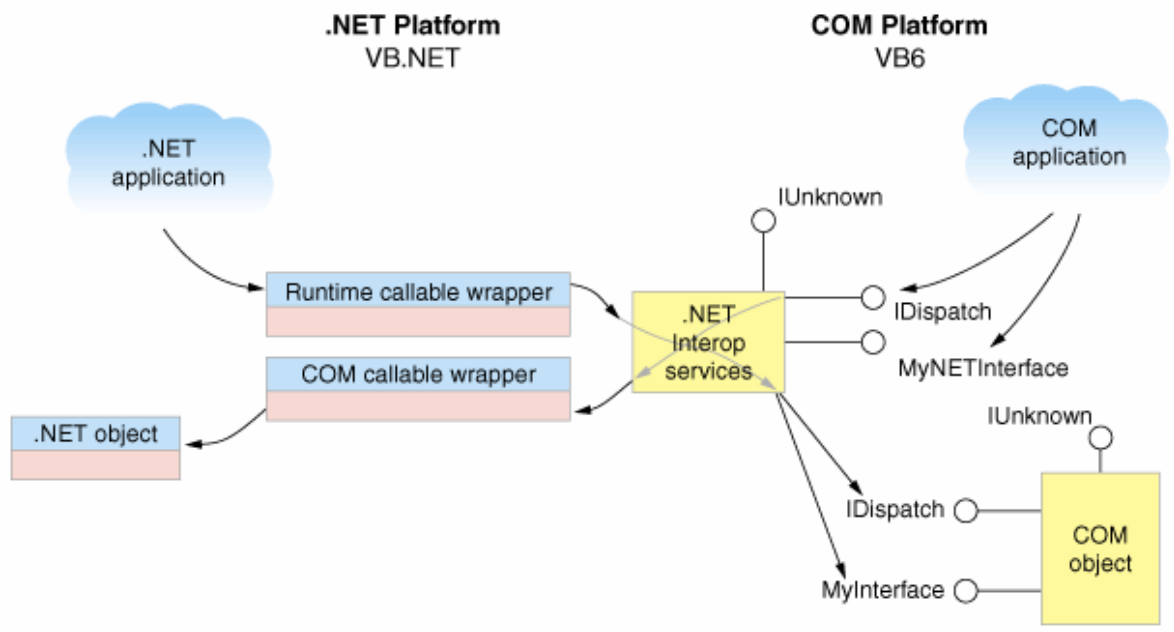


Figura 28. Interoperabilidad de .NET con COM.

Todo ambiente de ejecución tiene una noción de "componente de software". Assembly es un conjunto de archivos (módulos) que contienen un código intermedio denominado Common Intermediate Language (CIL), y metadata que sirve como una unidad primaria y fundamental de componente de software en el CLI. En términos del JVM assembly puede ser visto como un archivo JAR.

El tiempo de ejecución de lenguaje común (CLR) es un motor de ejecución de alto rendimiento. El código que tiene como objetivo del tiempo de ejecución y se administra mediante el tiempo de ejecución se conoce como código administrado. La responsabilidad de tareas como la creación de objetos, hacer las llamadas de método, etc. se delega al tiempo de ejecución de lenguaje común, el cual permite a éste proporcionar servicios adicionales al código ejecutado. A pesar de su nombre, el CLR también tiene un papel en las experiencias de tiempo de desarrollo del componente.

Mientras se esté ejecutando el componente, el tiempo de ejecución proporciona servicios, tal como administración de memoria (incluyendo recolección de basura), administración del proceso, administración de hilado y aplicación de la seguridad, además de resolver las dependencias que el componente pueda tener en otros componentes.

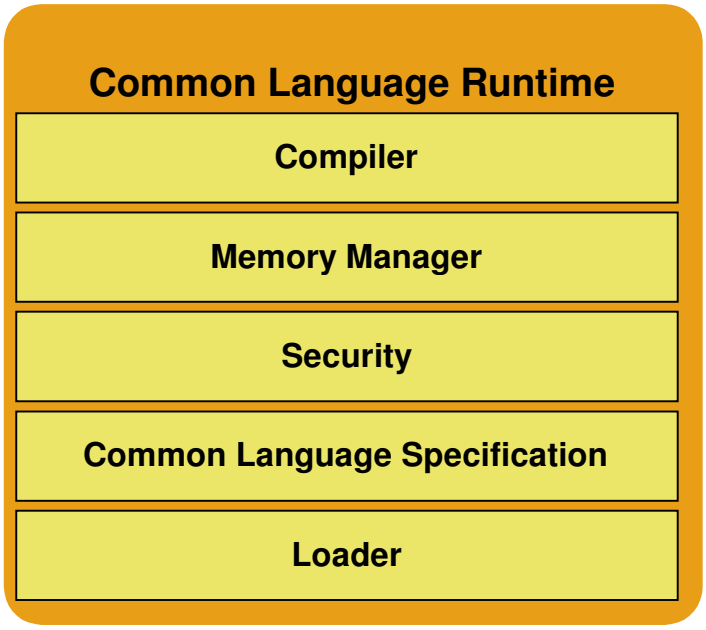


Figura 29. Common Language Runtime.

Al momento del desarrollo, el papel del tiempo de ejecución cambia ligeramente. Debido a que se automatiza tanto (por ejemplo, administración de memoria), el tiempo de ejecución hace que la experiencia del desarrollador sea muy simple.

En particular, las funciones como la administración del tiempo de vida, nombre de tipo sólido, manejo de excepción de lenguaje cruzado, administración de eventos basada en delegación, unión dinámica y reflexión reducen dramáticamente la cantidad de código que debe escribir un desarrollador para poder convertir la lógica de negocios en componentes reutilizables.

Los tiempos de ejecución no son algo nuevo para los lenguajes; casi todo lenguaje de programación tiene un tiempo de ejecución. Visual Basic tiene el tiempo de ejecución más conocido (el nombrado VBRUN), pero Microsoft Visual C++® también tiene uno (MSVCRT), como también lo tiene Microsoft Visual FoxPro®, Microsoft JScript®, SmallTalk, Perl, Python, Haskell y Java. El papel crítico del tiempo de ejecución de un lenguaje común, y lo que realmente lo hace diferente, es su disposición de un ambiente de tiempo de ejecución unificado a través de todos los lenguajes de programación.

Las funciones clave del tiempo de ejecución incluyen un sistema de tipo común (que permite la integración del lenguaje cruzado), los componentes autodescriptivos, la implementación y versión simplificadas y los servicios de seguridad integrados.

Sistema de tipo común e integración multilingüe.

El tiempo de ejecución utiliza un nuevo sistema de tipo común, capaz de expresar la semántica de los lenguajes de programación modernos. El sistema de tipo común define un conjunto estándar de tipos y reglas de datos para crear tipos nuevos. El tiempo de ejecución entiende la forma de crear y ejecutar estos tipos. Los compiladores para .NET Framework utilizan los servicios de tiempo de ejecución para definir los tipos de datos, manejar objetos y hacer llamadas de método, en lugar de utilizar los métodos específicos de la herramienta o del lenguaje.

El sistema de tipo común permite una profunda integración multilingüe. El código escrito en un lenguaje puede heredar la implementación de clases escritas en otro; las excepciones se pueden quitar del código escrito en un lenguaje y recibirse en código escrito en otro. Y para operaciones como depuración y creación de perfiles trabaja de manera transparente, sin importar los lenguajes utilizados para escribir el código. Esto significa que los desarrolladores ya no necesitan crear distintas versiones de sus bibliotecas reutilizables para cada lenguaje o compilador de programación, además de que los desarrolladores que utilizan bibliotecas de clase ya no estarán limitados a las bibliotecas desarrolladas para el lenguaje de programación que están usando.

Metadatos y componentes autodescriptivos
.NET Framework permite la creación de componentes autodescriptivos, que simplifican el desarrollo y la implementación, y mejoran la confiabilidad del sistema. La autodescripción se logra a través de metadatos, información contenida en forma binaria que complementa el código ejecutable, proporcionando detalles sobre las dependencias, versiones, etc. Los metadatos se empacan con el componente que los describe, resultando en componentes autodescriptivos.

Una ventaja clave de los componentes autodescriptivos es que ya no se necesita de otros archivos para utilizar un componente. Esto contrasta con el desarrollo típico de las aplicaciones actuales, el cual requiere archivos de encabezado por separado para definiciones de clase, archivos de Lenguaje de descripción de interfaz (IDL) por separado, bibliotecas de tipo por separado así como proxies y stubs por separado. Debido a que los metadatos se generan del código fuente durante el proceso de compilación y se almacenan con el código ejecutable, nunca están fuera de sincronización con el ejecutable.

Debido a que cada aplicación contiene una descripción completa de la misma, el tiempo de ejecución puede ensamblar dinámicamente una memoria caché de información sobre los componentes instalados en el sistema. Si esa memoria caché se daña de alguna forma, por ejemplo, el tiempo de ejecución puede reconstruirla sin que incluso lo sepa el usuario. Además de resolver los retos de desarrollo, la autodescripción elimina la dependencia en los registros de Windows para localizar componentes. Un beneficio de no basarse en los registros de Windows es la capacidad de una implementación sin tocar.

13.5 Implementación "Sin tocar"

Es la capacidad de copiar simplemente un archivo a una máquina objeto y ejecutarlo, sin que se requiera ningún registro. Además de eliminar las dependencias de registro, .NET Framework incluye otros avances de implementación que prácticamente eliminan los conflictos DLL, en los cuales las bibliotecas compartidas se desincronizan de las aplicaciones que están tratando de accederlas.

La implementación de lado a lado ahora permite que varias versiones de las mismas bibliotecas nombradas coexistan sin conflicto. .NET Framework incluye un sistema de nombramiento interno muy fuerte, que dificulta mucho más que dos bibliotecas tengan el mismo nombre de archivo y entren en error entre ellas. Si una aplicación nueva sobrescribe una biblioteca compartida, una aplicación existente que no pueda utilizar la nueva biblioteca compartida puede de hecho repararse a sí misma.

La siguiente vez que se inicie una aplicación existente, revisará sus archivos compartidos. Si encuentra que uno de los archivos compartidos tiene cambios y estos no son compatibles, puede solicitar al tiempo de ejecución que saque una versión con la que sepa que puede trabajar. Debido al sistema de seguridad, el tiempo de ejecución puede hacerlo de manera segura y la aplicación se puede reparar a sí misma.

Uno de los mayores problemas de las aplicaciones actuales es que en muchos casos tienen que tratar con diferentes archivos binarios (DLL´s), elementos de registro, conectividad abierta a bases de datos (ODBC), etc.

Para solucionarlo el Framework de .Net maneja un nuevo concepto denominado ensamblado. Los ensamblados son archivos con extensión EXE o DLL que contienen toda la funcionalidad de la aplicación de forma encapsulada. Por tanto la solución al problema puede ser tan fácil como copiar todos los ensamblados en el directorio de la aplicación.

Con los ensamblados ya no es necesario registrar los componentes de la aplicación. Esto se debe a que los ensamblados almacenan dentro de si mismos toda la información necesaria en lo que se denomina el manifiesto del ensamblado.

El manifiesto recoge todos los métodos y propiedades en forma de meta-datos junto con otra información descriptiva, como permisos, dependencias, etc.

Para gestionar el uso que hacen la aplicaciones de los ensamblados .Net utiliza la llamada caché global de ensamblados (GAC, Global Assembly Cache). Así, .Net Framework puede albergar en el GAC los ensamblados que puedan ser usados por varias aplicaciones e incluso distintas versiones de un mismo ensamblado, algo que no era posible con el anterior modelo COM.

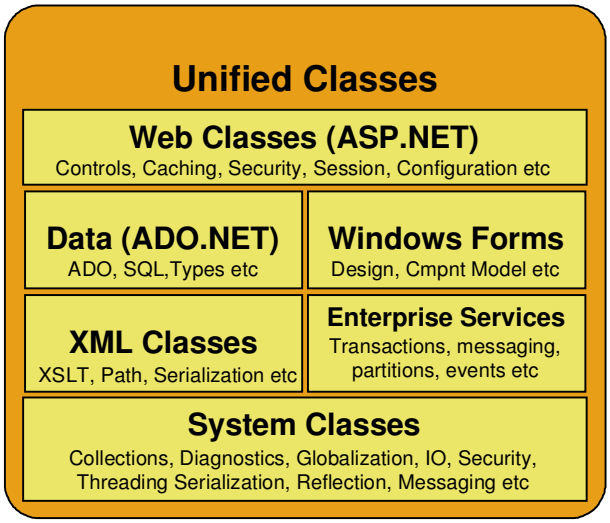


Figura 30. Lo que ve un programador: Las clases unificadas.

Cuando se está programando una aplicación muchas veces se necesitan realizar acciones como manipulación de archivos, acceso a datos, conocer el estado del sistema, implementar seguridad, etc.

El Framework organiza toda la funcionalidad del sistema operativo en un espacio de nombres jerárquico de forma que a la hora de programar resulta bastante sencillo encontrar lo que se necesita.

Para ello, el Framework posee un sistema de tipos universal, denominado Common Type System (CTS). Este sistema permite que el programador pueda interactuar los tipos que se incluyen en el propio Framework (biblioteca de clases de .Net) con los creados por él mismo (clases).

De esta forma se aprovechan las ventajas propias de la programación orientada a objetos, como la herencia de clases predefinidas para crear nuevas clases, o el polimorfismo de clases para modificar o ampliar funcionalidades de clases ya existentes.

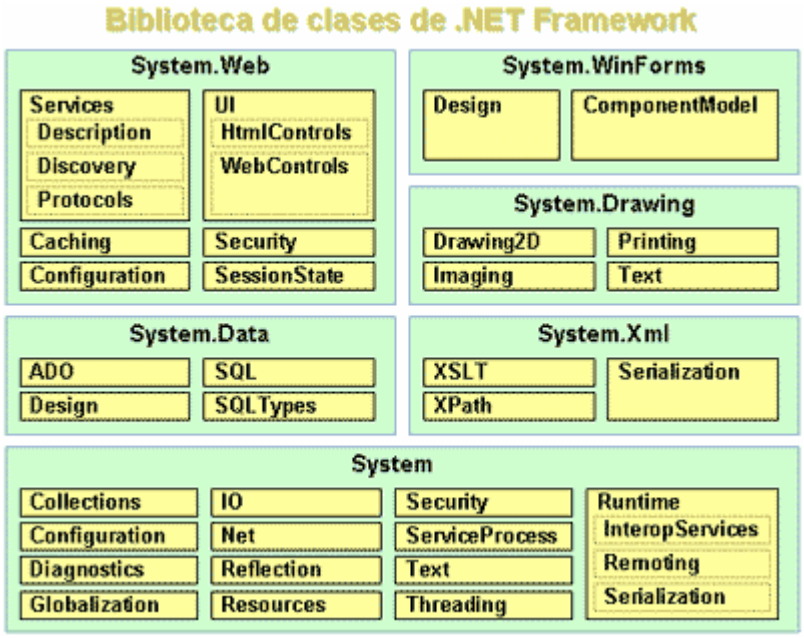


Figura 31. Biblioteca de clases de .NET Framework

La biblioteca de clases de .Net Framework incluye, entre otros, tres componentes clave:

- ASP.NET para construir aplicaciones y servicios Web.
- Windows Forms para desarrollar interfaces de usuario.
- ADO.NET para conectar las aplicaciones a bases de datos

Las clases de .NET Framework proporcionan un conjunto unificado, orientado a objetos, jerárquico y ampliable de bibliotecas de clases, o APIs que los desarrolladores pueden utilizar con los lenguajes con los que están familiarizados.

Hoy, los desarrolladores de Visual C++ utilizan Microsoft Foundation Classes, los desarrolladores de Visual J++ usan Windows Foundation Classes y los desarrolladores de Visual Basic utilizan el marco de referencia de Visual Basic. En otras palabras, las clases de .NET Framework unifican estas clases distintas, creando un súper conjunto de sus funciones.

El resultado es que los desarrolladores ya no tienen que aprender varios modelos de objeto o bibliotecas de clases. Al crear un conjunto común de APIs a través de todos los lenguajes de programación, .NET Framework permite herencias de lenguaje cruzado, manejo de errores y depuraciones. De hecho, todos los lenguajes de programación, desde JScript hasta C++, se vuelven iguales y los desarrolladores quedan en libertad de elegir el lenguaje correcto para el trabajo.

.NET Framework proporciona clases que se pueden invocar desde cualquier lenguaje de programación. Estas clases cumplen con un conjunto de lineamientos de nombre y de diseño para reducir aún más el tiempo de capacitación de los desarrolladores.

.NET Framework incluye un conjunto base de bibliotecas de clases que los desarrolladores esperarían en cualquier biblioteca estándar, como colecciones, entrada / salida, tipos de datos y clases numéricas. Además, existen clases que proporcionan acceso a todos los servicios del sistema operativo – como gráficas, operación en red, hilados, globalización, criptografía y acceso a datos – clases que pueden usar las herramientas de desarrollo, como depuración, así como un conjunto de clases que proporcionan los servicios necesarios para desarrollar aplicaciones a escala empresarial – como transacciones, eventos, particiones y mensajes.

La forma de organizar la biblioteca de clases de .Net dentro del código es a través de los espacios de nombres (namespaces), donde cada clase está organizada en espacios de nombres según su funcionalidad. Por ejemplo, para manejar archivos se utiliza el espacio de nombres System.IO y si lo que se quiere es obtener información de una fuente de datos se utilizará el espacio de nombres System.Data.

La principal ventaja de los espacios de nombres de .Net es que de esta forma se tiene toda la biblioteca de clases de .Net centralizada bajo el mismo espacio de nombres (System). Además, desde cualquier lenguaje se usa la misma sintaxis de invocación, ya que a todos los lenguajes se aplica la misma biblioteca de clases.

Los desarrolladores que escriben aplicaciones de cliente para Windows pueden utilizar las clases Windows Forms (System.Windows.Forms) para aprovechar todas las funciones de la interfaz de Windows, incluyendo los controles existentes de Microsoft ActiveX® y las nuevas funciones de Windows 2000, como las ventanas transparentes, en capas y flotantes. Los desarrolladores encontrarán que el modelo de programación y el soporte en el tiempo de diseño de las Windows Forms son muy intuitivos, debido a sus similitudes con los paquetes existentes de formas basadas en Windows.

13.6 Estadísticas de alto nivel y situación del mercado.

.NET Framework existe desde enero de 2002. Las versiones beta llegaron a más de 4 millones de desarrolladores en todo el mundo a través de 3.5 millones de CDs y 700,000 descargas únicas.

Están disponibles más de 20 lenguajes de programación para utilizarlos con .NET Framework. Ahora están disponibles numerosos componentes y controles de otros fabricantes.

Hoy día, Microsoft está implementando enérgicamente aplicaciones actuales basadas en .NET Framework. MSN® y Microsoft.com Smart 404 son sólo algunas de las muchas aplicaciones de Microsoft que han migrado a .NET Framework.

Desarrollo rápido

La capacidad multilenguaje de .NET Framework permite a los desarrolladores utilizar el lenguaje de programación que es más apropiado para una tarea dada y combinar lenguajes en una sola aplicación. La compatibilidad con .NET Framework se ha anunciado para más de 20 lenguajes de programación comerciales y académicos.

El diseño basado en componentes de .NET Framework permite a los desarrolladores escribir menos código al no tener que desarrollar detalles de infraestructura. El ejemplo .NET Pet Shop, la versión basada en .NET de la aplicación de ejemplo de prácticas recomendadas de Sun, Java Pet Store, implementa la misma funcionalidad que la versión Java 2 Enterprise Edition (J2EE), pero utiliza un tercio del código de esta versión.

Figura 32. .NET Pet Shop vs Java Pet Store – Lines of Code

Opciones mejoradas

En .NET Pet Shop, la implementación basada en .NET de Java Pet Store de Java, el rendimiento supera al de la versión J2EE en 28 veces, a la vez que atiende a 6 veces más usuarios concurrentes requiriendo una sexta parte de uso de la CPU.

.NET Framework ofrece notables ventajas de rendimiento y escalabilidad respecto a la tecnología ASP (Active Server Pages) anterior.

Estos resultados están basados en Windows 2000 Advanced Server. Microsoft ASP y Microsoft ASP.NET se ejecutan en una base de datos de SQL Server 2000.

Figura 33. Nile Benchmark of .NET

Se han anunciado compiladores de lenguaje compatibles con .NET Framework para los siguientes lenguajes de programación:

- APL
- C++
- C#
- COBOL
- Component Pascal
- Curriculum
- Eiffel
- Fortran
- Haskell
- Java
- Microsoft JScript®
- Mercury Mondrian
- Oberon
- Oz
- Pascal
- Perl
- Python
- RPG
- Scheme
- SmallTalk
- Standard ML
- Microsoft Visual Basic®

PRODUCTOS DE PLATAFORMA .NET

Para cada diferente combinación de necesidades existe un producto de plataforma .NET adecuado. Los principales productos actualmente son los siguientes:

1. .NET Framework Redistributable Package
2. .NET Framework SDK
3. Visual Studio .NET
4..NET Compact Framework (.NET Framework compacto)

.NET FRAMEWORK REDISTRIBUTABLE PACKAGE

Paquete de distribución .NET. Es el producto de plataforma .NET Framework que se debe instalar en caso de que únicamente se desee ejecutar aplicaciones .NET. En otras palabras, es lo que como desarrolladores deben instalar en cada una de las máquinas que quieran ejecutar aplicaciones en .NET.

El .NET Framework Redistributable Package contiene todas las herramientas que permiten a .NET soportar aplicaciones en cualquier lenguaje .NET; proporciona todo el soporte a las aplicaciones de cómputo móvil a través de los controles de computación móvil (mobile controls); soporta las características de IPv6 (Internet Protocol versión 6), y proporciona proveedores de datos .NET nativos para SQL Server, ODBC, y Oracle. Además, incluye las mejoras a la seguridad para aplicaciones Web, vía ASP.NET.

13.7 Evolución

La versión 1.0 de .NET Framework SDK se liberó a principios de 2002; a principios de 2003 apareció la versión 1.1. Algunas cosas integradas en la versión 1.1 no estaban en la versión 1.0, por lo que debían ser integradas de forma adicional; por ejemplo, el soporte a controles de computación móvil sólo estaban disponibles si se instalaba Microsoft Mobile Internet Toolkit; además, sólo se disponía de proveedores de datos .NET nativos para SQL Server. Era necesario implementar cada proveedor de datos nuevo mediante un proceso de instalación, a medida que se liberaba.

.NET Framework SDK

Kit de Desarrollo de Software de .NET Framework | .NET Framework Software Development Kit. Es el producto de plataforma .NET que debe instalar si además de ejecutar aplicaciones desea desarrollarlas; .NET Framework SDK cuenta con todas las herramientas para escribir, construir, probar e implementar aplicaciones. .NET Framework contiene documentación, ejemplos, y herramientas de línea de comando para compilar y analizar los programas desarrollados.

Es importante que mantenga la congruencia de versiones; si desarrolla en .NET Framework SDK, el .NET Framework Redistributable Package debe ser de una versión igual o posterior, a fin de garantizar que todo funcionará. Por lo general, las versiones más nuevas de .NET Framework Redistributable Package ejecutarán sin problemas los programas desarrollados con .NET Framework SDK de versiones anteriores.

Por el momento, .NET Framework SDK es gratuito, y probablemente siga así. La razón es muy sencilla: por sus características particulares, la competencia directa de la plataforma .NET es Java. Para desarrollar en Java, desde hace mucho tiempo usted puede descargar el JDK (Java Development Kit) de forma gratuita; por tanto, ponerle precio a .NET Framework SDK sería darle ventaja a la competencia, lujo que por el momento Microsoft no puede darse.

Visual Studio .NET.

Entorno Integrado de Desarrollo .NET. Es el producto de plataforma .NET que permite desarrollar en un ambiente gráfico e integrado totalmente a .NET Framework; Visual Studio.NET es la herramienta de desarrollo asistido más sofisticada de la historia. Incluye editores con ayuda contextual, diseñadores gráficos de aplicaciones, asistentes de desarrollo, generadores de código, etcétera. En Visual Studio.NET pueden desarrollarse de manera visual cualquier tipo de aplicación soportada por .NET, incluyendo las aplicaciones de computación móvil.

.NET Compact Framework

.NET Framework compacto. Es el producto de plataforma .NET que permite integrar a los dispositivos inteligentes un conjunto compacto de funciones disponibles en .NET Framework, a fin de que se puedan ejecutar aplicaciones .NET, principalmente en Pocket PC y equipos con Windows CE. Lo que pretende Microsoft es extender su plataforma de desarrollo para atender cualquier tipo de dispositivo de cómputo, desde PDA hasta teléfonos celulares. Por su parte, los controles de computación móvil atienden a todo tipo de dispositivos que tienen capacidad de enlazarse a una página en Internet, mientras que .NET Framework compact se encarga de aquellos dispositivos que poseen sus características de operación independientes de una conexión a Internet.

.NET Framework compacto debe instalarse adicionalmente a los demás productos. Probablemente en el futuro forme parte, tanto del SDK como de Visual Studio.

13.8 Consideraciones respecto al idioma de los productos

Idiomas de los productos. .NET Framework Redistributable Package, al igual que el resto de los productos de plataforma .NET, pueden soportar varios idiomas, aunque quizá no es lo más conveniente. Se Recomienda que lo instale en inglés, ya que contará con el soporte más actual, corrección de errores, ayuda y documentos técnicos actualizados todo el tiempo. El resto de los lenguajes siempre estarán rezagados.

La versión 1.0 de .NET Framework ya estaba disponible en español, pero la documentación no. La versión 1.1, no estaba disponible en español; para actualizar el lenguaje, era necesario instalar el .NET Framework Versión 1.1 Language Pack, que sólo soportaba lenguajes asiáticos.

13.9 Referencias, vínculos, fuentes y mucha más información.

Para obtener la última versión de .NET Framework, visite el sitio Web de Microsoft .NET. http://www.eu.microsoft.com/latam/net/.

Para obtener más información sobre compiladores, consulte la sección Partners del sitio Web de Visual Studio en http://msdn.microsoft.com/vstudio/partners/language/ Busque controles y componentes para utilizarlos con .NET Framework Beta en la sección Asociados http://msdn.microsoft.com/vstudio/partners/component/ del sitio Web de Visual Studio . Observe código fuente y las notas del producto de .NET Pet Shop en http://www.gotdotnet.com/team/compare/petshop.aspx.

Para obtener más información sobre soluciones para clientes generadas en .NET Framework, visite http://msdn.microsoft.com/vstudio/productinfo/casestudies/ para obtener una lista completa de los casos de estudio de .NET Framework.

14. ASP.NET

Aplicaciones Web: ASP.NET

El 20 de Enero del 2002 una nueva versión del tradicional Active Server Pages se lanzó al público, Es ASP.net.

ASP.net forma parte del .NET Framework de Microsoft, junto con VB.net, C++ .net, C#, Etc. Mientras ASP se escribía en VBScript, ASP.net puede ser escrito en cualquier lenguaje soportado por el .net Framework, es decir: VB.net; C# y JScript.net. Otro cambio radical es que ASP.net es un lenguaje totalmente orientado a objeto, La principal referencia de ASP.NET la puede ver http://www.asp.net/, un portal de Microsoft donde se describe la tecnología y sus ventajas. Junto con los motivos para utilizar ASP.NET, tiene más de 900 ejemplos que le pueden ayudar a conocer el entorno de forma rápida, junto con tutoriales más detallados para conocer con detalle ciertos aspectos de la plataforma.

ASP.NET es una plataforma que se integra con un servidor web (IIS, Apache, etc.) y que permite que se accedan a páginas con extensión "aspx" las cuales pueden tener junto al HTML, código en diferentes lenguajes de programación, aunque sólo se debe de utilizar uno por página.

14.1 Mejoras de ASP.Net comparado con su antecesor ASP

Rendimiento: la aplicación compila en una sola vez al lenguaje nativo, y luego, en cada petición tiene una compilación Just In Time, es decir se compila desde el código nativo, lo que permite mucho mejor rendimiento. También permite el almacenamiento del caché en el servidor

Rapidez en programación: mediante diversos controles, puede con pocas líneas y en menos de 5 minutos mostrar toda una base de datos y hacer rutinas complejas.

Web services: Consta de herramientas para compartir datos e información entre distintos sitios para integrar los web services a las aplicaciones.

Seguridad: tiene diversas herramientas que garantizan la seguridad de las aplicaciones.

Al ser un conjunto de clases dentro de la biblioteca de clases unificada, ASP.NET proporciona un modelo de aplicaciones Web en la forma de un conjunto de controles e infraestructura que facilita desarrollar las aplicaciones Web.

ASP.NET viene con un conjunto de controles del lado del servidor (algunas veces llamadas Web Forms) que reflejan los artilugios típicos de la interfaz HTML (incluyendo cuadros de lista, de texto y botones), así como un conjunto adicional de controles Web que son más complejos (como calendarios y banners).

De hecho, estos controles se ejecutan en el servidor Web y proyectan su interfaz como HTML a un explorador. En el servidor, los controles exponen un modelo de programación orientado a objetos que proporciona al desarrollador Web riqueza de programación orientada a objetos.

Una función importante de estos controles es que se pueden escribir para adaptarse a las capacidades del lado del cliente; las mismas páginas se pueden usar para enfocarse a una amplia gama de plataformas de cliente y factores de forma. En otras palabras, los controles de Web Forms pueden espiar en el cliente que está solicitando una página y devolver una experiencia de usuario apropiada: WML para teléfonos o HTML 3.2 para un explorador de nivel inferior y HTML dinámico para Internet Explorer 5.5

ASP.NET proporciona administración de estado de sesión del cluster y reciclamiento de proceso, la cual reduce aún más la cantidad de código que un desarrollador debe escribir y aumenta la confiabilidad de la aplicación.

ASP.NET utiliza estos conceptos para permitir a los desarrolladores ofrecer software como un servicio. Al usar las funciones de los servicios ASP .NET del Web XML, los desarrolladores de ASP.NET pueden simplemente
escribir su lógica de negocios y la infraestructura ASP.NET será responsable de proporcionar ese servicio a través de SOAP y de otros protocolos públicos.

ASP.NET funciona con todos los lenguajes y herramientas de desarrollo (incluyendo Visual Basic, C++, C# y JScript).

14.2 Dentro de ASP.NET

En el núcleo de ASP.NET está el tiempo de ejecución HTTP (diferente del tiempo de ejecución de lenguaje común), un motor de ejecución de alto rendimiento para procesar comandos http. El tiempo de ejecución HTTP es responsable de procesar todas las solicitudes http entrantes, resolviendo el URL de cada solicitud a una aplicación y luego despachando la solicitud a la aplicación para un procesamiento posterior. El tiempo de ejecución HTTP es multihilado y procesa las solicitudes de manera síncrona, lo cual significa que no puede bloquearse por un mal código de aplicación desde el procesamiento de solicitudes nuevas. Además, el tiempo de ejecución HTTP tiene un diseño flexible que está hecho para recuperarse automáticamente de violaciones de acceso, fugas de memoria, seguros, etc.

14.3 Actualización de las aplicaciones

ASP.NET utiliza las tecnologías de implementación de Microsoft .NET Framework, ganando así beneficios como la implementación XCOPY y la implementación de aplicaciones lado a lado.

Otro beneficio principal de ASP.NET es el soporte para actualizaciones de aplicaciones en vivo. Un administrador no necesita apagar el servidor Web ni la aplicación para actualizar los archivos de la misma: los archivos de la aplicación nunca se cierran, de tal manera que se pueden sobrescribir aún cuando se esté ejecutando la aplicación. Cuando se actualizan los archivos, el sistema se cambia automáticamente a la versión nueva.

14.4 Capacidad de ampliación

Dentro de una aplicación ASP.NET, en última instancia, las solicitudes HTTP se enrutan a través de una "tubería" de módu los HTTP a un manejador de solicitudes. Los módulos http y los manejadores de solicitudes son clases de .NET administradas que implementan interfaces específicas definidas por ASP.NET. Esta arquitectura modular facilita en gran medida agregar servicios a las aplicaciones: necesita proporcionar un módulo HTTP. Por ejemplo, la seguridad, administración de estado y seguimiento se implementan como módulos HTTP por ASP.NET. Los modelos de programación de nivel superior, como los web services XML y los Web Forms, también se implementan como manejadores de solicitudes. Se puede asociar una aplicación con varios manejadores de solicitudes, uno por URL, pero todas las solicitudes HTTP en una aplicación dada se enrutan a través de los mismos módulos HTTP.

14.5 Administración de estado

El Web es un modelo fundamentalmente sin estado, que no tiene correlación entre las solicitudes http. Esto puede hacer difícil escribir aplicaciones Web, ya que normalmente las aplicaciones necesitan mantener un estado a través de solicitudes múltiples. ASP.NET mejora los servicios de administración de estado introducidos por ASP y proporciona tres tipos de estados para las aplicaciones Web: aplicación, sesión y usuario. El estado de sesión ASP.NET se almacena en un proceso por separado y se puede configurar para que se almacene en una máquina por separado, o para que persista en una base de datos Microsoft SQL Server™. Esto hace que el estado de la sesión sea escalable, aún cuando se implemente una aplicación a través de las granjas Web más grandes.

El estado del usuario se parece al estado de sesión, pero generalmente no tiene tiempo límite y es continuo. Así, el estado del usuario es útil para almacenar las preferencias del mismo y otra información personalizada. Todos los servicios de administración de estados se implementan como módulos HTTP, de tal manera que se pueden agregar, ampliar o hasta eliminar fácilmente de un pipeline de la aplicación. Si se requieren servicios adicionales de administración de estado más allá de los que proporciona ASP.NET, se pueden proporcionar por medio de un módulo de un tercero.

14.6 Guardar en la memoria caché

El modelo de programación ASP.NET proporciona una API con memoria caché que permite a los programadores activar los servicios para guardar en la memoria caché (en software empresarial) y así mejorar el rendimiento.

Una memoria caché de salida guarda páginas completamente proporcionadas y una memoria caché fragmentada almacena páginas parciales. Se proporcionan clases para que las aplicaciones, módulos HTTP y manejadores de solicitudes puedan almacenar objetos de manera arbitraria en la memoria caché según sea necesario.
Como parte de ASP.NET, se proporcionan capacidades avanzadas para guardar en la memoria caché. ASP.NET está diseñado para proporcionar un ambiente robusto de aplicaciones Web, capaz de ejecutar proyectos de misión crítica por largos periodos de tiempo.
ASP.NET ya proporciona alguna mejora importante en el rendimiento de las aplicaciones Web, una mejora de rendimiento hasta tres veces superior sobre las aplicaciones existentes basadas en ASP y mejoras en la productividad aún más importantes.

ASP.NET incluye funciones avanzadas de compilación y de memoria caché que mejoran el rendimiento por un factor de dos a tres sobre las aplicaciones existentes de Active Server Pages.

Aplicación Web típica que ejecuta ASP contra ASP.NET; ambas en la misma computadora que ejecuta Windows 2000 y SQL Server 2000

¿Qué diferencias de sintaxis hay con respecto de ASP 3.0?

Al ser programación orientada a objetos todo es diferente. Hay que cambiar el pensamiento y todos los viejos conceptos que se tenían de ASP 3.0.

ASP.net es parecido a Visual Basic, en la página, todos los elementos son objetos, activos de servidor (que se generan ahí) y tienen propiedades y métodos.

En ASP.net existen los formularios activos del servidor que son como los formularios tradicionales de HTML, pero se le agrega en el TAG FORM el atributo runat=server, y dentro del formulario se agregan los controles activos de servidor. Cuando ocurre un evento, la página se auto envía a si misma y se procesa el controlador para ese evento. EJ:

```
<%@ Page language="VB"%>
<%@ Import Namespace="System.data.oledb" %>

<script runat="server">
Sub Nombre_change(sender as object, e as EventArgs)
Mensaje.text = "Buenos días " + Nombre.text
End sub
</script>

<html>
<body>
<font face="verdana" size=2>Saludos con Asp.net!</font><br><br>
<form runat="server">
```

```
Tu nombre: <asp:TextBox id="Nombre" OnTextChanged="Nombre_change" runat="server"
autopostback="true"/>
<asp:label id="Mensaje" runat="server" />
</form>
</body>
</html>
```

Si quiere observar el resultado, copie y peguen el código en el bloc de notas y guárdelo en su carpeta WWWROOT como bienvenida.aspx, tome nota de que para correr ASP.net deben tener el .NET Framework instalado en su PC, de otra manera no funcionará. En su navegador vaya a http://localhost/bienvenida.aspx. Digite su nombre y sitúe el cursor fuera del casillero de texto, y aparece un mensaje que le da buenos días. Observe como funciona.

En el código hay tres párrafos:

En las declaraciones del primer párrafo puede ver que le dan dos instrucciones, la primera, dice al compilador que el lenguaje utilizado es VB.net, de otra manera diría "C#", nótese la ausencia del ".net". En la segunda instrucción, vea que importa el espacio de nombre System.Data.OleDB, que sirve para trabajar con bases de datos, innecesario en esta página, pero se incluye para que mostrar como se incluye un espacio de nombre. Un espacio de nombre es el que permite realizar determinada función, por ejemplo, si quiere utilizar las características que trae ASP.net para trabajar con XML, importaría el espacio de nombre "System.XML".

El segundo pude observar sólo el método Nombre_change, correspondiente al evento del campo de texto Nombre, o sea, se ejecuta cuando el campo de texto Nombre cambia. Lo que hace es poner en la etiqueta Mensaje el valor "Buenos días " más el valor que contenga el campo de texto Nombre.

El tercero es el conocido HTML, pero con un formulario con RunAt="Server", o sea, estos controles se ejecutan en el servidor y luego procesados resultan código HTML común, o sea por ejemplo: <input name="Nombre" type="text" id="Nombre">

Una vez procesado todo el código, envía el siguiente resultado al navegador:

```
<html>
<body>
<font face="verdana" size=2>Saludos con Asp.net!</font><br><br>
<form name="_ctl0" method="post" action="bienvenida.aspx" id="_ctl0">
<input type="hidden" name="__EVENTTARGET" value="" />
<input type="hidden" name="__EVENTARGUMENT" value="" />
<input type="hidden" name="__VIEWSTATE"
value="dDwxMTU3NzQ3OTc0Ozs+jxrUecWhDY6AanZbrFANP9MYypQ=" />

<script language="javascript">
<!--
function __doPostBack(eventTarget, eventArgument) {
```

```
var theform = document._ctl0;
theform.__EVENTTARGET.value = eventTarget;
theform.__EVENTARGUMENT.value = eventArgument;
theform.submit();
}
// -->
</script>

Tu nombre: <input name="Nombre" type="text" id="Nombre"
onchange="__doPostBack('  Nombre' ,'  ' )" language="javascript" />
<span id="Mensaje"></span>
</form>
</body>
</html>
```

14.7 XSP (Ximian ASP.NET)

XSP es la implementación que está llevando Ximian de ASP.NET, la propuesta de páginas activas en el servidor que Microsoft ha presentado dentro de su plataforma .NET.

En la actualidad se ha implementado el parser de páginas ASP y el generador de código para poder ejecutar Aplicaciones ASP Web.

Dentro de la plataforma Mono los web services son una de las partes que más demanda reciben, ya que las plataformas web se encuentran ya implantadas de forma muy amplia y permiten acceder con navegadores genéricos e independientes de la plataforma a los servicios. Gracias a XSP puede obtener dichos servicios de forma cómoda utilizando la flexibilidad de la plataforma Mono en el momento de desarrollo.

Con XSP no tendrá como requisito desarrollar en la plataforma Microsoft ya que el software, funciona en GNU/Linux con Mono, como también dentro de ASP.NET, al menos mientras Microsoft no introduzca sistemas que bloquee esta portabilidad.

Los desarrolladores de software libre, podrían tener poco interés en ASP.NET al ser una plataforma cerrada, pero ahora que Ximian ha lanzado la oferta de una alternativa libre, esta adquiere un gran interés para ellos. Se tienen todos los recursos de ASP.NET para aprender, todas las librerías que se vayan haciendo (XSP es capaz de ejecutar las páginas aspx que corren en ASP.NET) para ASP.NET así que, todo parece ventaja. La migración de sistemas ASP.NET de plataformas cerradas como Windows 2000 con IIS a plataformas libres con GNU/Linux y Mono es ahora mucho más sencilla que, cuando había que utilizar extensiones de Apache para poder ejecutar código ASP en máquinas GNU/Linux, el interés por esta plataforma es enorme dentro del mundo de software libre.

Lo primero para comenzar con XSP es bajar el módulo del CVS de Mono:
acs@merry:~/src$ cvs -d :pserver:anonymous@anoncvs.go-mono.com:/mono co xsp
acs@merry:~/src$ cd xsp/
acs@merry:~/src/xsp$ ls
CVS doc Makefile README src test

Lo primero es leer el archivo README, que en este caso contiene información interesante.
XSP es la implementación de Ximian del parser de página y generador de código
de ASP.NET para la ejecución de Aplicaciones ASP Web.

Estructura:

src/
archivos de fuente para el parser/generador.

doc/
Documentación de las páginas ASP.NET

test/
Pequeños ejemplos de páginas ASP.NET

Responsable: gonzalo@ximian.com

El archivo no está en español, pero se ha traducido, Dentro del directorio "doc" tiene documentos que describen el lenguaje ASPX, el código C# generado desde una página ASPX y como ayudar en el proyecto.

Dentro del directorio "test" tiene varios ejemplos de páginas "aspx" para poder comprobar el correcto funcionamiento del parser y del generador de código, finalmente el directorio "src", es donde esta la implementación real de XSP. En este mismo directorio tiene un servidor web para pruebas que le servirá para comprobar el funcionamiento de las páginas ASPX.

La idea ahora es compilar el parser de ASPX, el generador de código, comprobar utilizando el servidor de web de pruebas que las páginas ASPX se procesan y se sirven de forma correcta.

14.8 Compilación de XSP

Llega el momento de comenzar con XSP. Para ello vaya al directorio de fuentes, "src" y lo compila. Para ello, compile las fuentes de "mcs" y las instala, luego compile "mono" con las opciones:

acs@merry:~/src/mono$ ./autogen.sh --prefix=/usr/local/ --with-gc=none

Hay que asegurarse de utilizar esta versión de mono y por ejemplo, no utilizar la versión empaquetada para Debian. Para ello, ponga las siguientes variables de entorno:
export PATH=/usr/local/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH

Compile XSP:
acs@merry:~/src$ cvs -d :pserver:anonymous@anoncvs.go-mono.com:/mono co xsp
cvs server: Updating xsp
U xsp/Makefile
...
U xsp/test/web_textbox.aspx
acs@merry:~/src$
acs@merry:~/src$ cd xsp
acs@merry:~/src/xsp$ ls
CVS doc Makefile README src test
acs@merry:~/src/xsp$ make install
acs@merry:~/src/xsp$ make install
(cd src && make)
make[1]: Entering directory ` /home/acs/src/xsp/src'
mcs /r:System.Web.dll /r:System.Drawing.dll /define:MONO /out:xsp.exe asp-parser.cs asp-tokenizer.cs aspelements.cs driver.cs generator.cs
Compilation succeeded
(cd HttpServer && make)
make[2]: Entering directory ` /home/acs/src/xsp/src/HttpServer'
mcs /r:System.Web.dll /define:MONO /out:server.exe server.cs
Compilation succeeded
make[2]: Leaving directory ` /home/acs/src/xsp/src/HttpServer'
make[1]: Leaving directory ` /home/acs/src/xsp/src'
mkdir rundir
mkdir rundir/output
cp src/xsp.exe src/HttpServer/server.exe src/HttpServer/redirector.sh rundir/
cp test/.aspx rundir/*
cp test/.png rundir/*
*mcs --target library -o rundir/output/tabcontrol.dll *
-r System.Web test/tabcontrol.cs
Compilation succeeded
*mcs --target library -o rundir/output/tabcontrol2.dll *
-r System.Web test/tabcontrol2.cs
Compilation succeeded

Now cd to rundir and run: mono server.exe 8000
Then point your web browser to http://127.0.0.1:8000/
Enjoy

Ya tiene compilado el parser, el generador de código y el servidor de web de pruebas. Además, le indica como tiene que arrancar el servidor de web (cd rundir; mono server.exe 8000) y a que URL debe de acceder con el navegador, al puerto 8000 de su máquina, donde tenemos allí operando el servidor web.
La primera página con la que se encontramos es la siguiente:

Puede observar que esta página lo lleva a múltiples ejemplos de código ASPX funcionando ya dentro de Mono. Ha llegado el momento de comenzar con Mono.

Análisis de una petición ASPX
Observe como se procesa un archivo ASPX y vea los pasos que se llevan a cabo para procesar dentro del servidor este tipo de archivos, pasos que concluyen devolviendo al cliente web en formato HTML los resultados de su petición. Para ello, lo primero es arrancar el servidor:
acs@merry:~/src/xsp/rundir$ mono server.exe 8000
Remember that you should rerun the server if you change
the aspx file!
Server started.

Este primer mensaje indica que el servidor de web mantiene cache de las páginas que ya se han accedido, dejando en esta cache precompilados los ASPX. Si cambia alguno, tiene que volver a reiniciar el servidor para que limpie el cache.

Se analizará el ejemplo "button.aspx" que es bastante útil para comenzar. El resultado de acceder a este ejemplo es:

Figura: Ejemplo de botón ASPX en XSP
Los registros que deja el servidor son:
Accepted connection.
Started processing...
mono xsp.exe button.aspx
Output goes to output/xsp_button.cs
Script file is output/xsp_button.aspx.sh
mcs --target library -L . -r corlib -r System -r System.Data -r System.Web -r System.Drawing -o output/1566455972button.dll output/xsp_button.cs
Output goes to output/output_from_compilation_xsp_button.txt
Script file is output/last_compilation_xsp_button.bat
Finished processing...

Lo primero que se hace es acceder al código de "button.aspx" y utilizar el generador de código "xsp.exe" que, a partir de la página "button.aspx", genera un programa en C# que se encargará de generar la página HTML para el cliente.

14.9 El código ASPX

Al utilizar ASPX para hacer las páginas web lo hace forma similar que PHP: tiene un lenguaje de programación más potente que HTML. Con este lenguaje incluso pude acceder a bases de datos, realizar conexiones por red y otro tipo de acciones que no son posibles utilizando HTML estático.

El gran objetivo con este tipo de lenguajes del lado del servidor es ofrecer a los desarrolladores de sitios web un sistema sencillo de utilizar una plataforma de desarrollo potente, como puede ser PHP o Mono, y ser invocados y devolver los resultados a través de interfaces web.

El código del programa "button.aspx" es el siguiente:

```
<%@ Page Language="C#" %>
<html>
<head>
   <script runat="server">
   void Button1_OnClick(object Source, EventArgs e)
   {
      HtmlButton button = (HtmlButton) Source;
      if (button.InnerText == "Enabled 1"){
         Span1.InnerHtml="You deactivated Button1";
         button.InnerText = "Disabled 1";
      }
      else {
         Span1.InnerHtml="You activated Button1";
         button.InnerText = "Enabled 1";
      }
   }

   </script>
</head>
<body>
   <h3>HtmlButton Sample</h3>
   <form id="ServerForm" runat="server">
      <button id=Button1 runat="server" OnServerClick="Button1_OnClick">
      Button1
      </button>

      <span id=Span1 runat="server" />
   </form>
</body>
</html>
```

Como puede observar es una página HTML en la que se ha introducido código C#. En general es mejor evitar mezclar HTML con código fuente, en este caso C#, para poder separar de forma clara el trabajo de diseño (HTML) del de programación, y utilizar plantillas para fundir el HTML en el código. En este primer ejemplo, y para simplificarlo, se mezclara ambos en un único archivo.

Al comienzo de la página se indica que el lenguaje a utilizar es "C#". Esto muestra que en ASPX se deja abierta la opción de utilizar diferentes lenguajes, aunque un cambio de lenguaje exige utilizar otro generador de código. En Mono de momento sólo se puede utilizar C#.

Dentro de la página HTML utiliza una etiqueta especial, "button", que es reconocida por el generador de código de XSP y que será sustituida por los elementos HTML necesarios. Podemos ver el código C# que se genera para generar este botón:

```
private System.Web.UI.Control __BuildControl_Button1 ()
    {
        System.Web.UI.HtmlControls.HtmlButton __ctrl;

        __ctrl = new System.Web.UI.HtmlControls.HtmlButton ();
        this.Button1 = __ctrl;
        System.Web.UI.IParserAccessor __parser = (System.Web.UI.IParserAccessor)
__ctrl;
        __ctrl.ID = "Button1";
        __ctrl.ServerClick += new System.EventHandler (this.Button1_OnClick);
        __parser.AddParsedSubObject (new System.Web.UI.LiteralControl
("\n\t\tButton1\n\t\t"));

        return __ctrl;
    }
```

Puede observar que este método se encargar de crear el botón (Button1), de asociarle lo que ocurre cuando se pulsa el botón y de asignarle una etiqueta. Vea que este control se trata como si fuera casi un "widget" de un toolkit gráfico, aunque finalmente termina generando código HTML. Esto en realidad son las tripas de la implementación y no tienen especial interés para el desarrollador.

Todas las partes del HTML que necesitan ser procesadas por el servidor dentro de Mono, se marcan con la etiqueta "runat=server". De esta forma, el servidor sabe que partes debe de entregar al cliente web tal cual, sin modificar, y cuales deben de ser procesadas dentro de XSP.

Partes de ASPX
Controles
En ASPX existen tres tipos de controles, estos son los WebServerControl, los HTMLServerControl y los controles de validación. Los primeros son elementos de la interfaz gráfica que se van a mostrar al usuario y se van a ejecutar en la aplicación, tales como puede ser mostrar un calendario, un botón o mismamente una imagen. Los segundos son muy parecidos a lo que es la sintaxis html son las etiquetas o tags, únicamente que se van a poder hacer referencia sin necesidad de tags como si fuesen simples elementos que el programador hace referencia.

Controles del servidor web (WebServerControl)
Muchos controles del servidor web, equivalen a controles HTML, tales como botones, controles de texto, etcétera.
Entre los controles puede ver los siguientes:
* AdRotator: Para mostrar una secuencia de anuncios
* Button: Se muestra un botón al que le puede asociar un evento, al estilo GTK+ o GTK#
* Calendar: Muestra un calendario
* CheckBox: Típica casilla de comprobación con su equivalente como control HTML.
* CheckBoxList: Conjunto de casillas de comprobación
* DataGrid: Para mostrar información de una BBDD
* DataList: Lista con información de una BBDD
* DropDownList: Lista desplegable
* HyperLink: Vínculo a un web site
* Image: Muestra una imágen
* ImageButton: Muestra un botón con una imágen
* Label: Muestra una etiqueta de texto
* LinkButton: Botón con un enlace a una url
* ListBox: Lista de elementos
* Panel: Panel que contiene otros controles, bien, un Button, CheckBox o cualquier otro.
* RadioButton: Botón de radio, una única opción a seleccionar
* RadioButtonList: Grupo de botones de radio
* Table: Una tabla con datos
* TableRow: Fila de una tabla
* TextBox: Caja de texto

Todos estos controles forman parte de lo que se llaman los controles del servidor al igual que los controles HTML porque se procesan en la parte del servidor donde estos residen, bien sea la máquina cliente o servidor, al igual que son traducidos a controles HTML para poderse mostrar en el navegador web.

14.10 Controles HTML

Bueno los controles HTML se asemejan a lo que son las etiquetas en HTML, sólo que viene a ser más parecido a lo que el programador necesita, ya que dota al programador de una serie de controles que hacen referencia a lo que son las etiquetas en HTML. Para ello se mostrara a continuación cuales se tiene disponibles, obviamente la mayoría de los definidos por la W3C en HTML 4.0.

Los controles HTML o HTMLServerControl al igual que sus equivalentes son los siguientes:
HtmlAnchor: <a>
HtmlButton: <button>
HtmlForm: <form>
HtmlGenericControl: Resto de etiquetas no asignadas a los controles.
Image: <img>
HtmlInputButton

14.11 Ejecución del ASPX

Se ha visto que la página ASPX pasa a generar un programa en C# que es el que generará el HTML. Ahora llega el momento de ver como el servidor compila ese programa y lo ejecuta, algo que se realiza en el script "Script file is output/xsp_button.aspx.sh".

redirector.sh mono output/xsp_button.cs xsp.exe button.aspx

Tras ello, se pasa a crear un archivo "dll" que será el que utilice el servidor para obtener la página HTML. El "dll" se genera con:

mcs --target library -L . -r corlib -r System -r System.Data -r System.Web -r System.Drawing -o output/1566455972button.dll output/xsp_button.cs

Como puede ver, lo que se hace es compilar el archivo "xsp_button.cs" que fue anteriormente generado a partir de la página HTML con ASPX por XSP. Este "dll" se guardará ya para futuras peticiones de esta página, evitando tener que volver a generar la página C# y compilarla.

En realidad, dentro del servidor se crea un "AppDomain" en el que se carga el "dll". Esto es así ya que puede descargar de memoria un "AppDomain", lo que permitirá refrescar las librerías "dll" en caso de modificaciones.

15. INTRODUCCIÓN A C#

15.1 Origen y necesidad de un nuevo lenguaje.

C# (leído en inglés "C Sharp" y en español "C Almohadilla") es el nuevo lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Sus principales creadores son Scott Wiltamuth y Anders Hejlsberg, éste último también conocido por haber sido el diseñador del lenguaje Turbo Pascal y la herramienta RAD Delphi. Aunque es posible escribir código para la plataforma .NET en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado en ella, por lo que programarla usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes ya que C# carece de elementos heredados innecesarios en .NET. Por esta razón, se suele decir que C# es el lenguaje nativo de .NET.

La sintaxis y estructuración de C# es muy similar a la C++, ya que la intención de Microsoft con C# es facilitar la migración de códigos escritos en estos lenguajes a C# y facilitar su aprendizaje a los desarrolladores habituados a ellos. Sin embargo, su sencillez y el alto nivel de productividad son equiparables a los de Visual Basic.

Un lenguaje que hubiese sido ideal utilizar para estos menesteres es Java, pero debido a problemas con la empresa creadora del mismo Sun, Microsoft ha tenido que desarrollar un nuevo lenguaje que añadiese a las ya probadas virtudes de Java las modificaciones que Microsoft tenía pensado añadirle para mejorarlo aún más y hacerlo un lenguaje orientado al desarrollo de componentes.

En resumen, C# es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo. El hecho de ser relativamente reciente no implica que sea inmaduro, pues Microsoft ha escrito la mayor parte de la BCL usándolo, por lo que su compilador es el más depurado y optimizado de los incluidos en el .NET Framework SDK.

15.2 Características de C#.

Con la idea de que los programadores más experimentados puedan obtener una visión general del lenguaje, a continuación se recoge de manera resumida las principales características de C# Alguna de las características aquí señaladas no son exactamente propias del lenguaje sino de la plataforma .NET en general. Sin embargo, también se comentan aquí también en tanto que tienen repercusión directa en el lenguaje:

a) Sencillez:

C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo:

- El código escrito en C# es auto contenido, lo que significa que no necesita de archivos adicionales al propio fuente tales como archivos de cabecera o archivos IDL.
- El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile (no como en C++), lo que facilita la portabilidad del código.
- No se incluyen elementos poco útiles de lenguajes como C++ tales como macros, herencia múltiple o la necesidad de un operador diferente del punto (.) acceder a miembros de espacios de nombres (::).

b) Modernidad:

C# incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrándose son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular, como un tipo básico decimal que permita realizar operaciones de alta precisión con reales de 128 bits (muy útil en el mundo financiero), la inclusión de una instrucción foreach que permita recorrer colecciones con facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico string para representar cadenas o la distinción de un tipo bool específico para representar valores lógicos.

c) Orientación a objetos:

Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos, aunque eso es más bien una característica del CTS que de C#. Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++ es que el de C# es más puro en tanto que no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código.

C# soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo. En lo referente a la encapsulación es importante señalar que aparte de los típicos modificadores public, private y protected, C# añade un cuarto modificador llamado internal, que puede combinarse con protected e indica que al elemento a cuya definición precede sólo puede accederse desde su mismo ensamblado. Respecto a la herencia -a diferencia de C++ y al igual que Java- C# sólo admite herencia simple de clases ya que la múltiple provoca más quebraderos de cabeza que facilidades y en la mayoría de los casos su utilidad puede ser simulada con facilidad mediante herencia múltiple de interfaces.

De todos modos, esto vuelve a ser más bien una característica propia del CTS que de C#. Por otro lado y a diferencia de Java, en C# se ha optado por hacer que todos los métodos sean por defecto sellados y que los redefinibles hayan de marcarse con el modificador virtual (como en C++), lo que permite evitar errores derivados de redefiniciones accidentales. Además, un efecto secundario de esto es que las llamadas a los métodos serán más eficientes por defecto al no tenerse que buscar en la tabla de funciones virtuales la implementación de los mismos a la que se ha de llamar. Otro efecto secundario es que permite que las llamadas a los métodos virtuales se puedan hacer más eficientemente al contribuir a que el tamaño de dicha tabla se reduzca.

d) Orientación a componentes:

La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros).

Gestión automática de memoria: Todo lenguaje de .NET tiene a su disposición el recolector de basura del CLR. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos.

Sin embargo, dado que la destrucción de los objetos a través del recolector de basura y sólo se realiza cuando éste se active ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en el fuente-, C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción using.

e) Seguridad de tipos:

C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que permite evita que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura. Para ello se toman medidas del tipo:

- Sólo se admiten conversiones entre tipos compatibles. Esto es, entre un tipo y antecesores suyos, entre tipos para los que explícitamente se haya definido un operador de conversión, y entre un tipo y un tipo hijo suyo del que un objeto del primero almacenase una referencia del segundo (downcasting) Obviamente, lo último sólo puede comprobarlo en tiempo de ejecución el CLR y no el compilador, por lo que en realidad el CLR y el compilador colaboran para asegurar la corrección de las conversiones.

- No se pueden usar variables no inicializadas. El compilador da a los campos un valor por defecto consistente en ponerlos a cero y controla mediante análisis del

flujo de control del fuente que no se lea ninguna variable local sin que se le haya asignado previamente algún valor.

- ♣ Se comprueba que todo acceso a los elementos de una tabla se realice con índices que se encuentren dentro del rango de la misma.
- ♣ Se puede controlar la producción de desbordamientos en operaciones aritméticas, informándose de ello con una excepción cuando ocurra. Sin embargo, para conseguirse un mayor rendimiento en la aritmética estas comprobaciones no se hacen por defecto al operar con variables sino sólo con constantes (se pueden detectar en tiempo de compilación).
- ♣ A diferencia de Java, C# incluye delegados, que son similares a los punteros a funciones de C++ pero siguen un enfoque orientado a objetos, pueden almacenar referencias a varios métodos simultáneamente, y se comprueba que los métodos a los que apunten tengan parámetros y valor de retorno del tipo indicado al definirlos.
- ♣ Pueden definirse métodos que admitan un número indefinido de parámetros de un cierto tipo, y a diferencia lenguajes como C/C++, en C# siempre se comprueba que los valores que se les pasen en cada llamada sean de los tipos apropiados.

f) Instrucciones seguras:

Para evitar errores muy comunes, en C# se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes. Por ejemplo, toda condición ha de ser una expresión condicional y no aritmética, con lo que se evitan errores por confusión del operador de igualdad (==) con el de asignación (=); y todo caso de un switch ha de terminar en un break o goto que indique cuál es la siguiente acción a realizar, lo que evita la ejecución accidental de casos y facilita su reordenación.

g) Sistema de tipos unificado:

A diferencia de C++, en C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada System.Object, por lo que dispondrán de todos los miembros definidos en ésta clase (es decir, serán "objetos").

A diferencia de Java, en C# esto también es aplicable a los tipos de datos básicos. Además, para conseguir que ello no tenga una repercusión negativa en su nivel de rendimiento, se ha incluido un mecanismo transparente de boxing y unboxing con el que se consigue que sólo sean tratados como objetos cuando la situación lo requiera, y mientras tanto puede aplicárseles optimizaciones específicas. El hecho de que todos los tipos del lenguaje deriven de una clase común facilita enormemente el diseño de colecciones genéricas que puedan almacenar objetos de cualquier tipo.

h) Extensibilidad de tipos básicos:

C# permite definir, a través de estructuras, tipos de datos para los que se apliquen las mismas optimizaciones que para los tipos de datos básicos. Es decir, que se puedan almacenar directamente en pila (luego su creación, destrucción y acceso serán más rápidos) y se asignen por valor y no por referencia. Para conseguir que lo último no tenga efectos negativos al pasar estructuras como parámetros de métodos, se da la posibilidad de pasar referencias a pila a través del modificador de parámetro ref.

i) Extensibilidad de operadores:

Para facilitar la legibilidad del código y conseguir que los nuevos tipos de datos básicos que se definan a través de las estructuras estén al mismo nivel que los básicos predefinidos en el lenguaje, al igual que C++ y a diferencia de Java, C# permite redefinir el significado de la mayoría de los operadores -incluidos los de conversión, tanto para conversiones implícitas como explícitas- cuando se apliquen a diferentes tipos de objetos.

Las redefiniciones de operadores se hacen de manera inteligente, de modo que a partir de una única definición de los operadores ++ y -- el compilador puede deducir automáticamente como ejecutarlos de manera prefijas y postfija; y definiendo operadores simples (como +), el compilador deduce cómo aplicar su versión de asignación compuesta (+=) Además, para asegurar la consistencia, el compilador vigila que los operadores con opuesto siempre se redefinan por parejas (por ejemplo, si se redefine ==, también hay que redefinir !=), También se da la posibilidad, a través del concepto de indizador, de redefinir el significado del operador [] para los tipos de dato definidos por el usuario, con lo que se consigue que se pueda acceder al mismo como si fuese una tabla. Esto es muy útil para trabajar con tipos que actúen como colecciones de objetos.

j) Extensibilidad de modificadores:

C# ofrece, a través del concepto de atributos, la posibilidad de añadir a los meta datos del módulo resultante de la compilación de cualquier fuente información adicional a la generada por el compilador que luego podrá ser consultada en tiempo ejecución a través de la librería de reflexión de .NET. Esto, que más bien es una característica propia de la plataforma .NET y no de C#, puede usarse como un mecanismo para definir nuevos modificadores.

k) Versionable:

C# incluye una política de versionado que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoquen errores difíciles de detectar en tipos hijos previamente desarrollados y ya extendidos con miembros de igual nombre a los recién introducidos.

Si una clase introduce un nuevo método cuyas redefiniciones deban seguir la regla de llamar a la versión de su padre en algún punto de su código, difícilmente seguirían esta regla miembros de su misma signatura definidos en clases hijas previamente a la definición del mismo en la clase padre; o si introduce un nuevo campo con el mismo nombre que algún método de una clase hija, la clase hija dejará de funcionar. Para evitar que esto ocurra, en C# se toman dos medidas:

- Se obliga a que toda redefinición deba incluir el modificador override, con lo que la versión de la clase hija nunca sería considerada como una redefinición de la versión de miembro en la clase padre ya que no incluiría override. Para evitar que por accidente un programador incluya este modificador, sólo se permite incluirlo en miembros que tengan la misma signatura que miembros marcados como redefinibles mediante el modificador virtual. Así además se evita el error tan frecuente en Java de creerse haber redefinido un miembro, pues si el miembro con override no existe en la clase padre se producirá un error de compilación.

- Si no se considera redefinición, entonces se considera que lo que se desea es ocultar el método de la clase padre, de modo que para la clase hija sea como si nunca hubiese existido. El compilador avisará de esta decisión a través de un mensaje de aviso que puede suprimirse incluyendo el modificador new en la definición del miembro en la clase hija para así indicarle explícitamente la intención de ocultación.

l) Eficiente:

En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador unsafe) y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad procesamiento muy grandes.

m) Compatible:

Para facilitar la migración de programadores, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes, sino que el CLR también ofrece, a través de los llamados Platform Invocation Services (PInvoke), la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como las DLLs de la API Win32. Nótese que la capacidad de usar punteros en código inseguro permite que se pueda acceder con facilidad a este tipo de funciones, ya que éstas muchas veces esperan recibir o devuelven punteros.

También es posible acceder desde código escrito en C# a objetos COM. Para facilitar esto, el .NET Framework SDK incluye una herramientas llamadas tlbimp y regasm mediante las que es posible generar automáticamente clases proxy que permitan, respectivamente, usar objetos COM desde .NET como si de objetos .NET se tratase y registrar objetos .NET para su uso desde COM. Finalmente, también se da la posibilidad de usar controles ActiveX desde código .NET y viceversa. Para lo primero se utiliza la utilidad aximp, mientras que para lo segundo se usa la ya mencionada regasm.

15.3 Escritura de aplicaciones.

Aplicación básica ¡Hola Mundo!

Básicamente una aplicación en C# puede verse como un conjunto de uno o más archivos de código fuente con las instrucciones necesarias para que la aplicación funcione como se desea y que son pasados al compilador para que genere un ejecutable. Cada uno de estos archivos no es más que un archivo de texto plano escrito usando caracteres Unicode y siguiendo la sintaxis propia de C#.

Como primer contacto con el lenguaje, nada mejor que el típico programa de iniciación "¡Hola Mundo!" que lo único que hace al ejecutarse es mostrar por pantalla el mensaje ¡Hola Mundo! Su código es[1]:

```
1: class HolaMundo
2: {
3:       static void Main()
4:       {
5:             System.Console.WriteLine("¡Hola Mundo!");
6:       }
7: }
```

Todo el código escrito en C# se ha de escribir dentro de una definición de clase, y lo que en la línea 1: se dice es que se va a definir una clase (class) de nombre HolaMundo cuya definición estará comprendida entre la llave de apertura de la línea 2: y su correspondiente llave de cierre en la línea 7: Dentro de la definición de la clase (línea 3:) se define un método de nombre Main cuyo código es el indicado entre la llave de apertura de la línea 4: y su respectiva llave de cierre (línea 6:) Un método no es más que un conjunto de instrucciones a las que se les asocia un nombre, de modo que para posteriormente ejecutarlas baste referenciarlas por su nombre en vez de tener que rescribirlas.

[1] Los números de la línea no forman parte del código, sino que solo se incluyen para facilitar su posterior explicación.

La partícula que antecede al nombre del método indica cuál es el tipo de valor que se devuelve tras la ejecución del método, y en este caso es void que significa que no se devuelve nada. Por su parte, los paréntesis que se colocado tras el nombre del método indican cuáles son los parámetros éste toma, y como en este caso están vacíos ello significa que el método no toma parámetros. Los parámetros de un método permiten variar el resultado de su ejecución según los valores que se les dé en cada llamada. La palabra static que antecede a la declaración del tipo de valor devuelto es un modificador del significado de la declaración de método que indica que el método está asociado a la clase dentro de la que se define y no a los objetos que se creen a partir de ella. Main() es lo que es denomina el punto de entrada de la aplicación, que no es más que el método por el que comenzará su ejecución. Necesita del modificador static para evitar que para llamarlo haya que crear algún objeto de la clase donde se haya definido.

Finalmente, la línea 5: contiene la instrucción con el código a ejecutar, que lo que se hace es solicitar la ejecución del método WriteLine() de la clase Console definida en el espacio de nombres System pasándole como parámetro la cadena de texto con el contenido ¡Hola Mundo! Nótese que las cadenas de textos son secuencias de caracteres delimitadas por comillas dobles aunque dichas comillas no forman parte de la cadena. Por su parte, un espacio de nombres puede considerarse que es algo similar para las clases a lo que un directorio es para los archivos; es decir, es una forma de agruparlas.

El método WriteLine() se usará muy a menudo en los próximos temas, por lo que es conveniente señalar ahora que una forma de llamarlo que se utilizará en repetidas ocasiones consiste en pasarle un número indefinido de otros parámetros de cualquier tipo e incluir en el primero subcadenas de la forma {i}. Con ello se consigue que se muestre por la ventana de consola la cadena que se le pasa como primer parámetro pero sustituyéndole las subcadenas {i} por el valor convertido en cadena de texto del parámetro que ocupe la posición i+2 en la llamada a WriteLine(). Por ejemplo, la siguiente instrucción mostraría:

Tengo 5 años por pantalla si x valiese 5:
System.Console.WriteLine("Tengo {0} años", x);
Para indicar cómo convertir cada objeto en una cadena de texto basta redefinir su método ToString().

Antes de seguir es importante resaltar que C# es sensible a las mayúsculas, los que significa que no da igual la capitalización con la que se escriban los identificadores. Es decir, no es lo mismo escribir Console que COnsole o CONSOLE, y si se hace de alguna de las dos últimas formas el compilador producirá un error debido a que en el espacio de nombres System no existe ninguna clase con dichos nombres. En este sentido, cabe señalar que un error común entre programadores acostumbrados a Java es llamar al punto de entrada main en vez de Main, lo que provoca un error al compilar ejecutables en tanto que el compilador no detectará ninguna definición de punto de entrada.

Puntos de entrada.

Ya se ha dicho que el punto de entrada de una aplicación es un método de nombre Main que contendrá el código por donde se ha de iniciar la ejecución de la misma. Hasta ahora sólo se ha visto una versión de Main() que no toma parámetros y tiene como tipo de retorno void, pero en realidad todas sus posibles versiones son:

static void Main()
static int Main()
static int Main(string[] args)
static void Main(string[] args)

Como se ve, hay versiones de Main() que devuelven un valor de tipo int. Un int no es más que un tipo de datos capaz de almacenar valor enteros comprendidos entre – 2.1471483.648 y 2.1471483.647, y el número devuelto por Main() sería interpretado como código de retorno de la aplicación. Éste valor suele usarse para indicar si la aplicación a terminado con éxito (generalmente valor 0) o no (valor según la causa de la terminación anormal).

También hay versiones de Main() que toman un parámetro donde se almacenará la lista de argumentos con los que se llamó a la aplicación, por lo que sólo es útil usar estas versiones del punto de entrada si la aplicación va a utilizar dichos argumentos para algo. El tipo de este parámetro es string[], lo que significa que es una tabla de cadenas de texto, y su nombre -que es el que habrá de usarse dentro del código de Main() para hacerle referencia es args en el ejemplo, aunque podría dársele cualquier otro.

15.4 Compilación en línea de comandos.

Una vez escrito el código anterior con algún editor de textos –como el Bloc de Notas de Windows- y almacenado en formato de texto plano en un archivo HolaMundo.cs[2], para compilarlo basta abrir una ventana de consola (MS-DOS en Windows), colocarse en el directorio donde se encuentre y pasárselo como parámetro al compilador así:

csc HolaMundo.cs.

csc.exe es el compilador de C# incluido en el .NET Framework SDK para Windows de Microsoft, y es posible llamarlo desde cualquier directorio en tanto que al instalarlo se añade una referencia al mismo en el path. Si utiliza otros compiladores de C# puede que varié la forma en que se realice la compilación, por lo que lo que aquí se explica en principio sólo podría ser válido para el compilador de Microsoft para Windows.

[2] El nombre que se dé al archivo puede ser cualquiera, aunque se recomienda darle la extensión **.cs** ya que es la utilizada por convenio.

Tras la compilación se obtendría un ejecutable llamado HolaMundo.exe cuya ejecución produciría la siguiente salida por la ventana de consola:

¡Hola Mundo!

Si la aplicación que se vaya a compilar no utilizase la ventana de consola para mostrar su salida sino una interfaz gráfica de ventanas, entonces habría que compilarla pasando al compilador la opción /t con el valor winexe antes del nombre del archivo a compilar. Si no se hiciese así se abriría la ventana de consola cada vez que ejecutase la aplicación de ventanas, lo que suele ser indeseable en este tipo de aplicaciones. Así, para compilar Ventanas.cs como ejecutable de ventanas sería conveniente escribir:

csc /t:winexe Ventanas.cs

Nótese que aunque el nombre winexe dé la sensación de que este valor para la opción /t sólo permite generar ejecutables de ventanas, en realidad lo que permite es generar ejecutables sin ventana de consola asociada. Por tanto, también puede usarse para generar ejecutables que no tengan ninguna interfaz asociada, ni de consola ni gráfica.

Si en lugar de un ejecutable -ya sea de consola o de ventanas- se desea obtener una librería, entonces al compilar hay que pasar al compilador la opción /t con el valor library. Por ejemplo, siguiendo con el ejemplo inicial habría que escribir:

csc /t:library HolaMundo.cs

En este caso se generaría un archivo HolaMundo.dll cuyos tipos de datos podrían utilizarse desde otros fuentes pasando al compilador una referencia a los mismos mediante la .cs ya que es la utilizada por convenio opción /r. Por ejemplo, para compilar como ejecutable un fuente A.cs que use la clase HolaMundo de la librería HolaMundo.dll se escribiría:

csc /r:HolaMundo.dll A.cs

En general /r permite referenciar a tipos definidos en cualquier ensamblado, por lo que el valor que se le indique también puede ser el nombre de un ejecutable. Además, en cada compilación es posible referenciar múltiples ensamblados ya sea incluyendo la opción /r una vez por cada uno o incluyendo múltiples referencias en una única opción /r usando comas o puntos y comas como separadores. Por ejemplo, las siguientes tres llamadas al compilador son equivalentes:

csc /r:HolaMundo.dll;Otro.dll;OtroMás.exe A.cs
csc /r:HolaMundo.dll,Otro.dll,OtroMás.exe A.cs
csc /t:HolaMundo.dll /r:Otro.dll /r:OtroMás.exe A.cs

Hay que señalar que aunque no se indique nada, en toda compilación siempre se referencia por defecto a la librería mscorlib.dll de la BCL, que incluye los tipos de uso más frecuente. Si se usan tipos de la BCL no incluidos en ella habrá que incluir al compilar referencias a las librerías donde estén definidos (en la documentación del SDK sobre cada tipo de la BCL puede encontrar información sobre donde se definió). Tanto las librerías como los ejecutables son ensamblados. Para generar un módulo de código que no forme parte de ningún ensamblado sino que contenga definiciones de tipos que puedan añadirse a ensamblados que se compilen posteriormente, el valor que ha de darse al compilar a la opción /t es module. Por ejemplo:

csc /t:module HolaMundo.cs

Con la instrucción anterior se generaría un módulo llamado HolaMundo.netmodule que podría ser añadido a compilaciones de ensamblados incluyéndolo como valor de la opción /addmodule. Por ejemplo, para añadir el módulo anterior a la compilación del fuente librería Lib.cs como librería se escribiría:

csc /t:library /addmodule:HolaMundo.netmodule Lib.cs

Aunque hasta ahora todas las compilaciones de ejemplo se han realizado utilizando un único archivo de código fuente, en realidad nada impide que se puedan utilizar más. Por ejemplo, para compilar los archivos A.cs y B.cs en una librería A.dll se ejecutaría:

csc /t:library A.cs B.cs

Nótese que el nombre que por defecto se dé al ejecutable generado siempre es igual al del primer fuente especificado pero con la extensión propia del tipo de compilación realizada (.exe para ejecutables, .dll para librerías y .netmodule para módulos) Sin embargo, puede especificarse como valor en la opción /out del compilador cualquier otro tal y como muestra el siguiente ejemplo que compila el archivo A.cs como una librería de nombre Lib.exe:

csc /t:library /out:Lib.exe A.cs

Véase que aunque se haya dado un nombre terminado en .exe al archivo resultante, éste sigue siendo una librería y no un ejecutable e intentar ejecutarlo produciría un mensaje de error. Obviamente no tiene mucho sentido darle esa extensión, y sólo se le ha dado en este ejemplo para demostrar que, aunque recomendable, la extensión del archivo no tiene porqué corresponderse realmente con el tipo de archivo del que se trate.

A la hora de especificar archivos a compilar también es pueden utilizar los caracteres de comodín típicos del sistema operativo. Por ejemplo, para compilar todos los archivos con extensión .cs del directorio actual en una librería llamada Varios.dll se haría:

*csc /t:library /out:varios.dll *.cs*

Con lo que hay que tener cuidado, y en especial al compilar varios fuentes, es con que no se compilen a la vez más de un tipo de dato con punto de entrada, pues entonces el compilador no sabría cuál usar como inicio de la aplicación. Para orientarlo, puede especificarse como valor de la opción /main el nombre del tipo que contenga el Main() ha usar como punto de entrada. Así, para compilar los archivos A.cs y B.cs en un ejecutable cuyo punto de entrada sea el definido en el tipo Principal, habría que escribir:

csc /main:Principal A.cs B.cs

Obviamente, para que esto funcione A.cs o B.cs tiene que contener alguna definición de algún tipo llamado Principal con un único método válido como punto de entrada. (obviamente si contiene varias se volvería a tener el problema de no saber cuál usar).

15.5 Compilación con Visual Studio.NET

Para compilar una aplicación en Visual Studio.NET primero hay que incluirla dentro de algún proyecto. Para ello basta pulsar el botón New Project en la página de inicio que se muestra nada más arrancar dicha herramienta, tras lo que se obtendrá una pantalla con el aspecto mostrado en la Ilustración 1.

En el recuadro de la ventana mostrada etiquetado como Project Types se ha de seleccionar el tipo de proyecto a crear. Obviamente, si se va a trabajar en C# la opción que habrá que escoger en la misma será siempre Visual C# Projects. En el recuadro Templates se ha de seleccionar la plantilla correspondiente al subtipo de proyecto dentro del tipo indicado en Project Types que se va a realizar.

Para realizar un ejecutable de consola, como en este caso, hay que seleccionar el icono etiquetado como Console Application. Si se quisiese realizar una librería habría que seleccionar Class Library, y si se quisiese realizar un ejecutable de ventanas habría que seleccionar Windows Application. Nótese que no se ofrece ninguna plantilla para realizar módulos, lo que se debe a que desde Visual Studio.NET no pueden crearse.

Por último, en el recuadro de texto Name se ha de escribir el nombre a dar al proyecto y en Location el del directorio base asociado al mismo. Nótese que bajo de Location aparecerá un mensaje informando sobre cual será el directorio donde finalmente se almacenarán los archivos del proyecto, que será el resultante de concatenar la ruta especificada para el directorio base y el nombre del proyecto.

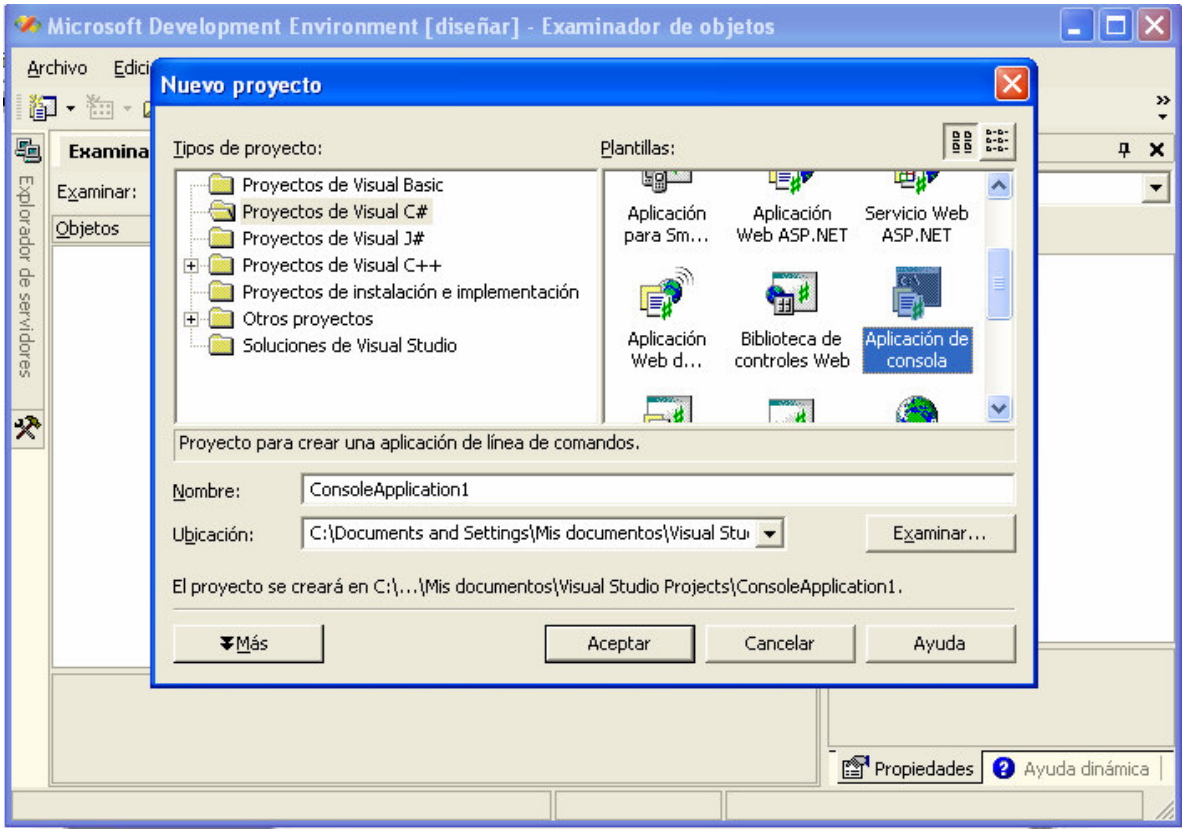


Figura 34. Ventaja de creación de nuevo proyecto en Visual Studio.NET.

Una vez configuradas todas estas opciones, al pulsar botón OK Visual Studio creará toda la infraestructura adecuada para empezar a trabajar cómodamente en el proyecto. Como puede apreciarse en la Figura 35, esta infraestructura consistirá en la generación de un fuente que servirá de plantilla para la realización de proyectos del tipo elegido (en este caso, aplicaciones de consola en C#):

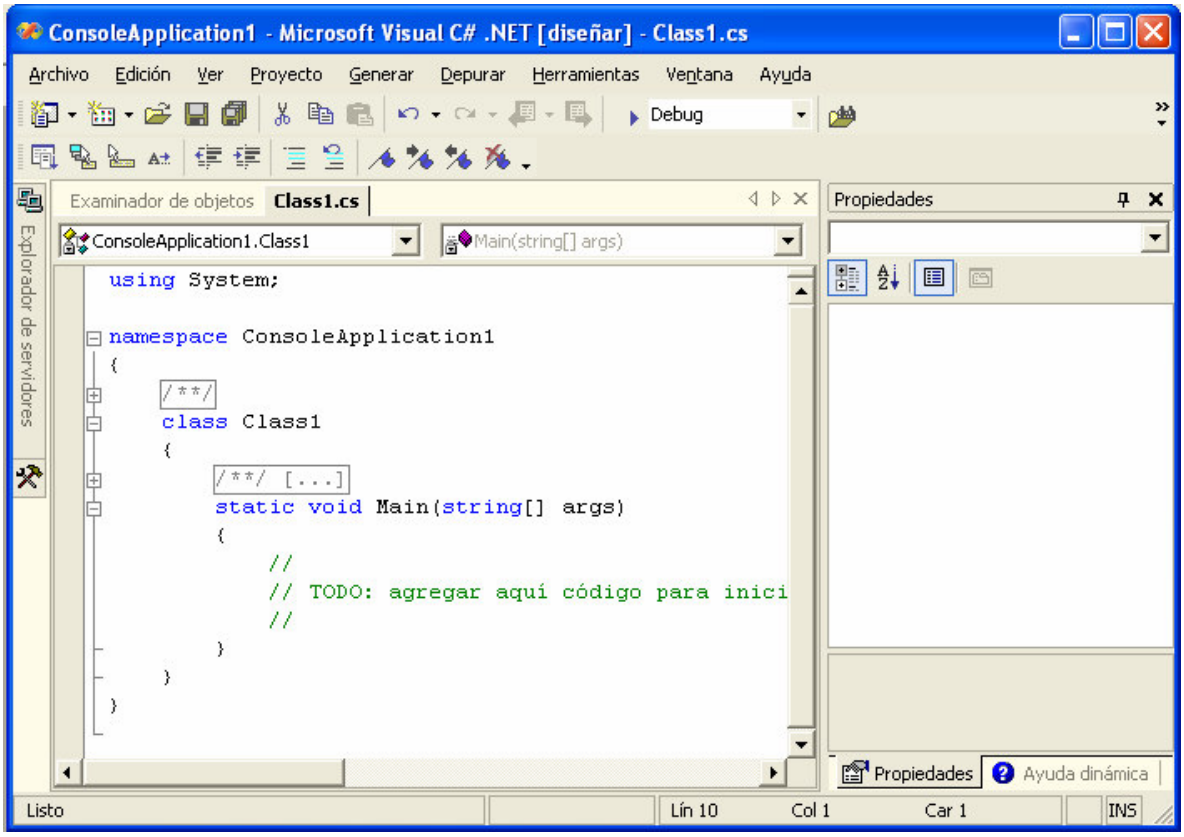


Figura 35. Plantilla para aplicaciones de consola generada por Visual Studio.NET

A partir de esta plantilla, escribir el código de la aplicación de ejemplo es tan sencillo con simplemente digitar System.Console.WriteLine("¡Hola Mundo!") dentro de la definición del método Main() creada por Visual Studio.NET. Claro está, otra posibilidad es borrar toda la plantilla y sustituirla por el código para HolaMundo mostrado anteriormente.

Sea haga como se haga, para compilar y ejecutar tras ello la aplicación sólo hay que pulsar CTRL+F5 o seleccionar Debug - Start Without Debugging en el menú principal de Visual Studio.NET. Para sólo compilar el proyecto, entonces hay que seleccionar Build - Rebuild All. De todas formas, en ambos casos el ejecutable generado se almacenará en el subdirectorio Bin\Debug del directorio del proyecto.

En el extremo derecho de la ventana principal de Visual Studio.NET puede encontrar el denominado Solution Explorer (si no lo encuentra, seleccione View – Solution Explorer), que es una herramienta que permite consultar cuáles son los archivos que forman el proyecto.

Si selecciona en él el icono correspondiente al proyecto en que esta trabajando y pulsa View - Property Pages obtendrá una hoja de propiedades del proyecto con el aspecto mostrado:

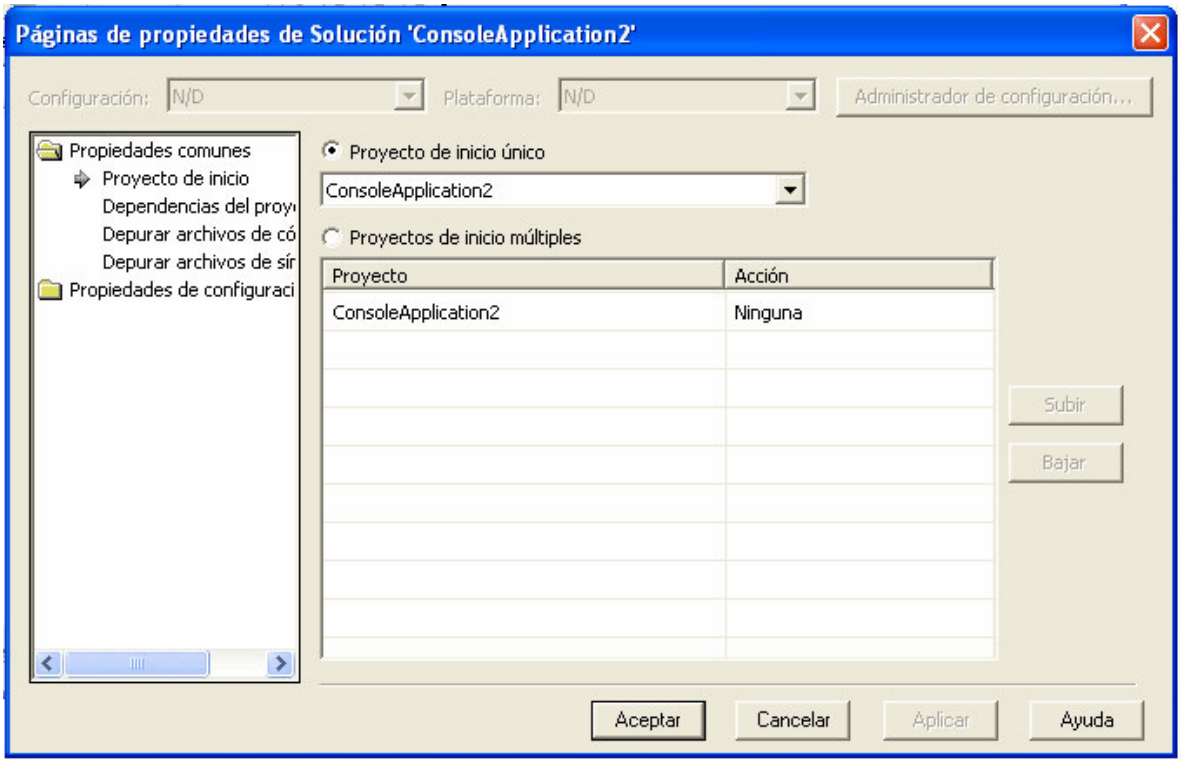


Figura 36. Hoja de propiedades del proyecto en Visual Studio .NET

Esta ventana permite configurar de manera visual la mayoría de opciones con las que se llamará al compilador en línea de comandos. Por ejemplo, para cambiar el nombre del archivo de salida (opción /out) se indica su nuevo nombre en el cuadro de texto Common Properties - General - Assembly Name; para cambiar el tipo de proyecto a generar (opción /t) se utiliza Common Properties - General - Output Type (como verá si intenta cambiarlo, no es posible generar módulos desde Visual Studio.NET); y el tipo que contiene el punto de entrada a utilizar (opción /main) se indica en Common Properties – General - Startup Object Finalmente, para añadir al proyecto referencias a ensamblados externos (opción /r) basta seleccionar Project - Add Reference en el menú principal de VS.NET.

15.6 El preprocesador.

Concepto de preprocesador.

El preprocesado es un paso previo[3] a la compilación mediante el que es posible controlar la forma en que se realizará ésta. El preprocesador es el módulo auxiliar que utiliza el compilador para realizar estas tareas, y lo que finalmente el compilador compila es el resultado de aplicar el preprocesador al archivo de texto fuente, resultado que también es un archivo de texto. Nótese pues, que mientras que el compilador hace una traducción de texto a binario, lo que el preprocesador hace es una traducción de texto a texto.

Aquellos que tengan experiencia en el uso del preprocesador en lenguajes como C++ y conozcan los problemas que implica el uso del mismo pueden respirar tranquilos, ya que en C# se han eliminado la mayoría de características de éste que provocaban errores difíciles de detectar (macros, directivas de inclusión, etc.) y prácticamente sólo se usa para permitir realizar compilaciones condicionales de código.

Directivas de preprocesado.

CONCEPTO DE DIRECTIVA. SINTAXIS

El preprocesador no interpreta de ninguna manera el código fuente del archivo, sino que sólo interpreta de dicho archivo lo que se denominan directivas de preprocesado. Estas directivas son líneas de texto del archivo fuente que se caracterizan porque en ellas el primer carácter no blanco que aparece es una almohadilla (carácter #) Por ejemplo:

```
#define TEST
#error Ha habido un error fatal
```

No se preocupe ahora si no entiende el significado de estas directivas. Lo único debe saber es que el nombre que se indica tras el símbolo # es el nombre de la directiva, y el texto que se incluye tras él (no todas las directivas tienen porqué incluirlo) es el valor que se le da. Por tanto, la sintaxis de una directiva es:

#<nombreDirectiva> <valorDirectiva>

Es posible incluir comentarios en la misma línea en que se declara una directiva, aunque estos sólo pueden ser comentarios de una línea que empiecen con // Por ejemplo, el siguiente comentario es válido:

```
#define TEST // Ha habido algún error durante el preprocesado
```

[3] En realidad, en C# se realiza a la vez que el análisis léxico del código fuente; pero para simplificar la explicación considere que se realiza antes que éste, en una etapa previa independiente.

Pero este otro no, pues aunque ocupa una línea tiene la sintaxis de los comentarios que pueden ocupar varias líneas:

#define TEST / Ha habido algún error durante el preprocesado */*

Definición de identificadores de preprocesado.

Como ya se ha comentado, la principal utilidad del preprocesador en C# es la de permitir determinar cuáles regiones de código de un archivo fuente se han de compilar. Para ello, lo que se hace es encerrar las secciones de código opcionales dentro de directivas de compilación condicional, de modo que sólo se compilarán si determinados identificadores de preprocesado están definidos. Para definir un identificador de este tipo la directiva que se usa sigue esta sintaxis:

#define <nombreIdentificador>

Esta directiva define un identificador de preprocesado <nombreIdentificador>. Son válidos aquellos formados por uno o más caracteres alfanuméricos tales que no sean ni true ni false y no empiecen con un numero. Por ejemplo, para definir un identificador de preprocesado de nombre PRUEBA se haría:

#define PRUEBA

Por convenio se da a estos identificadores nombres en los que todas las letras se escriben en mayúsculas, como en el ejemplo anterior. Aunque es sólo un convenio y nada obliga a usarlo, ésta será la nomenclatura que usara el presente documento, que es la usada por Microsoft en sus códigos de ejemplo. Conviene familiarizarse con ella porque por un lado hay mucho código escrito que la usa y por otro usarla facilita la lectura de nuestro código a los demás al ser la notación que esperarán encontrar.

Es importante señalar que cualquier definición de identificador ha de preceder a cualquier aparición de código en el archivo fuente. Por ejemplo, el siguiente código no es válido, pues antes del #define se ha incluido código fuente (el class A):

```
class A
#define PRUEBA
{}
```

Sin embargo, aunque no pueda haber código antes de un #define, sí que es posible incluir antes de él otras directivas de preprocesado con total libertad. Existe una forma alternativa de definir un identificador de preprocesado y que además permite que dicha definición sólo sea válida en una compilación en concreto.

Esta forma consiste en pasarle al compilador en su llamada la opción /d:<nombreIdentificador> (forma abreviada de /define:<nombreIdentificador>), caso en que durante la compilación se considerará que al principio de todos los archivos fuente a compilar se encuentra definido el identificador indicado. Las siguientes tres formas de llamar al compilador son equivalentes y definen identificadores de preprocesado de nombres PRUEBA y TRAZA durante la compilación de un archivo fuente de nombre ejemplo.cs:

csc /d:PRUEBA /d:TRAZA ejemplo.cs
csc /d:PRUEBA,TRAZA ejemplo.cs
csc /d:PRUEBA;TRAZA ejemplo.cs

Nótese en el ejemplo que si queremos definir más de un identificador usando esta técnica tenemos dos alternativas: incluir varias opciones /d en la llamada al compilador o definir varios de estos identificadores en una misma opción /d separándolos mediante caracteres de coma (,) o punto y coma (;).

Si se trabaja con Visual Studio.NET en lugar de directamente con el compilador en línea de comandos, entonces puede conseguir mismo efecto a través de View - Property Pages - Configuration Options - Build - Conditional Compilation Constants, donde nuevamente usado el punto y coma (;) o la coma (,) como separadores, puede definir varias constantes. Para que todo funcione bien, antes de seleccionar View ha de seleccionar en el Solution Explorer (se abre con View - Solution Explorer) el proyecto al que aplicar la definición de las constantes.

Finalmente, respecto al uso de #define sólo queda comentar que es posible definir varias veces una misma directiva sin que ello provoque ningún tipo de error en el compilador, lo que permite que podamos pasar tantos valores a la opción /d del compilador como queramos sin temor a que entren en conflicto con identificadores de preprocesado ya incluidos en los fuentes a compilar.

Eliminación de identificadores de preprocesado.

Del mismo modo que es posible definir identificadores de preprocesado, también es posible eliminar definiciones de este tipo de identificadores previamente realizadas. Para ello la directiva que se usa tiene la siguiente sintaxis:

#undef <nombreIdentificador>

En caso de que se intente eliminar con esta directiva un identificador que no haya sido definido o cuya definición ya haya sido eliminada no se producirá error alguna, sino que simplemente la directiva de eliminación será ignorada. El siguiente ejemplo muestra un ejemplo de esto en el que el segundo #undef es ignorado:

```
#define VERSION1
#undef VERSION1
#undef VERSION1
```

Al igual que ocurría con las directivas #define, no se puede incluir código fuente antes de las directivas #undef, sino que, todo lo más, lo único que podrían incluirse antes que ellas serían directivas de preprocesado.

Compilación condicional

Como se ha repetido varias veces, la principal utilidad del preprocesador en C# es la de permitir la compilación de código condicional, lo que consiste en sólo permitir que se compile determinadas regiones de código fuente si las variables de preprocesado definidas cumplen alguna condición determinada. Para conseguir esto se utiliza el siguiente juego de directivas:

```
#if <condición1>
        <código1>
#elif <condición2>
        <código2>
...
#else
        <códigoElse>
#endif
```

El significado de una estructura como esta es que si se cumple <condición1> entonces se pasa al compilador el <código1>, si no ocurre esto pero se cumple <condición2> entonces lo que se pasaría al compilador sería <código2>, y así continuamente hasta que se llegue a una rama #elif cuya condición se cumpla. Si no se cumple ninguna pero hay una rama #else se pasará al compilador el <códigoElse>, pero si dicha rama no existiese entonces no se le pasaría código alguno y se continuaría preprocesando el código siguiente al #endif en el fuente original.

Aunque las ramas #else y #endif son opcionales, hay que tener cuidado y no mezclarlas ya que la rama #else sólo puede aparecer como última rama del bloque #if...#endif. Es posible anidar varias estructuras #if...#endif, como muestra el siguiente código:

```
#define PRUEBA
using System;
class A
{
        public static void Main()
        {
                #if PRUEBA
                        Console.Write ('Esto es una prueba');
                                #if TRAZA
```

```
                    Console.Write("con traza");
                #elif !TRAZA
                    Console.Write("sin traza");
                #endif
            #endif
        }
}
```

Como se ve en el ejemplo, las condiciones especificadas son nombres de identificadores de preprocesado, considerándose que cada condición sólo se cumple si el identificador que se indica en ella está definido. O lo que es lo mismo: un identificador de preprocesado vale cierto (true en C#) si está definido y falso (false en C#) si no.

El símbolo ! incluido en junto al valor de la directiva #elif es el símbolo de "no" lógico, y el #elif en el que se usa lo que nos permite es indicar que en caso de que no se encuentre definido el identificador de preprocesado TRAZA se han de pasar al compilador las instrucciones a continuación indicadas (o sea, el Console.Write("sin traza");). El código fuente que el preprocesador pasará al compilador en caso de que compile sin especificar ninguna opción /d en la llamada al compilador será:

```
using System;
class A
{
        public static void Main()
        {
            Console.Write("Esto es una prueba");
            Console.Write("sin traza");
        }
}
```

Nótese como en el código que se pasa al compilador ya no aparece ninguna directiva de preprocesado, pues lo que el preprocesador le pasa es el código resultante de aplicar al original las directivas de preprocesado que contuviese.

Asimismo, si compilase el código fuente original llamando al compilador con /d:TRAZA, lo que el preprocesador pasaría al compilador sería:

```
using System;
class A
{
        public static void Main()
        {
            Console.Write ("Esto es una prueba");
            Console.Write("sin traza");
        }
}
```

Hasta ahora solo ha visto que la condición de un #if o #elif puede ser un identificador de preprocesado, y que este valdrá true o false según esté o no definido. Pues bien, estos no son el único tipo de condiciones válidas en C#, sino que también es posible incluir condiciones que contengan expresiones lógicas formadas por identificadores de preprocesado, operadores lógicos (! para "not", && para "and" y || para "or"), ope radores relacionales de igualdad (==) y desigualdad (!=), paréntesis ((y)) y los identificadores especiales true y false. Por ejemplo:

```
#if TRAZA // Se cumple si TRAZA esta definido.
#if TRAZA==true // Idem al ejemplo anterior aunque con una sintaxis menos cómoda
#if !TRAZA // Sólo se cumple si TRAZA no está definido.
#if TRAZA==false // Idema al ejemplo anterior aunque con una sintaxis menos cómoda
#if TRAZA == PRUEBA // Solo se cumple si tanto TRAZA como PRUEBA están
// definidos o si no ninguno lo está.
#if TRAZA != PRUEBA // Solo se cumple si TRAZA esta definido y PRUEBA no o
// viceversa
#if TRAZA && PRUEBA // Solo se cumple si están definidos TRAZA y PRUEBA.
#if TRAZA || PRUEBA // Solo se cumple si están definidos TRAZA o PRUEBA.
#if false // Nunca se cumple (por lo que es absurdo ponerlo)
#if true // Siempre se cumple (por lo que es absurdo ponerlo)
```

Es fácil ver que la causa de la restricción antes comentada de que no es válido dar un como nombre true o false a un identificador de preprocesado se debe al significado especial que estos tienen en las condiciones de los #if y #elif

Generación de avisos y errores.

El preprocesador de C# también ofrece directivas que permiten generar avisos y errores durante el proceso de preprocesado en caso de que ser interpretadas por el preprocesador. Estas directivas tienen la siguiente sintaxis:

```
#warning <mensajeAviso>
#error <mensajeError>
```

La directiva #warning lo que hace al ser procesada es provocar que el compilador produzca un mensaje de aviso que siga el formato estándar usado por éste para ello y cuyo texto descriptivo tenga el contenido indicado en <mensajeAviso>; y #error hace lo mismo pero provocando un mensaje de error en vez de uno de aviso.

Usando directivas de compilación condicional se puede controlar cuando se han de producir estos mensajes, cuando se han de procesar estas directivas. De hecho la principal utilidad de estas directivas es permitir controlar errores de asignación de valores a los diferentes identificadores de preprocesado de un código, y un ejemplo de ello es el siguiente:

```
#warning Código aun no revisado
#define PRUEBA
#if PRUEBA && FINAL
#error Un código no puede ser simultáneamente de prueba y versión final
#endif
class A
{}
```

En este código siempre se producirá el mensaje de aviso, pero el #if indica que sólo se producirá el mensaje de error si se han definido simultáneamente los identificadores de preprocesado PRUEBA y FINAL.

Como puede deducirse del ejemplo, el preprocesador de C# considera que los mensajes asociados a directivas #warning o #error son todo el texto que se encuentra tras el nombre de dichas directivas y hasta el final de la línea donde éstas aparecen. Por tanto, todo comentario que se incluya en una línea de este tipo será considerado como parte del mensaje a mostrar, y no como comentario como tal. Por ejemplo, ante la directiva:

```
#error La compilación ha fallado // Error
```

Lo que se mostrará en pantalla es un mensaje de la siguiente forma:

```
archivo.cs(3,5): error CS1029: La compilación ha fallado // Error
```

Cambios en la numeración de líneas.

Por defecto el compilador enumera las líneas de cada archivo fuente según el orden normal en que estas aparecen en el mismo, y este orden es el que sigue a la hora de informar de errores o de avisos durante la compilación. Sin embargo, hay situaciones en las que interesa cambiar esta numeración, y para ello se ofrece una directiva con la siguiente sintaxis:
#line <número> "<nombrearchivo>".

Esta directiva indica al preprocesador que ha de considerar que la siguiente línea del archivo fuente en que aparece es la línea cuyo número se le indica, independientemente del valor que tuviese según la numeración usada en ese momento. El valor indicado en "<nombrearchivo>" es opcional, y en caso de aparecer indica el nombre que se ha de considerar que tiene el archivo a la hora de dar mensajes de error. Un ejemplo:

#line 127 "csmace.cs"

Este uso de #line indica que el compilador ha de considerar que la línea siguiente es la línea 127 del archivo csmace.cs. A partir de ella se seguirá usando el sistema de numeración normal (la siguiente a esa será la 128 de csmace.cs, la próxima la 129, etc.) salvo que más adelante se vuelva a cambiar la numeración con otra directiva #line.

Aunque en principio puede parecer que esta directiva es de escasa utilidad, lo cierto es que suele venir bastante bien para la escritura de compiladores y otras herramientas que generen código en C# a partir de código escrito en otros lenguajes.

Marcación de regiones de código.

Es posible marcar regiones de código y asociarles un nombre usando el juego de directivas #region y #endregion. Estas directivas se usan así:

#region <nombreRegión>
<código>
#endregion

La utilidad que se dé a estas marcaciones depende de cada herramienta, pero en el momento de escribir estas líneas la única herramienta disponible que hacía uso de ellas era Visual Studio.NET, donde se usa para marcar código de modo que desde la ventana de código podamos expandirlo y contraerlo con una única pulsación de ratón.

En concreto, en la ventana de código de Visual Studio aparecerá un símbolo [-] junto a las regiones de código así marcadas de manera que pulsando sobre él todo el código contenido en la región se comprimirá y será sustituido por el nombre dado en <nombreRegión>.

Tras ello, el [-] se convertirá en un [+] y si volvemos a pulsarlo el código contraído se expandirá y recuperará su aspecto original. A continuación se muestra un ejemplo de cada caso:

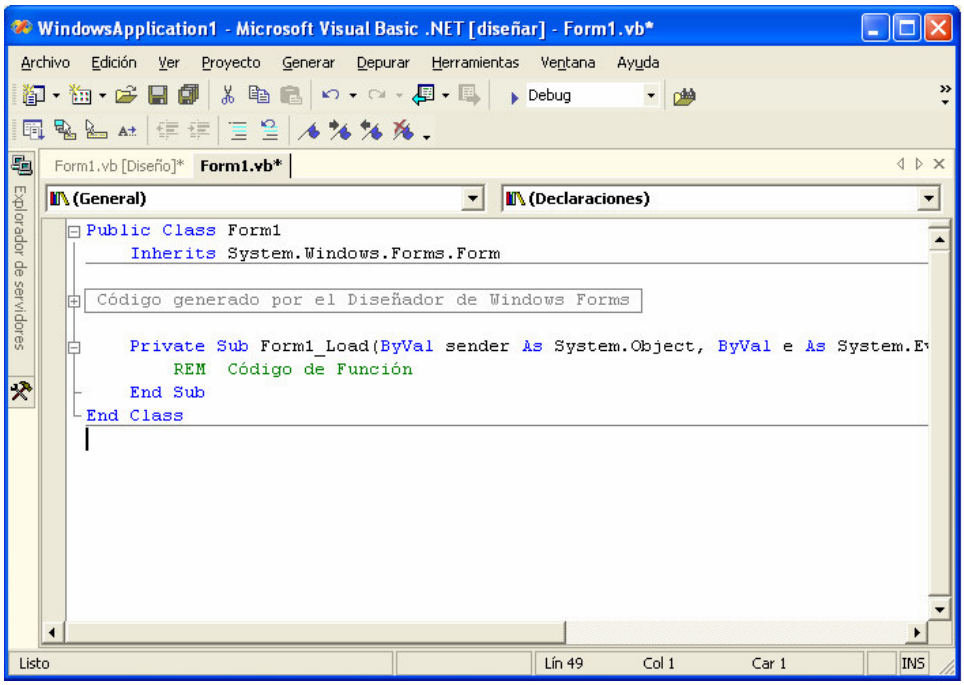


Figura 37. Código de región expandido.

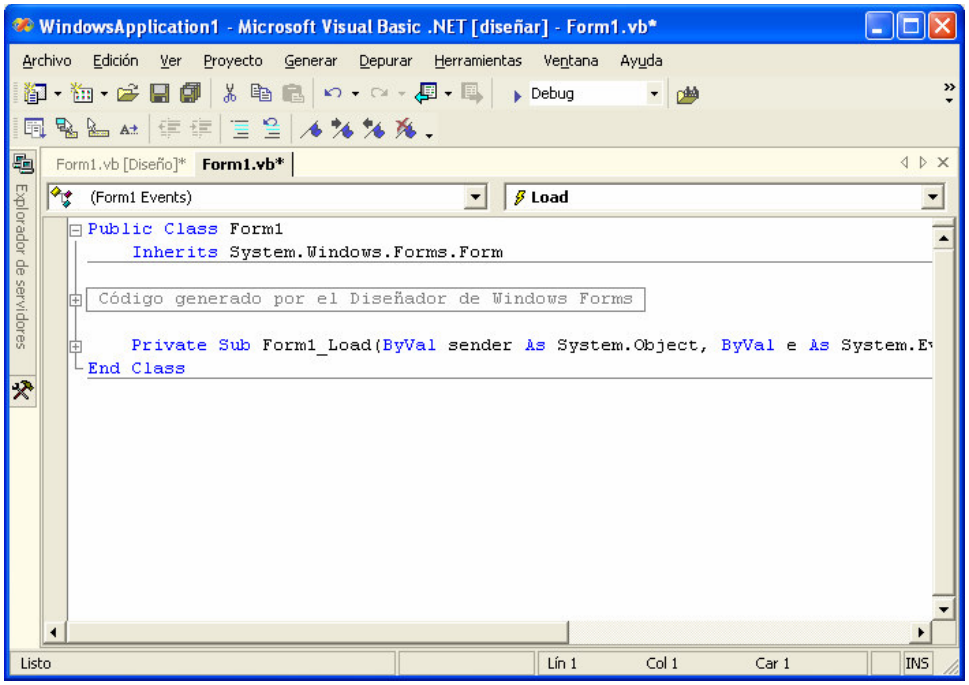


Figura 38. Código de region comprimido

Hay que tener cuidado al anidar regiones con directivas de compilación condicional, ya que todo bloque #if...#endif que comience dentro de una región ha de terminar también dentro de ella. Por tanto, el siguiente uso de la directiva #region no es valido ya que Región Errónea termina estando el bloque #if...#endif abierto:

```
#region RegiónErrónea
        #if A
#endregion
        #endif
```

15.7 El compilador de C# de Microsoft

Hay que explicar pormenorizadamente cómo utilizarlo y qué opciones de compilación admite, pues muchas de ellas se basan en conceptos relacionados con características del lenguaje.

Aunque en un principio lo que se va es a explicar cómo usar el compilador en línea de comandos, dado que Visual Studio.NET también hace uso interno de él para compilar, al final se incluirá un epígrafe dedicado a explicar cómo controlar desde dicha herramienta visual las opciones que se utilizarán al llamarlo.

Sintaxis general de uso del compilador.

El nombre del ejecutable del compilador de C# incluido en el .NET Framework SDK es csc.exe y podrá encontrarlo en la carpeta Microsoft.NET\Framework\v1.0.2914 incluida dentro del directorio de instalación de su versión de Windows[4]. De todas formas, el programa de instalación del SDK lo añade automáticamente al path, por lo que en principio puede llamársele sin problemas desde cualquier directorio.

La forma más básica de llamar al compilador consiste en pasarle como argumentos los nombre de los fuentes a compilar, caso en que intentaría generar en el directorio desde el que se le llame un ejecutable a partir de ellos con el mismo nombre que el primero de los fuentes indicados y extensión .exe Por ejemplo, ante una llamada como:

```
csc FuenteA.cs FuenteB.cs FuenteC.cs
```

[4] El nombre de la carpeta v1.0.2914 variará si utiliza una versión del SDK diferente de la Beta 2.

El compilador intentará generar un fuente FuenteA.exe en el directorio desde el que se lo llamó cuyo código sea el resultante de compilar FuenteA.cs, FuenteB.cs y FuenteC.cs Obviamente, para que ello sea posible el compilador habrá de disponer de permiso de escritura y espacio suficiente en dicho directorio y además alguno de los fuentes indicados tendrá que disponer de un punto de entrada válido.

Este comportamiento por defecto puede variarse especificando en la llamada a csc opciones de compilación adicionales que sigan la sintaxis:

<indicadorOpción><opción>

El <indicadorOpción> puede ser el carácter / o el carácter -, aunque en adelante sólo se hará uso de /. Respecto a <opción>, pueden indicarse dos tipos de opciones:

Flags son opciones cuya aparición o ausencia tienen un determinado significado para el compilador. Se indican de esta manera:

<nombreFlag> <activado?>

<activado> es opcional e indica si se desea activar el significado del flag. Puede ser el carácter + para indicar que sí o el carácter – para indicar que no, aunque en realidad darle el valor + es innecesario porque es lo que se toma por defecto. También hay algunos flags que no admiten ninguno de los dos caracteres, pues se considera que siempre que aparezcan en la llamada al compilador es porque se desea activar su significado y si no apareciesen se consideraría que se desea desactivarlo.

A continuación se muestran algunos ejemplos de uso de un flag llamado /optimize al compilar . No se preocupe por saber ahora para que sirve, sino simplemente fíjese en cómose se usa y note que los dos primeros ejemplos son equivalentes:

csc /optimize Fuente.cs
csc /optimize+ Fuente.cs
csc /optimize- Fuente.cs

Opciones con valores: A diferencia de los flags, son opciones cuya aparición no es válida por sí misma sino que siempre que se usen han de incluir la especificación de uno o varios valores. La forma en que se especifican es:

<nombreFlag>:<valores>

Los <valores> indicados pueden ser cualesquiera, aunque si se desea especificar varios hay que separarlos entre sí con caracteres de coma (,) ó punto y coma (;) Como es lógico, en principio los <valores> indicados no pueden incluir caracteres de espacio ya que éstos se interpretarían como separadores de argumentos en la llamada a csc. Sin embargo, lo que sí se permite es incluirlos si previamente se les encierra entre comillas dobles (“).

Obviamente, como las comillas dobles también tiene un significado especial en los argumentos de csc tampoco será posible incluirlas directamente como carácter en <valores>. En este caso, para solventar esto lo que se hace es interpretarlas como caracteres normales si van precedidas de \ y con su significado especial si no.

De nuevo, esto lleva al problema de que el significado de \ si precede a "también puede ser especial, y para solucionarlo lo ahora que se hace es incluirlo duplicado (\\) si aparece precediendo a un " pero no se desea que tome su significado especial. Ejemplos equivalentes de cómo compilar dando valores a una opción /r son:

csc /r:Lib.dll /r:Lib2.dll Fuente.cs
csc /r:Lib1.dll,Lib2.dll Fuente.cs
csc /r:Lib1.dll;Lib3.dll Fuente.cs

Aunque en los ejemplos mostrados siempre se han incluido las opciones antes que los nombres de los fuentes a compilar, en realidad ello no tiene porqué ser así y se pueden mezclar libremente y en cualquier orden opciones y nombres de fuentes a compilar (salvo excepciones que en su momento se explicarán).

Opciones de compilación.

Antes de empezar es preciso comentar que la mayoría de estas opciones disponen de dos nombres diferentes: un nombre largo que permite deducir con facilidad su utilidad y un nombre corto menos claro pero que permite especificarlas más abreviadamente. Cuando se haga referencia por primera vez a cada opción se utilizará su nombre largo y entre paréntesis se indicará su nombre corto justo a continuación. El resto de referencias a cada opción se harán usando indistintamente uno u otro de sus nombres.

Las opciones básicas son:

- /recurse: Si en vez de indicar el nombre de cada archivo a compilar como se ha dicho se indica como valor de esta opción se consigue que si el compilador no lo encuentra en la ruta indicada lo busque en los subdirectorios de la misma. Por ejemplo, la siguiente llamada indica que se desea compilar el archivo fuente.cs ubicado dentro del directorio c:\Mis Documentos o algún subdirectorio suyo:

  csc /recurse:"Mis Documentos" \fuente.cs

- /target (/t): Por defecto al compilar se genera un ejecutable cuya ejecución provoca la apertura de una ventana de consola si al lanzarlo no hubiese ninguna abierta. Esto puede cambiarse dando uno de los valores indicados en la Tabla (Colocar #) a esta opción:

Valores admitidos por la opción /t de csc.	
Valor	**Tipo de archivo a generar**
exe ó ninguno	Ejecutable con ventana de consola (valor por defecto)
winexe	Ejecutable sin ventana de consola. Útil para escribir aplicaciones de ventanas o sin interfaz
library	Librería
module	Módulo de código no perteneciente a ningún ensamblado

Tanto las librerías como los ejecutables son simples colecciones de tipos de datos compilados. La única diferencia entre ellos es que los segundos disponen de un método especial (Main()) que sirve de punto de entrada a partir del que puede ejecutarse código usando los mecanismo ofrecidos por el sistema operativo (escribiendo su nombre en la línea de comandos, seleccionándolo gráficamente, etc.).

La diferencia de un módulo con los anteriores tipos de archivos es que éste no forma parte de ningún ensamblado mientras que los primeros sí. El CLR no puede trabajar con módulos porque estos carecen de manifiesto, pero crearlos permite disponer de código compilado que pueda añadirse a ensamblados que se generen posteriormente y que podrán acceder a sus miembros internal.

- /main: Si al compilar un ejecutable hubiese más de un punto de entrada válido entre los tipos definidos en los fuentes a compilar se ha de indicar como valor de esta opción cuál es el nombre del tipo que incluye la definición del Main() a utilizar, pues si no el compilador no sabría con cual de todas quedarse. Como es lógico, lo que nunca puede hacerse es definir más de un punto de entrada en un mismo tipo de dato, pues entonces ni siquiera a través de la opción /main podría resolverse la ambigüedad.

- /out (/o): Por defecto el resultado de la compilación de un ejecutable es un archivo .exe con el nombre del fuente compilado que contenga el punto de entrada, y el de la compilación de un módulo o librería es un archivo con el nombre del primero de los fuentes a compilar indicados y extensión dependiente del tipo de archivo generado (.netmodule para módulos y .dll para librerías) Si se desea darle otro nombre basta indicarlo como valor de esta opción. El valor que se le dé ha de incluir la extensión del archivo a generar, lo que permite compilar archivos con extensiones diferentes a las de su tipo. Por ejemplo, para crear un módulo A.exe a partir de un fuente A.cs puede hacerse:

csc /out:A.exe /t:module A.cs

Obviamente, aunque tenga extensión .exe el archivo generado será un módulo y no un ejecutable, por lo que si se intenta ejecutarlo se producirá un error informando de que no es un ejecutable válido. Como puede deducirse, cambiar la extensión de los archivos generados no suele ser útil y sólo podría venir bien para dificultar aposta la comprensión del funcionamiento de una aplicación o para identificar ensamblados con algún significado o contenido especial.

- /reference (/r): Por defecto sólo se buscan definiciones de tipos de datos externas a las fuentes a compilar en la librería mscorlib.dll que forma parte de la BCL. Es importante es señalar que aunque en la plataforma .NET pueden crearse archivos de recursos tanto en formato .txt como .resx, el compilador de C# sólo los admite si están compilados en formato .resources. Para ello, en el SDK se incluye una utilidad llamad resgen.exe que permite compilar en dicho formato archivos de recursos escritos en cualquiera de los formatos anteriores con sólo pasárselos como argumentos. Por ejemplo, si se le llama así:

  resgen misrecursos.resx

  Suponiendo que el contenido de misrecursos.resx sea el de un archivo .resx válido, tras esta llamada se habrá generado en el directorio desde el que se le llamó un archivo misrecursos.resources con el contenido de misrecursos.resx. Para añadir este archivo al ensamblado resultante de una compilación se puede utilizar la opción /linkresource (/linkres) Así por ejemplo, para crear un ensamblado fuente1.dll formado por el código resultante de compilar fuente1.cs y los recursos de misrecursos.resources podría compilarse con:

  csc /t:library fuente1.cs /linkres:misrescursos.resources

  De este modo el archivo de recursos formará parte del ensamblado generado pero permanecerá en un archivo separado de fuente1.dll. Si se desease incrustarlo en él habría que haber compilado con la opción /resource (/res) en vez de /linkres tal y como se muestra a continuación:

  csc /t:library fuente1.cs /res:misrescursos.resources

  Como un tipo especial de recurso que comúnmente suele incrustarse en los ejecutables de los programas es el icono (archivo gráfico en formato .ico) con el que desde las interfaces gráficas de los sistemas operativos se les representará, csc ofrece una opción específica llamada /win32icon en cuyo valor puede indicársele el icono a incrustar:

  csc programa.cs /win32icon:programa.ico

Hay que recordar el uso de archivos de recursos no es un aspecto introducido en la plataforma .NET sino disponible desde hace tiempo en la plataforma Windows en forma de archivos .res. Por compatibilidad con este antiguo formato de recursos, csc incorpora una opción /win32res que permite incrustarlos de igual forma a como /res incrusta los novedosos archivos .resources.

En cualquier caso, hay que señalar que siempre que se añada un archivo de recursos a un ensamblado la visibilidad que se considerará para los recursos que incluya es public.

Acceso al compilador desde Visual Studio.NET

A las opciones de compilación de un proyecto se accede desde VS.NET a través de las páginas de propiedades del mismo, las cuales tiene el aspecto mostrado en la Ilustración 9 y se obtienen seleccionando el proyecto en el Solution Explorer y pulsando sobre View Property Pages en el menú principal de Visual Studio.

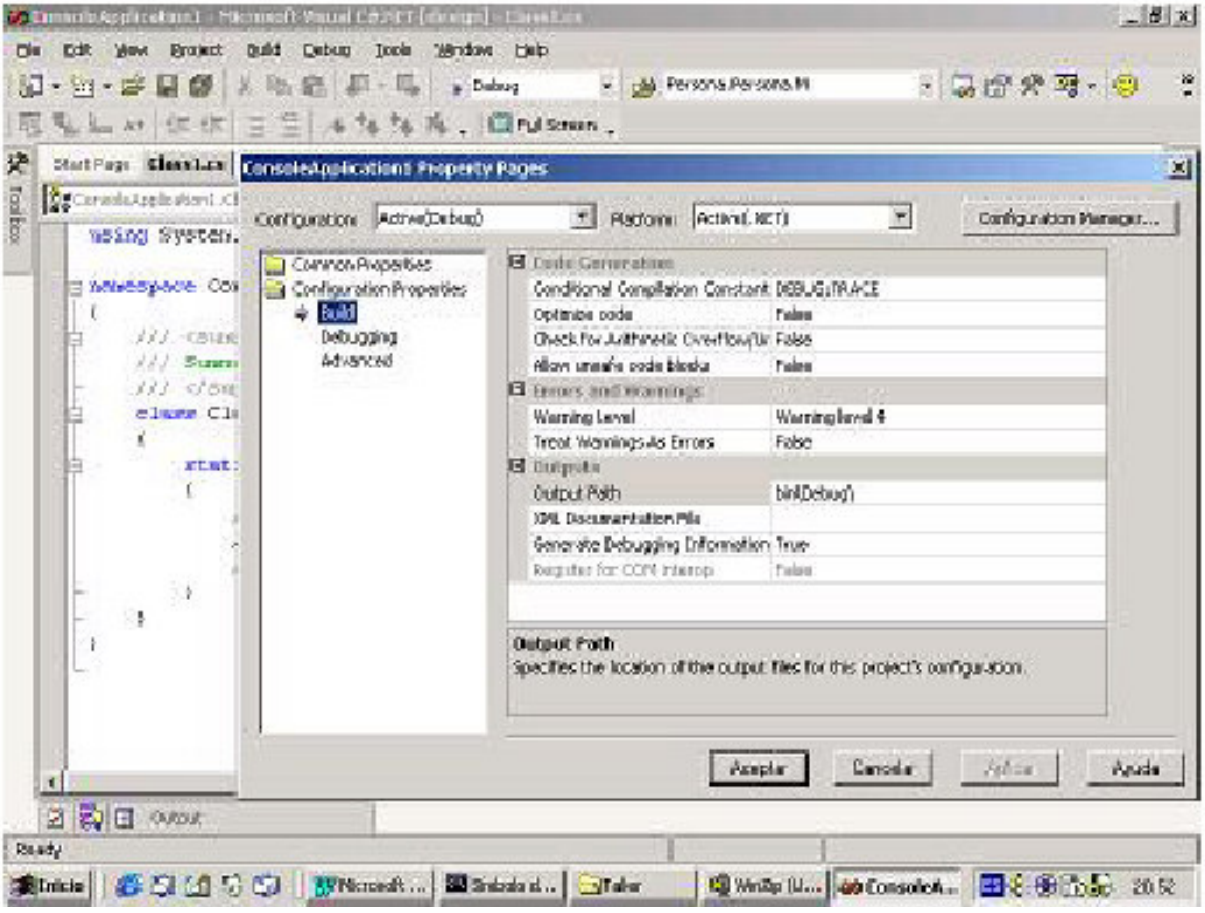


Figura 39. Páginas de propiedades del proyecto en Visual Studio .NET

Para la mayoría de opciones admitidas por csc.exe se incluye en estas páginas controles tales como cajas de texto y listas desplegables que permiten configurarlas de una manera visual, cómoda e intuitiva.

Como puede observar, desde VS.NET no es posible acceder a muchas de las opciones del compilador en línea de comandos. En los casos de /codepage, /fullpaths, /lib, /help, /nologo, /recurse y /utf8output esto es lógico ya que son opciones que pierden su sentido desde dentro en una interfaz gráfica. Hay otros casos en que ello se debe a que se ofrecen desde el menú principal de VS.NET otros mecanismos alternativos para especificarlas, como son los indicados en la Tabla (Colocar #):

Acceso a opciones fuera de las páginas de propiedades	
Opción	***Mecanismo de acceso***
/bugreport	Help - Customer Feedback
/resource	Seleccionar el recurso en Project - Add Existing Item
Opción	Mecanismo de acceso
/reference	Seleccionar la referencia en Project - Add Reference

Finalmente, queda un grupo de opciones que no disponibles simplemente porque la implementación de VS.NET (al menos en la Beta 2) no las contempla, y son @, /linkresource, /nostdlib, /noconfig, /nowarn y /win32res. En este sentido, mención aparte merece el valor module de /t, que tampoco puede usarse en tanto que VS.NET no soporta el trabajo con módulos.

15.8 Documentación de Referencia.

15.9 Bibliografía.

Entre las fuentes de información sobre C# en inglés cabe destacar el documento "C# Language Specification" escrito por Anders Hejlsberg, Scott Wiltamu th y Peter Golde que Microsoft ha remitido al ECMA para la estandarización del lenguaje. Este documento incluye la especificación completa del mismo y Microsoft permite descargarlo gratuitamente desde la dirección http://www.msdn.microsoft.com/net/ecma.

Sin embargo, si lo que busca son libros que expliquen el lenguaje con algo menos de rigurosidad pero de manera mucho más fácil de entender y aplicar, entonces puede consultar la siguiente bibliografía:

- "A programmer's introduction to C#" escrito por Eric Gunnerson y publicado por Apress en 2000.
- C# and the .NET Framework", escrito por Andrew Troelsen y publicado por Apress en 2001
- "C# Essentials", escrito por Beb Albahari, Peter Drayton y Brand Merr il y publicado por O'Reilly en 2000.
- "C# Programming with the Public Beta", escrito por Burton Harvey, Simon Robinson, Julian Templeman y Karli Watson y publicado por Wrox Press en 2000.
- "Inside C#", escrito por Tom Archer y publicado por Microsoft en 20 00
- "Presenting C#", escrito por Christoph Wille y publicado por Sams Publishing en 2000.
- "Professional C#", escrito por Simon Robinson, Burt Harvey, Craig McQueen, Christian.
- Nagel, Morgan Skinner, Jay Glynn, Karli Watson, Ollie Cornes, Jerod Moemeka y publicado por Wrox Press en 2001.
- "Programming C#", escrito por Jesse Liberty y publicado por O'Reilly en 2001

Entre todos estos libros quizás el principalmente recomendable tras leer esta obra pueda ser "Professional C#", pues es el más moderno y abarca numerosos conceptos sobre la aplicación de C# para acceder a la BCL.

Por otra parte, en relación con los libros publicados en 2000 hay que señalar que fueron publicados para el compilador de C# incluido en la Beta 1 del SDK, por lo que no tratan los aspectos nuevos introducidos a partir de la Beta 2 y puede que contengan código de ejemplo que haya quedado obsoleto y actualmente no funcione.

15.10 Información en Internet sobre C#

En esta sección se recogen los principales portales, grupos de noticias y listas de distribución dedicados al lenguaje. Seguramente cuando lea estas líneas habrán surgido muchos más, puede usar la lista ofrecida para encontrar enlaces a los nuevos a partir de los que aquí se recogen.

Portales

Si busca un portal sobre C# escrito en castellano se puedo recomendar "El Rincón en Español de C#" (http://tdg.lsi.us.es/~csharp), que es el primero dedicado a este lenguaje escrito en castellano. Ha sido desarrollado por profesores de la Facultad de Informática y Estadística de Sevilla, y entre los servicios que ofrece cabe destacar sus aplicaciones de ejemplo, FAQ, seminario "on-line" y lista de distribución de correo. Si no le importa que el portal esté en inglés, entonces es de obligada visita el ".NET Developers Center" (http://www.msdn.microsoft.com/net) de Microsoft, ya que al ser los creadores del C# y la plataforma .NET su información sobre los mismos suele ser la más amplia, fiable y actualizada. Entre los servicios que ofrece cabe destacar la posibilidad de descargar gratuitamente el .NET Framework SDK y Visual Studio .NET19, sus numerosos vídeos y artículos técnicos, y sus ejemplos de desarrollo de software profesional de calidad usando estas tecnologías.

Aparte del portal de Microsoft, otros portales dedicados a C# que pueblan la Red son:

- "C# Corner" (http://www.c-sharpcorner.com)
- "C# Help" (http://www.csharphelp.com)
- "C# Station" (http://www.csharp-station.com)
- "Codehound C#" (http://www.codehound.com/csharp)
- "csharpindex.com" (http://www.csharpindex.com)
- "Developersdex" (http ://www.developersdex.com/csharp)
- ".NET Wire" (http://www.dotnetwire.com)

15.11 Grupos de noticias y listas de correo

Microsoft ha puesta a disposición de los desarrolladores numerosos grupos de noticias dedicados a resolver dudas sobre C#, .NET y Visual Studio.NET. Los ofrecidos en castellano son:

microsoft.public.vsnet
microsoft.public.es.csharp

Respecto a los proporcionados en inglés, señalar que aunque algunos de ellos se recogen en la opción Online Community de la página de inicio de VS.NET, la lista completa día a día crece cada vez más:

- microsoft.public.dotnet.academic
- microsoft.public.dotnet.distributed_apps
- microsoft.public.dotnet.faqs
- microsoft.public.dotnet.general
- microsoft.public.dotnet.framework
- microsoft.public.dotnet.framework.adonet
- microsoft.public.dotnet.framework.aspnet

- microsoft.public.dotnet.framework.aspnet.mobile
- microsoft.public.dotnet.framework.aspnet.webservices
- microsoft.public.dotnet.framework.clr
- microsoft.public.dotnet.framework.component_services
- microsoft.public.dotnet.framework.documentation
- microsoft.public.dotnet.framework.interop
- microsoft.public.dotnet.framework.odbcnet
- microsoft.public.dotnet.framework.perfomance
- microsoft.public.dotnet.framework.remoting
- microsoft.public.dotnet. framework.sdk
- microsoft.public.dotnet.framework.setup
- microsoft.public.dotnet.framework.windowsforms
- microsoft.public.dotnet.languages.csharp
- microsoft.public.dotnet.languages.jscript
- microsoft.public.dotnet.languages.vb
- microsoft.public.dotnet .languages.vb.upgrade
- microsoft.public.dotnet.languages.vc
- microsoft.public.dotnet.languages.vc.libraries
- microsoft.public.dotnet.samples
- microsoft.public.dotnet.scripting
- microsoft.public.dotnet.vsa
- microsoft.public.dotnet.xml
- microsoft.publ ic.vsnet.debuggin
- microsoft.public.vsnet.documentation
- microsoft.public.vsnet.enterprise.tools
- microsoft.public.vsnet.faqs
- microsoft.public.vsnet.general
- microsoft.public.vsnet.ide
- microsoft.public.vsnet.samples
- microsoft.public.vsnet.service packs
- microsoft.public.vsnet.setup
- microsoft.public.vsnet.visual_studio_modeler
- microsoft.public.vsnet.vsa
- microsoft.public.vsnet.vsip
- microsoft.public.vsnet.vss

16. MONO PROJECT.

Apartado publicado el
(21/11/2003 10:07) por diarioTi.com

Novell se atrasa con .Net para Windows y Unix
MADRID: El proyecto Mono fue iniciado en 2001 por el desarrollador Miguel de Icaza al alero de Ximian, empresa que posteriormente fue comprada por Novell. Hace aproximadamente un año, de Icaza comentó a los medios que la primera versión de Mono sería presentada antes de fines de 2003.

Sin embargo, Novell escribe en un comunicado que Mono 1.0 probablemente estará listo durante el segundo trimestre de 2004. Mono hará posible que los desarrolladores diseñen software compatible con .Net 1.1 para las plataformas Linux y Unix. La primera versión será distribuida con distintos perfiles y bibliotecas, incluyendo .Net 1.0, .Net 1.1 y los perfiles especificados por ECMA.

Básicamente, Mono 1.0 tendrá soporte para las arquitecturas x86 y PowerPC.

Mono 1.2, que probablemente será lanzado hacia fines de 2004, incluirá bibliotecas que harán posible desarrollar aplicaciones con interfaces gráficos, al igual que soporte para algunas funciones de .Net 1.2.

En un comunicado, Chris Stone, subdirector de Novell, comenta que "Para poder tener éxito, los desarrolladores necesitan un entorno productivo, inferfaces estables de programación y una agenda tecnológica bien definida. Mono ofrece todo lo anterior, además de las ventajas que .Net representa para Linux y Unix".

Según Novell, Mono hará que sea mucho más fácil desarrollar y usar aplicaciones para Linux y Unix.

Mono, un vínculo entre Windows y Linux
La firma Ximian está promoviendo el Proyecto Mono, cuya meta es crear una versión para Unix y Linux de la plataforma de desarrollo .NET de Microsoft.

La calidad de Linux como sistema operativo no se pone en duda. Tampoco su importancia como contrapeso al abrumador dominio de Windows. Gracias a su bajo (o ningún) costo, Linux es una alternativa para usuarios sin muchos recursos. Sin embargo, en Linux hay una evidente carencia de algunos tipos de aplicaciones que sí existen para Windows, sobre todo para uso profesional. Pero quizá esta situación cambie gracias a una tecnología que, paradójicamente, proviene de Microsoft: .NET.

La empresa Ximian (http://www.ximian.com/), que fue comprada en el 2003 por Novell, está promoviendo el Proyecto Mono, cuyo objetivo es permitir que los programadores de .NET creen aplicaciones que funcionen por igual en los sistemas operativos Windows, Linux y Unix.

Desde el lanzamiento de .NET han surgido proyectos que buscan crear versiones de .NET Framework para sistemas operativos diferentes a Windows, ya que si bien Microsoft promueve .NET como una plataforma cruzada y permite su libre uso, no ha mostrado interés en desarrollar versiones para sistemas operativos diferentes a Windows (el .NET Framework es la infraestructura de software que permite desarrollar y ejecutar aplicaciones de tecnología .NET en Windows).

De esos proyectos el más exitoso es Mono Project (http://www.go-mono.com/). Es una iniciativa de software libre, pero cuenta con la colaboración técnica de Microsoft, lo que le asegura un buen futuro. Aunque Mono se considera como una versión de .Net Framework, tiene ciertas diferencias, y su desarrollo actual dista del que ha alcanzado este último.

Como ventaja, las aplicaciones desarrolladas para Mono funcionarán en cualquier sistema operativo, incluido Windows, lo que no sucede con .NET Framework, limitado a los sistemas Microsoft.

16.1 Disponible gratis

Este proyecto es de muy amplio alcance, pues se están desarrollando versiones de Mono para Linux (hay ediciones para Red Hat, Debian y Suse), Mac, Unix y Windows.

También se trabaja en herramientas de programación GNU, que estarán disponibles pronto. El desarrollo de Mono marcha bastante bien. Hay varias 'secciones' terminadas, como el soporte para el lenguaje de programación C#, y se está completando para el lenguaje Visual Basic.

Quien desee utilizar Mono puede bajar gratis el paquete de instalación desde la dirección www.go-mono.com/download.html (debe verificar que sea la versión para su sistema operativo). La instalación es rápida y, en el caso de Windows, no genera ningún conflicto con .NET Framework.

Las aplicaciones para Mono se desarrollan sobre todo utilizando C#, pero es posible que en un futuro cercano se puedan emplear también herramientas comerciales tan potentes como Visual Basic.NET y Visual C++.NET, lo que resultaría ideal para los programadores profesionales.

16.2 Wine: solución parcial

La idea de que las aplicaciones de Windows se ejecuten en Linux no es nueva. De por sí, Linux ha venido abriéndose a Windows; sus versiones recientes pueden leer discos con formato MS-DOS y muchas de sus aplicaciones son compatibles con Microsoft Office.

Pero la alternativa más conocida es Wine (http://www.winehq.com/), un proyecto que tiene más de una década de vida. Wine emula a Windows dentro de Linux, pero no es perfecto. No funciona en todas las distribuciones (sí en las más famosas, como Red Hat y Suse) y el rendimiento de las aplicaciones es pobre, algo típico en las emulaciones. Además, no es fácil de configurar e instalar, lo que es una barrera para usuarios principiantes.

Finalmente, no todos los programas para Windows se ejecutan en Wine.
De hecho, ese funcionamiento se limita a software muy bien diseñado, como Word

16.3 Microsoft se acerca a Ximian

Miguel de Icaza es uno de los personajes más conocidos dentro del mundo Linux. Este mexicano fundó en 1999 la empresa Ximian, responsable de Gnome y de otros programas muy interesantes.

Tras la compra de Suse por parte de Novell, la segunda empresa decidió adquirir también Ximian, cuya sede está en Boston, y mantuvo a De Icaza en la dirección de Gnome y del proyecto Mono.

La operación le permite a Ximian tener el soporte financiero de Novell, que aspira a convertirse en el principal proveedor de Linux. De Icaza ha recibido críticas de muchos programadores y usuarios linux, pues gracias a Mono ha entablado una relación cordial con Microsoft, que patrocinó su presencia en varios certámenes académicos.

Luego de mostrar durante mucho tiempo cierta hostilidad hacia la comunidad del software libre, en julio del 2003 Microsoft anunció que proveería asistencia técnica al proyecto Mono. David Stutz, gerente del programa .NET para plataformas abiertas, declaró que Microsoft apoyaría el trabajo de Ximian “como una prueba de la apertura y viabilidad de .NET”.

Stutz también afirmó que el trabajo de Ximian les da a los desarrolladores la posibilidad de elegir entre Microsoft.NET y Mono.

Actualizado al 23/07/2004

¿Que es Mono?

Mono es una implementación de varias tecnologías:

- Un compilador para el lenguaje C#
- Un entorno de ejecución virtual: Un compilador JIT (justo-a-tiempo), gestión de memoria, interprete (mint), motor multiproceso.
- Una máquina virtual para los bytecodes del Lenguaje Intermedio Común (CLI)
- Una implementación de la librería de clases de .NET: manipulación XML, Remoting, Reflection.Emit, Xslt, etc.
- Librería de clases multiplataforma para el acceso a bases de datos: Postgress, MySQL, DB2, TDS, Sybase, Oracle, ODBC y Gnome-GDA.
- Liberia de clases UNIX: Mono.Posix
- Librería de clases GNOME: la familia Gtk#
- Código para compilar a código nativo antes de ser ejecutado (Compilación AOT, AOT=ahead-of-time, antes-de-tiempo)

En el mundo Microsoft, a este conjunto se le suele llamar la plataforma .NET en contraposición a .NET, que un término comercial no muy concreto. Cuando se refiere a la plataforma .NET en este documento, se hace referencia a estas tecnologías.

Hay gente a la que le puede parecer que todo esto es muy parecido a Java y la máquina virtual de Java. Tienen razón, esto es muy parecido a Java.

Pero el CLI (Lenguaje Intermedio Común, el equivalente de los bytecodes de Java) tiene una característica que no se encuentra en Java: la representación de éste es lo suficientemente potente como para servir para varios lenguajes: puedes mezclar lenguajes como C++, C, Fortran, Eiffel, Lisp, Java, C# y Visual Basic, por ejemplo, en el mismo programa.

El CLI y la promesa de una independencia de lenguaje Bertrand Meyer (el creador del lenguaje Eiffel) escribió un artículo interesante. En este artículo explicaba que esta tecnología permite considerar a los lenguajes de programación la base para solucionar un determinado problema, y no depender de las librerías de las que se disponga.

Viene a decir que gracias al CLI, cualquier lenguaje podrá acceder a unas librerías comunes sin importar en qué lenguaje estuvieron escritas. Por ejemplo, los ingenieros de software ya no tendrían que escoger Fortran únicamente porque es el lenguaje en el que la librería matemática está disponible: ahora puede escoger el lenguaje más apropiado según el problema con el que se enfrenten.

16.4 Mono y GNOME

GNOME siempre ha intentado tener un buen soporte para múltiples lenguajes de programación, porque se han dado cuenta de que no importa lo mucho que le guste C como lenguaje, siempre hay personas que desean utilizar las librerías GNOME desde su lenguaje favorito, que a menudo no es C.

La estrategia hasta ahora ha sido crear enlaces por separado para cada uno de los lenguajes que se utiliza. Y ha funcionado bastante bien. Hay comunidades enteras que usan Python, Perl, Guile o Ada con los recubrientos para construir aplicaciones gtk+ y GNOME que se utilizan tanto para el desarrollo rápido de aplicaciones como para aplicaciones robustas.

Sin embargo, mantener los enlaces (bidings) actualizados para los diferentes lenguajes ha resultado ser un proceso que consume grandes cantidades de tiempo y recursos. No importa lo automatizado que esté el proceso: siempre hay una cantidad ingente de trabajo que necesita ser realizado manualmente.

16.5 Evolución de la plataforma de desarrollo

Microsoft tiene unas API s terribles a las que debe hacer frente. Cualquier persona que haya usado Win32 sabe que las varias capas que se superponen hacen que sea la API más horrible jamás construida.

Además, como evolución de las API s, componentes, gestión de memoria a versiones arqueadas de COM han hecho que la plataforma sea todavía más horrible para programar.

Microsoft ha solucionado todas estas desventajas con un nuevo lenguaje de programación. Ha incorporado ideas de Java, y las han extendido para satisfacer las necesidades de los programadores. Se pude decir que lo tomaron donde Java lo dejó.

Ahora, la plataforma UNIX, GNOME incluido, tienen algunos de estos problemas: Las API s han evolucionado. Las librerías han sido creadas por grupos dispersos (PNG, JPEG, Gtk+, Xml, Bonobo, etc) y el resultado es que el programador debe aprender más de lo que le gustaría en el transcurso del desarrollo de una aplicación de tamaño medio.

Ximian fundó hace tiempo el recubrimiento de Perl, y se estuvo haciendo mucho trabajo en Bonobo (más del que se hace hoy en día) porque se creía que esto ayudaría a conseguir independencia de lenguaje y que ayudaría a promocionar el desarrollo de aplicaciones con lenguajes de script.

Cuando salió C#, el CLR y la librería de clases, se dieron cuenta de que estaban resolviendo el problema de una forma muy interesante. Al menos es interesante desde un punto de vista técnico. La nueva plataforma prometía mucho.

Desde entonces, un par de programadores de Ximian se dedicaron a implementar la especificación de la plataforma .NET. Estas personas vinieron precisamente de las áreas de interoperatibilidad entre lenguajes: Dick Porter había estado trabajando antes en la implementación de ORBit y SOAP; dietmar Mauer venía del desarrollo de Bonobo y Paolo Molaro estuvo trabajando en el recubrimiento de Gtk+/Gnome/Bonobo para Perl.

16.6 Evolution, Gnumeric y GNOME

Los proyectos grandes de software deben de hacer frente a problemas que no existen en proyectos menores. Los programas que tienen un ciclo de vida largo deben hacer frente de distinta forma a la gestión de memoria que los programas pequeños.

Llega un momento en el que se da cuenta que ha perdido demasiado tiempo escribiendo destructores, solucionando un problema de memoria, usando funciones no seguras de bajo nivel, y que ha implementado demasiadas listas enlazadas

La plataforma .NET habla sobre todo de productividad: aunque Microsoft haya desarrollado estas tecnologías teniendo en mente web services, el mayor beneficio de ésta es aumentar la productividad del programador.

Evolution tardó dos años en desarrollarse y en algunos momentos llegó a tener 17 ingenieros trabajando en el proyecto.

Porqué está Mono relacionado con GNOME?

Mono hará uso de Gtk+, Gnome-Db, Libart, Gnome-Print y otras tecnologías de GNOME como parte de su implementación de la librería de clases. Porque eso es con lo que el equipo está familiarizado.

Así pues, puede escribir aplicaciones en Windows con la librería Gtk# y cuando ejecute en su plataforma UNIX, simplemente se integrará en el escritorio GNOME. Lo mismo es cierto en el caso de MacOS X.

Igualmente puede escribir aplicaciones en Windows usando Windows.Forms, y ejecutarlas el UNIX, sólo que a través de Wine, con lo que se consigue una integración parcial. De cualquier forma, Windows.Forms no va a ser continuado, pues Microsoft va a remodelar todo esto para la siguiente versión, y entonces se discutirá si merece la pena escribir un librería equivalente basada en Gtk#. De cualquier forma, Mono patrocina el uso de Gtk# y en estos momentos es una elección muy estable para escribir GUI s.

16.7 GNU está basada en una tecnología propietaria

GNU es una implementación de Unix. Linux es una implementación del núcleo de Unix. Antes de que existiera Linux y Berkeley Unix, Unix era una tecnología propietaria, construida por ATT (el cual en aquellos días era un monopolio).

Aún así, varios programadores cogieron lo que era bueno de Unix, e implementaron su propia versión. El lenguaje escogido fue C (el cual fue desarrollado en ATT). Incluso C++ fue desarrollado en ATT.

Piensa en Mono siguiendo el mismo proceso: se esta trayendo la mejor tecnología que existe a la plataforma de software libre. Y al mismo tiempo resulta ser una evolución magnífica de la plataforma de desarrollo.

16.8 Otros usos de Mono

Los programadores de Windows ya saber programar en ella, por lo tanto es fácil la migración de programadores de Windows a esta plataforma.

C# es un lenguaje Orientado a Objetos y de carácter general. Sirve para prácticamente cualquier tarea. Es un gran lenguaje, pero si no le gusta, simplemente puede escoge otro soportado por la plataforma .NET/Mono (prácticamente cualquier lenguaje es susceptible de ser compilado para la plataforma .NET).

16.9 Características de Mono

Independencia de lenguaje: puede usar clases escritas en cualquier lenguaje soportado por Mono (por ahora, C#, Mono Basic, Java, Nemerle, MonoLOGO). Independencia de plataforma: las aplicaciones son muy portables, y compatibles en binario entre plataformas Gran soporte para bases de datos: MS SQL, MySQL, Postgres, OLE DB, ..., en total hasta 27 bases de datos.

Velocidad: el lenguaje intermedio se compila en cada plataforma con unos compiladores muy rápidos (JIT), y por lo tanto es mucho más rápido que lenguajes interpretados con PHP o Python, y más rápido en la compilación (JIT) que Java. Únicamente un poco más lento que C.

Gestión automática de memoria: es una fuente inagotable de errores y se pierde una gran cantidad de tiempo programando esto. Si se automatiza, se obtiene más tiempo que se puede dedicar a resolver el verdadero problema.

Seguridad.

Aplicaciones web: Cualquier lenguaje soportado por Mono se puede usar para aplicaciones web. No hay necesidad de lenguajes especiales como PHP.

Web services: soporte para SOAP

Soporte para XML: Mono tiene muchas clases para trabajar con xml

Extensa librería de clases: Criptografía, HTTP, Bases de datos, GUI, etc.

Aplicaciones GUI multiplataformas: se pueden escribir aplicaciones con interfaz gráfica que se ejecutan invariantemente en multitud de plataformas. Por ejemplo, Gtk# es muy potente y disponible prácticamente en cualquier plataforma.

Aplicaciones GNOME, KDE y Windows: Mono tiene recubrimientos para las librerías gnome y qt. System.Windows.Forms también están implementadas, aunque por ahora en estado muy inicial y hacen uso del emulador de windows Wine

16.20 Primeros pasos con la plataforma Mono

La plataforma Mono, la versión libre de .NET, ya funciona totalmente bajo GNU/Linux, sin necesidad de apoyarse en Windows para sus aspectos de desarrollo y funcionamiento básico.

Esto hito marca el momento en el que los que trabajan en entornos GNU/Linux puedan comenzar a disfrutar totalmente de esta nueva plataforma de desarrollo y, quizá aún más importante, marca el momento a partir del cual pueden ayudar en la construcción de esta nueva plataforma de desarrollo.

16.21 Instalación de Mono

La creación de Mono ha sido un arduo proceso en el que se ha dependido mucho de las herramientas de Microsoft para ir creando la plataforma. Hasta el 16 de Julio de 2002 no ha sido posible desarrollar totalmente todas las labores de Mono en GNU/Linux. El desarrollo del compilador y todas las clases de la plataforma Mono han necesitado ser desarrolladas inicialmente con el compilador y el entorno .NET de Microsoft. Poco a poco se ha ido logrando independencia en GNU/Linux, primero logrando que el compilador de Mono, desarrollado en C#, pudiera ser ejecutado dentro del entorno de ejecución Mono en GNU/Linux.

Una vez que se tuvo un compilador de C# en GNU/Linux, se aceleró el desarrollo y poco a poco, la mayoría de las clases de la plataforma .NET se fueron implementando y comprobando su funcionamiento en GNU/Linux, hasta lograr el hito actual de que todas las clases fundamentales cd Mono, corlib, fueran compiladas y ejecutadas con el compilador C# (mcs) en GNU/Linux y ejecutadas en el entorno Mono de GNU/Linux (mono).

Hoy en día Mono funciona de manera bastante estable, y la librería de clases se ha extendido y ya incluye varios componentes, como Gtk# (recubrimiento a las librerías de GNOME, para poder hacer aplicaciones gráficas con Mono), Mono.Data (acceso a bases de datos), etc.

2.1. Instalación de Mono en Debian

Los paquetes de Debian Sid están ya en el repositorio oficial de Debian. También se puede disponer de paquetes experimentales siguiendo las instrucciones mostradas en http://pkg-mono.alioth.debian.org/

Para instalar los paquetes de Mono en Debian Sid (unstable) basta con ejecutar apt-get update y tras ello apt-get install mono. Si además quiere probar la biblioteca Gtk# tendrá que ejecutar apt-get install libgtk-cil. Es necesario tener instaladas las bibliotecas gtk2 y glib2 para poder utilizar Mono.

2.2. Instalación en Fedora Core (usando Yum)

Añadir estas líneas a /etc/yum.conf:

```
[mono]name=Mono 1.0 Release: Fedora Core Linux 2, x86
baseurl=http://mono2.ximian.com/archive/1.0/fedora-2-i386/
```

si usa Fedora Core 2. Si usas la versión 1 de Fedora, entonces las líneas deberían ser

```
[mono]name=Mono 1.0 Release: Fedora Core Linux 1, x86
baseurl=http://mono2.ximian.com/archive/1.0/fedora-1-i386/
```

y entonces ejecuta
yum update

para actualizar la lista de paquetes y
yum mono-complete

para instalar toda la plataforma Mono.

2.3. Instalación en Gentoo

En el caso de gentoo algunos paquetes están en portage. Para instalar mono, mcs y el resto de componentes básicos tiene que ejecutar emerge mono. Si además quiere instalar Gtk# deberá ejecutar emerge gtk-sharp. Para instalar algunos otros paquetes relacionados con mono: emerge gecko-sharp, emerge gtk-sourceview, emerge ikvm, emerge monodevelop, emerge xsp...

Sin embargo, si quiere tener instalada la ultimísima versión, debemos introducir en /etc/portage/package.keywords (si no existe se crea) lo siguiente:

```
dev-util/monodevelop ~x86
dev-util/monodoc ~x86
dev-dotnet/gtksourceview-sharp ~x86
dev-dotnet/gecko-sharp ~x86
x11-libs/gtk-sharp ~x86
dev-dotnet/mono ~x86
dev-libs/icu ~x86
dev-dotnet/xsp ~x86
dev-dotnet/ikvm ~x86
```

Entonces ahora sí se puede hacer

```
emerge mono
emerge gtk-sharp
emerge monodevelop
...
```

2.4. Instalación de Mono desde código fuente

La instalación desde el código fuente puede interesarle si le gusta compilar los programas que usa o si no existen paquetes para la distribución, por ejemplo. Para una instalación básica (mono y el compilador de C#, mcs), solamente necesita el paquete "Mono Runtime" disponible en http://mono-project.com/downloads/index.html. No necesitará el paquete "Mono Class Libraries and C# Compiler 1.0", pues se distribuye de forma precompilada dentro del paquete "Mono Runtime". El paquete "Mono Class Libraries and C# Compiler" solamente es útil para programadores que quieran conocer todo el código fuente o si quiere hacer una compilación de mono "desde cero".

Para llevar a cabo la compilación necesitará pkg-config, disponible en http://www.freedesktop.org/Software/pkgconfig, y glib-2.0, disponible en http://gtk.org
Una vez situado en el directorio en el que haya descomprimido el paquete, debe ejecutar:

```
./configure --prefix=/usr/local
```

que, si todo ha ido bien, finalizará con un mensaje parecido a este:

```
GC:         included
ICU:        no, if you want full i18n support download it from:

http://oss.software.ibm.com/icu/index.html
__thread:    yes
SIGALTSTACK: no
Engine:      Building and using the JIT
2.0 Alpha:   no
JNI support: yes
```

Ahora llega el proceso de compilación en sí, que se consigue con el comando make. Para poder instalar el sistema tal vez tenga que ingresar en la cuenta del súper usuario (comando su) y ejecutar make install

Nota: debido a varias experiencia personales, han habido problemas en la compilación, más concretamente en cuanto al soporte para JNI, que se solucionó repitiendo todo el proceso pero sustituyendo el primer comando por ./configure --prefix=/usr/local --with-ikvm-jni=no. También es posible que tenga que exportar la variable PKG_CONFIG_PATH, lo que se hace ejecutando export PKG_CONFIG_PATH /usr/lib/pkgconfig:/usr/local/lib/pkgconfig, aunque los directorios pueden cambiar dependiendo de la distribución.

Si además quiere compilar las librerías gtk#, necesitará el paquete Gtk# disponible http://mono-project.com/downloads/index.html. Los pasos a seguir son idénticos a los de mono, ./configure --prefix=/usr/local, make y make install, aunque esta vez tal vez necesites tener algunos paquetes adicionales, como gtk, libxml, libxml-dev, etc.

16.22 El lenguaje C#

Ahora que ya tiene instalado Mono pero, ¿qué puede hacer con él? Pues sin duda lo primero es hacer un sencillo programa en C# y probar a compilarlo. Ejemplo:

```
using System;

public class Hola {

	public static void Main () {
		Console.WriteLine ("¡Mono ya vive en GNU/Linux!");

	}
}
```

Como es habitual, este ejemplo no es muy original, que muestra una cadena por la consola. Pero el ejemplo minimalista es el que mejor le permite centrarse en los detalles iniciales del programa y del lenguaje.

Puede crear el código con su editor de textos preferido y guardarlo en un archivo ejemplo.cs. Poco a poco se ira familiarizando con la extensión "cs", csharp, que acompañará a todos los archivos fuentes en C#. Ha llegado el momento de compilar su primer programa en C#. Que se haga el silencio.

```
acs@localhost:~/$ mcs ejemplo.cs
Compilation succeeded
```

mcs es el compilador de C# de Mono, desarrollado íntegramente en C#. Es capaz ya de compilarse a sí mismo, y a todas las clases del núcleo de Mono por lo que es ya muy completo y un ejemplo tan sencillo como este, no le va a poner en ningún aprieto.

La compilación produce como salida el ejecutable ejemplo.exe. Esta terminación a más de uno no le traerá buenos recuerdos. Sí, es la terminación de los ejecutables de MS-DOS y Windows. Pero en este caso en realidad, hace referencia a código IL (Lenguaje Intermedio), que viene a ser como el bytecode de Java. Entonces si quiere ejecutarlos

acs@localhost:~/$ mono ejemplo.exe
Mono ya vive en GNU/Linux!

Explorando la librería de clases (con monodoc)

Monodoc es una herramienta para mono que incluye un navegador con documentación sobre la librería de clases y sobre otras librerías. Para ello tendrá que tener instalado el paquete monodoc. Si utiliza debian Sid, esto es muy fácil, simplemente ejectura apt-get install monodoc.

Si por ejemplo lo que le interesa es conocer la clase Console que se ha utilizado anteriormente para imprimir información por pantalla entonces abra en monodoc las pestañas referentes a Class Library -> System y hacemos click sobre Console Class. Tiene un texto con enlaces a una lista con los miembros de la clase, una descripción de la clase (por ahora en inglés) y un pequeño ejemplo (aquí ligeramente modificado):

```
using System;

public class ConsoleTest {
	public static void Main() {
		Console.Write("Hola ");
		Console.WriteLine("Mundo!");
		Console.Write("Cuál es tu nombre?: ");
		string name = Console.ReadLine();
		Console.Write("Hola, ");
		Console.Write(name);
		Console.WriteLine("!");
	}
}
```

, que nos muestra los pasos que se siguen para imprimir y para tomar datos por la consola.

Si lo compila con mcs prueba.cs y lo ejecuta con mono prueba.exe obtendrá algo así:
Hola Mundo!
	Cuál es tu nombre?: Edgardo
	Hola, Edgardo.
Gtk# para las ventanas

Sin lugar a dudas una de las grandes atracciones actuales de Mono es Gtk#, en recubrimiento de las librerías de GNOME para Mono, realizado en C#. A pesar de lo desafortunado del nombre, Gtk# no es solamente un recubrimiento a la librería gtk, sino que incluye también otras librerías relacionadas con GNOME. Gtk# ha demostrado estar preparada para poder realizar aplicaciones complejas, por lo que se convierte a la herramienta gráfica ideal para desarrollar aplicaciones en Mono. Ejemplos de aplicaciones realizadas en Gtk# incluyen:

Se mostrara con que facilidad se programan aplicaciones utilizando Gtk#, utilizando una sintaxis mucho menos pesada que en la programación de Gtk en C. Y nada mejor que un sencillo ejemplo para verlo.

```
using Gtk;
using System;

class Hello {

	static void Main()
	{
		Application.Init ();

		Button btn = new Button ("Que mona Gtk#!");
		btn.Clicked += new EventHandler (EscribeHola);

		Window window = new Window ("Mono Gtk#");
		window.DeleteEvent += new DeleteEventHandler (Salir);
		window.DefaultHeight = 200;
		window.DefaultWidth = 300;
		window.Add (btn);
		window.ShowAll ();

		Application.Run ();
	}


	static void Salir (object obj, DeleteEventArgs args)
	{
		Application.Quit ();
	}

	static void EscribeHola (object obj, EventArgs args)
	{
		Console.WriteLine("Hola Mundo");
		Application.Quit ();
	}
}
```

Para compilar el ejemplo es necesario tener instalado "gtk-sharp" mencionado anteriorment. Para compilar necesitamos especificar las bibliotecas a utilizar:

```
mcs -pkg:gtk-sharp prueba.cs
```

A diferencia de Java, se usa el estilo de C y C++ donde en vez de utilizar una variable con todas las bibliotecas disponibles (CLASSPATH), esta especifica al momento de compilar. El resultado de la ejecución de mono prueba.exe es el siguiente:

Figura 1. Ventana de Gtk# (Me falta conseguirla porque no estaba)

16.23 Cómo colaborar en Mono

Mono no ha sido todavía completado del todo. Es un proyecto bajo intenso desarrollo y con una comunidad entusiasta. Existen diversas formas de colaborar, desde escribiendo documentación hasta resolviendo errores presentes en mono. Para colaborar en mono lee la página sobre cómo colaborar en http://mono-project.com/contributing/index.html

16.24 Previsiones de futuro en Mono

Los campos a los que quiere llegar son muy amplios, tan ambiciosos como los de la plataforma .NET. Desde web services hasta aplicaciones de escritorio, todo tratado de una forma homogénea con reutilización de código entre distintos lenguajes de programación. Tras la versión 1.0 se trabajarán especialmente los siguientes aspectos:

- Implementación de C# 2.1
- Portar el compilador JIT (just-in-time) a máquinas con procesadores AMD64
- Mejorar el compilador para Visual Basic
- Mejora de Gtk# y el diseño de interfaces gráficas
- Soporte para Windows.Forms
- Optimizaciones del compilador JIT

16.25 Programas Mono en entornos Windows

Tanto el entorno de ejecución Mono como el compilador funcionan perfectamente en Windows, por lo que todos los desarrollos que haga en GNU/Linux, se podrán ejecutar también en Windows. ¿Qué posibilidades se abren en este nuevo entorno?
Conclusiones del Proyecto Mono

La arquitectura Java basada en la idea de tener aplicaciones multiplataforma revolucionó la industria informática a finales del siglo XX. Sin embargo, las limitaciones que puso SUN al uso de la tecnología y el no terminar de estandarizar la tecnología, abrió la puerta a nuevas plataformas.

Microsoft, basándose en gran parte en las ideas de la plataforma Java, mejoró la plataforma permitiendo romper con la necesidad de utilizar un único lenguaje de programación y creando un entorno que, además de ser multiplataforma, es multilenguaje, algo similar a CORBA pero ya en el lado de la implementación y con muchas más funcionalidades incorporadas.

Miguel de Icaza, desde Ximian, supo ver el gran atractivo de esta nueva plataforma .NET, y en cuanto Microsoft se decidió a estandarizarla, lanzó el proyecto Mono, con el objetivo de implementar la plataforma .NET en diversas plataformas, entre ellas GNU/Linux y con licencia libre.

Hoy esa plataforma es ya una realidad, y se pueden llevar a cabo aplicaciones en Mono complejas. Aún queda mucho trabajo por hacer, pero la parte más dura, la de la creación del entorno de ejecución, el compilador y las clases ya está casi superada.

16.26 Referencias

Mono: http://www.mono-project.com/about/index.html
Ximian http://www.novell.com/linux/ximian.html

17. PHP.

El lenguaje PHP es un lenguaje de programación de estilo clásico, (es un lenguaje de programación con variables, sentencias condicionales, bucles, funciones, etc.) No es un lenguaje de marcas como podría ser HTML, XML o WML. Está más cercano a JavaScript o a C, para aquellos que conocen estos lenguajes.

Pero a diferencia de Java o JavaScript que se ejecutan en el navegador, PHP se ejecuta en el servidor, por eso permite acceder a los recursos que tenga el servidor, un ejemplo podría ser una base de datos. El programa PHP es ejecutado en el servidor y el resultado enviado al navegador. El resultado es normalmente una página HTML pero igualmente podría ser una pagina WML.

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que su navegador lo soporte, es independiente del navegador, pero sin embargo para que sus páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP.

La ventaja que tiene PHP sobre otros lenguajes de programación que se ejecutan en el servidor (como podrían ser los script CGI Perl), es que permite intercalar las sentencias PHP en las paginas HTML.

PHP ("Hypertext Preprocessor") es un lenguaje ampliamente utilizado en aplicaciones de servidor en conjunción de Apache, sin embargo, para ser capaz de desarrollar dichas aplicaciones es necesario instalar su correspondiente módulo.

A continuación se describe la instalación del módulo PHP dinámico, como alternativa puede optar por instalar el módulo PHP estático, sin embargo, este último requiere que Apache sea re-compilado nuevamente, mientras el primero, como su nombre lo implica, es cargado automáticamente sin la necesidad de re-compilar Apache.

17.1 Proceso de Instalación

Descargar el módulo más reciente de PHP en http://www.php.net/downloads.php. Una vez obtenido el archivo, éste debe ser descomprimido en un directorio temporal (/tmp por lo general) para poder iniciar la instalación.

Verifique la presencia de los módulos http_code.c y mod_so.c ejecutando el comando httpd -l; por "Default" estos módulos deben encontrarse en su instalación de Apache 2, sin embargo, en caso de no estar presentes debe configurarlos e instalarlos acordemente. Descienda al directorio raíz de PHP en el directorio temporal y ejecute el siguiente comando:

./configure --with-mysql --with-apxs2=/usr/local/apache2/bin/apxs

El parámetro --with-mysql indica que el módulo PHP debe ser compilado con soporte para la Base de Datos MySQL, lo anterior permite que las aplicaciones puedan interactuar con dicho depósito de información, un caso muy típico al utilizar PHP y Software Open-Source como Apache.

--with-apxs2=/usr/local/apache2/bin/apxs indica el directorio del ejecutable apxs utilizado para compilar módulos, dicho ejecutable se encuentra en la instalación de Apache 2. NOTA: En caso de estar realizado esta instalación para la versión 1.x de Apache, el parámetro a utilizar sería simplemente --with-apx (sin el dos).

Posteriormente ejecute los siguientes comandos para compilar e instalar respectivamente el módulo PHP

make; make install;

Al realizar este último paso, el módulo PHP será copiado automáticamente al directorio /usr/local/apache2/modules.

Finalmente, debe generar el archivo principal de configuración para PHP llamado php.ini, en un sistema como Linux esta ubicación sería el directorio /usr/local/lib/.

La misma distribución de PHP incluye dos archivos muestra de este tipo, uno de ellos llamado php.ini-dist que contiene parámetros básicos y php.ini-recommended que incluye parámetros recomendados para ambientes de producción, según sean sus requerimientos, puede copiar cualquiera de estos archivos a la ubicación antes mencionada.

17.2 Configuración httpd.conf

También debe realizar algunas modificaciones al archivo principal httpd.conf de Apache:
Verifique que exista el siguiente renglón en su estructura:

LoadModule php#_module modules/libphp#.so
Donde # el es número de la versión de PHP

Esta configuración le indica al Servidor que cargue el módulo PHP.
También debe agregar la siguiente configuración para indicarle al Servidor de Páginas que todo documento con extensión .php sea procesado por PHP.

AddType application/x-httpd-php .php

Pruebas
Para verificar que ha instalado el módulo PHP correctamente, realice la siguiente prueba:
Reinice el Servidor Apache a través del comando apachectl restart, según descrito en la sección Configuración y Ejecución para que sea cargado el módulo PHP.
Coloque el siguiente renglón en un archivo llamado index.php:

<?phpinfo()?>

Mueva este último archivo al directorio raíz donde residen sus páginas HTML , este directorio correspondería aquel definido en el parámetro DocumentRoot en httpd.conf, el cual de no haberse realizado ningún cambio estaría en /usr/local/apache2/htdocs/
Finalmente, intente visitar este documento en su Navegador ("Netscape","Opera" u otro) utilizando la dirección definida en ServerName, ejemplo: http://miservidor.com/index.php.
Al visitar esta última página debe observar todos los parámetros de configuración para el módulo de PHP, lo cual indica que ha quedado instalado correctamente el módulo.

17.3 Instalación de PHP con XML-RPC.

Es imprescindible instalar expat, antes de instalar el soporte XML-RPC de PHP. El proyecto expat, que encuentra en http://sourceforge.net/projects/expat es una librería programada en C cuyo objetivo es hacer de "parser" (intérprete) de XML.

Instalación:
mkdir/home/install/programacion/php/
cd /home/install/programacion/php/
wget "http://unc.dl.sourceforge.net/sourceforge/expat/expat-1.95.4.tar.gz"
cd /usr/src/
tar -zxvf /home/install/programacion/php/expat-1.95.4.tar.gz
rm -Rf expat
ln -s expat-1.95.4 expat
cd /usr/src/expat/
make clean
rm -f config.status config.cache
./configure
make
make install
echo "/usr/local/lib" >> /etc/ld.so.conf
ldconfig

Una vez que se encuentre instalado expat, compile PHP nuevamente:
Compile PHP4 como modulo de Apache
cd /usr/src/php/
make clean
rm -f config.status config.cache
./configure \
--with-apache=/usr/src/apache \
--with-zlib \
--enable-sockets \
--with-kerberos=/usr/kerberos \
--enable-inline-optimization \
--with-expat-dir=/usr/local \
--with-xmlrpc \
--enable-overload \
--with-imap
make
make install

Hay algunas opciones imprescindibles (with-expat-dir, --with-xmlrpc, ...) para disponer de soporte XML-RPC en PHP. Otras van en función de los proyectos que cree y sus requisitos.

17.4 SOAP sobre PHP

Ya ha visto como funciona XML-RPC sobre PHP. Otro protocolo de web services se llama SOAP. SOAP también puede ser utilizado desde PHP. Aunque XML-RPC es más sencillo, hay que reconocer que la mayoría de los web services públicos e interesantes de hoy en día, están definidos bajo el estándar SOAP. Es por eso, especialmente interesante saber como invocar uno de estos servicios desde SOAP.

Como ha podido ver con XML-RPC, crear un web service es más complicado que invocarlo. En SOAP pasa lo mismo. Observe la parte sencilla, y la más útil, o sea, su invocación.

La implementación de SOAP sobre PHP más recomendable es NuSOAP, que se encuentra en http://dietrich.ganx4.com/nusoap/index.php y puede descargar en http://dietrich.ganx4.com/nusoap/downloads/nusoap-0.6.1.zip

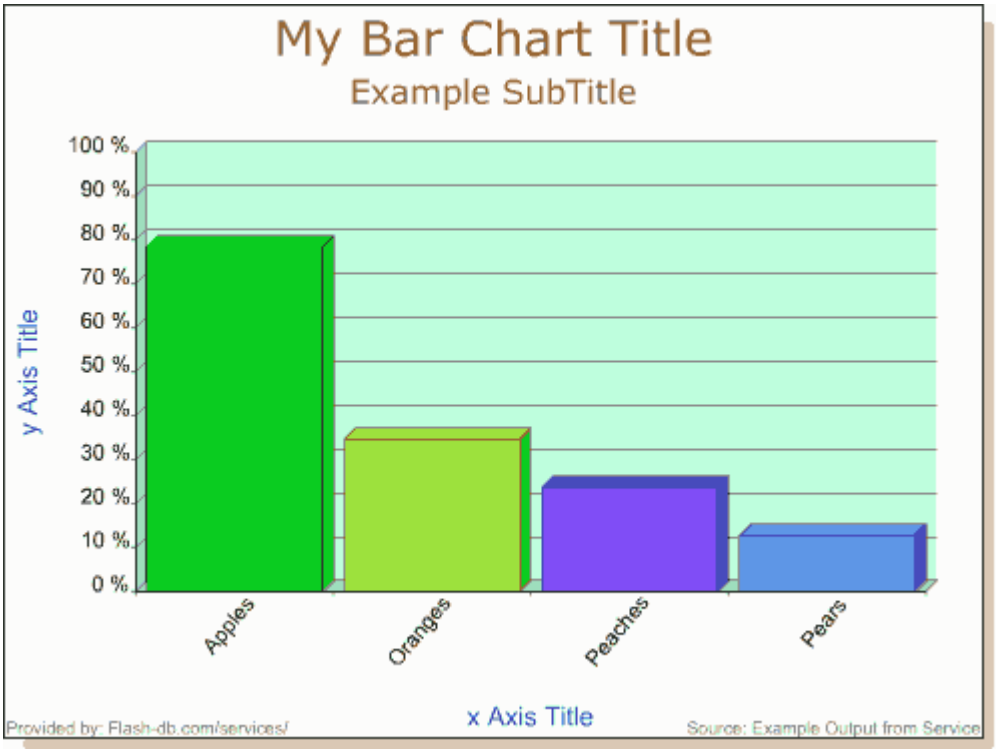
Está desarrollada por la empresa NuSphere (http://www.nusphere.com/) y licenciada bajo licencia GPL.

Para ponerla en marcha digite este script en PHP de ejemplo:

```
#!/usr/local/bin/php -q
<?php
require_once('nusoap-0.6.1/nusoap.php');
$soapclient = new soapclient('http://services.xmethods.net/soap/urn:xmethods-delayed-quotes.wsdl','wsdl');
echo $soapclient->call('getQuote',array('symbol'=>'ibm'));
?>
```

Así de sencillo es obtener la cotización en bolsa de las acciones de IBM en un programa en PHP utilizando un web Service SOAP. WSDL es equivalente a la URL en los servicios XML-RPC. Es la dirección de Internet donde hay que invocar el proceso que nos interesa.
Referencias: "Construcción de Servicios Web con SOAP" http://www.revista.unam.mx/vol.3/num1/art3/
Referencias: "WSDL" => http://www.w3.org/TR/2001/NOTE-wsdl-20010315

Ejemplo:
un programador Web puede considerar muy útil, una rutina que produzca gráficas de datos numéricos en formato Flash. Existe un Servicio Web que hace esto: http://www.flash-db.com/serviceHelp/flashCharts.php



Implementar en PHP una llamada al generador de gráficas en Flash:

```
#!/usr/local/bin/php -q
<?php
require_once('nusoap-0.6.1/nusoap.php');

// Definimos los parámetros de entrada
$parameters = array(

    "mainTitle"      =>"My Bar Chart Title",
    "subTitle"       =>"Example SubTitle",
    "xAxisTitle"     =>"x Axis Title",
    "yAxisTitle"     =>"y Axis Title",
    "sourceTitle"    =>"Source: Example Output from Service",
    "minValue"       =>"0",
    "maxValue"       =>"100",
    "unitType"       =>" %",
    "drawGrid"       =>"1",
    "animate"        =>"1",
    "show3D"         =>"1",
```

```
    "dataArray"       =>"Apples:78.5,Oranges:34.8,Peaches:23.7,Pears:12.9"
);
// Se indica la URL del servicio SOAP mediante WSDL
$soapclient = new soapclient(' http://www.flashdb.com/services/ws/flashBarChart.wsdl' ,
' wsdl' );

// Se invoca al Servicio Web SOAP
$result     = $soapclient->call(' doFlashBarChart' ,$parameters);

// El resultado viene en formato binario base64 ...
$binary = base64_decode($result);

// Grabamos el archivo flash en el filesystem
$movieName = "./NewChart.swf";
$fp = fopen($movieName,"w");
fwrite($fp, $binary, 12000);
fclose( $fp );
?>
```

El resultado obtenido, se puede visualizar en un navegador usando el siguiente código HTML:

```
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#versi
on=5,0,0,0"
  WIDTH="500" HEIGHT="400">
<PARAM NAME=movie VALUE="NewChart.swf">
<PARAM NAME=quality VALUE=high>
<PARAM NAME=bgcolor VALUE=#FFFFFF>
<EMBED src="NewChart.swf" quality=high bgcolor=#FFFFFF WIDTH="500"
HEIGHT="400"
TYPE="application/x-shockwave-flash"
PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer">
</EMBED>
</OBJECT>
```

Otro ejemplo más, de gran utilidad es la invocación a Google desde PHP. Google uno de los buscadores de más prestigio de la red ha incorporado recientemente la posibilidad de poder usar su base de datos de páginas desde cualquier herramienta de desarrollo.

El producto denominado "Google Web API" permite a los desarrolladores poder interrogar y tomar información de casi tres mil millones de documentos Web directamente desde Google. Para lograr eso, Google usa SOAP y WSDL de forma que los desarrolladores puedan programar en su entorno favorito tal como PHP, Java, Perl, o Visual Studio .NET.

Aprovechando la potencia de Google en un script de PHP:

```
#!/usr/local/bin/php -q
<?php
require_once('nusoap-0.6.1/nusoap.php');

// Licencia de Google
$key = 'iwnUXUtHj3bteg5FWfBJDwui3SPeB+iy';

// Por ejemplo ...
$Query          = "linux+programación";
$startPage      = 0;

$parameters = array(
        'key'       => $key,
        'q'         => $Query,
        'start'     => $startPage,
        'maxResults'=> 10,
        'filter'    => false,
        'restrict'  => '',
        'safeSearch'=> false,
        'lr'        => '',
        'ie'        => '',
        'oe'        => ''
);

$soapclient = new soapclient('http://api.google.com/search/beta2');
$result = $soapclient->call('doGoogleSearch',$parameters, 'urn:GoogleSearch');
print_r ($result);
?>
```

El resultado es algo así:

```
Array
(
   [documentFiltering] => false
   [estimatedTotalResultsCount] => 77400
   [directoryCategories] =>

   [searchTime] => 0.194757
   [resultElements] => Array
      (
         [0] => Array
            (
               [cachedSize] => 28k
               [hostName] =>
               [snippet] => Programas y Aplicaciones de GNU/Linux
               [directoryCategory] => Array
                  (
                     [specialEncoding] =>
                     [fullViewableName] =>
                  )

               [relatedInformationPresent] => true
               [directoryTitle] =>
               [summary] =>
               [URL] => http://linux.bankhacker.com/software/Programacion/
               [title] => Linux Programacion
            )


[1] => Array
            (
               [cachedSize] => 9k
               [hostName] =>
               [snippet] =>  Hosting Linux y Programación
               [directoryCategory] => Array
                  (
                     [specialEncoding] =>
                     [fullViewableName] =>
                  )

               [relatedInformationPresent] => true
               [directoryTitle] =>
               [summary] =>
               [URL] => http://linux.bankhacker.com/
               [title] => Software Linux: Programas y Aplicaciones de GNU/Linux
            )
```

```
            ...
        )

    [endIndex] => 10
    [searchTips] =>
    [searchComments] =>
    [startIndex] => 1
    [estimateIsExact] => false
    [searchQuery] => linux+programación
)
```

Servidores SOAP con PHP:

Ya hemos visto como utilizar los web services basados en SOAP con PHP Pero ¿cómo construir un web service basado en SOAP? Ejemplo:

```
#!/usr/local/bin/php -q
<?php
    require_once('  nusoap-0.6.1/nusoap.php'  );

    $server = new soap_server;
    $server->register('  hello'  );
    function hello ($name){
        return "Hello $name.";
    }
    $server->service($HTTP_RAW_POST_DATA);
?>
```

18. JAVA.

18.1 Arquitectura J2EE

La arquitectura J2EE proporciona componentes y una plataforma independiente del ambiente en la que cada componente es ensamblado y desarrollado en el contenedor correspondiente. El servidor J2EE proporciona servicios de transacción, administración y otros detalles de bajo nivel. Ya que los servicios que proporciona en servidor J2EE son desarrollados para solventar los problemas empresariales o de negocios.

La arquitectura J2EE proporciona las APIs siguientes:

- Java Naming and Directory Interface (JNDI): Proporciona nombre y direcciona funcionalidad a aplicaciones escritas en Java.
- Remote Method Invocation (RMI), se puede crear distribuciones Java con la tecnología basada en aplicaciones, en cada método de objetos remotos de Java que pueden ser invocados por aplicaciones escritas en otra maquina virtual de Java (JVM) posiblemente en diferentes host.
- Java Database Connectivity (JDBC), puede accesar algún origen de datos como una base de datos, hojas electrónicas, o archivos planos que utilicen el lenguaje de programación Java.
- Java Transaction API (JTA): especifica interfaces estándar Java entre transacciones administradas que son envueltas por un sistema distribuido de transacciones ejemplo: the resource manager, the application server, and the transactional applications.
- Java Menssagin Services (JMS): este proporciona aplicaciones para crear, enviar, recibir y leer mensajes definidos por un set de interfaces y una semántica asociada que aloja programas escritos en el lenguaje de programación Java para comunicarse con otras implementaciones de lenguajes.
- JavaMail: proporciona un conjunto abstracto de clases con un sistema de correo. API proporciona una plataforma y un protocolo independiente para construir tecnología Java basada en mensajes de correo y aplicaciones.
- JavaBeans Activation Framework: proporciona servicios estándar para determinar el tipo de datos, el acceso al encapsulamiento, las operaciones de descubrimiento están habilitadas, y son apropiadas para el buen rendimiento de las operaciones, ejemplo si un navegador web obtiene una imagen JPEG, esta puede habilitar el navegador para identificar que se trata de un tipo de datos imagen JPEG y este puede localizar en el instante el objeto para manipular, abrir o visualizar esta imagen.

18.2 J2EE Server

J2EE Server es la parte del tiempo de ejecución del producto J2EE. J2EE server provee EJB y contenedores web. Una de las ventajas del estándar J2EE es que brinda un contenedor simple y fácil para los desarrolladores ya que el ensamble de aplicaciones es el mismo que en cualquier compilador que utilice un ambiente de trabajo J2EE.

J2EE server proporciona un sistema de niveles de servicios para los componentes de aplicaciones J2EE:

- Los nombres de los servicios, proveen los componentes de aplicaciones J2EE accesando a JNDI lookup services. Para usar JNDI lookup services los componentes pueden ser ubicados por los objetos definidos por el usuario, EJB, JDBC objetos de origen de datos y conexiones de mensajes.
- Puede usar un servicio de transacciones para especificar las relaciones y métodos que constituyen una transacción o todos los métodos que sean tratados como una sola transacción.
- La conectividad remota es administrada por comunicación de bajo nivel entre clientes y EJB. Esto provee transparencia en ubicación para los clientes. Los clientes accesan los EJB como si ellos iniciaran o si fueran de la misma maquina virtual de Java.

Ejemplo de J2EE servers son Oracle 9i Application Server, BEA Web Logic, IBM WebSphere.

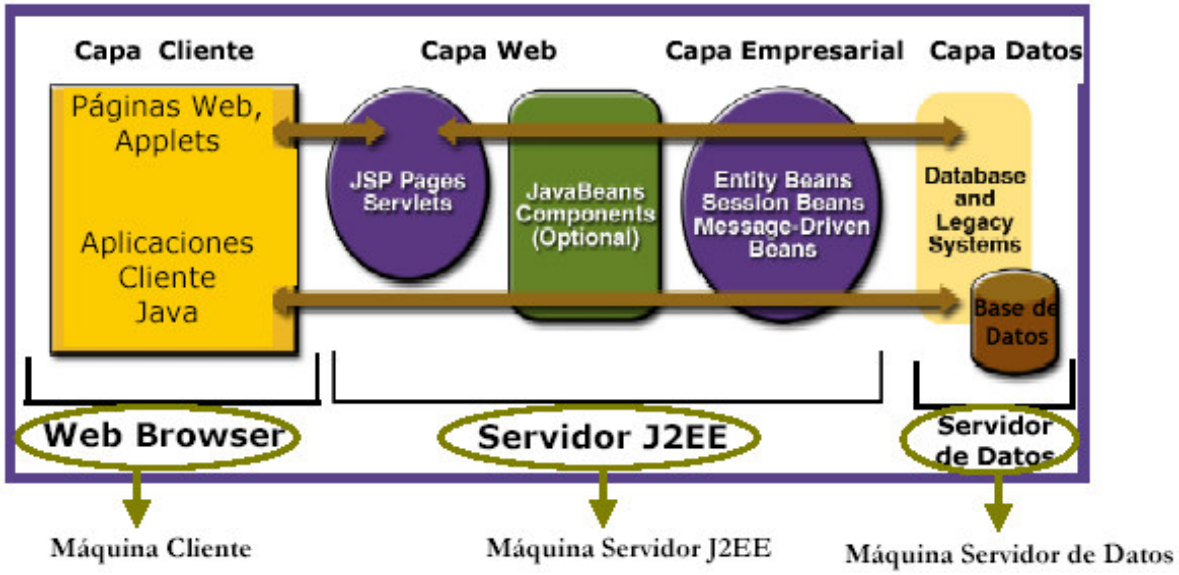
Contenedores J2EE:
La arquitectura J2EE proporciona los siguientes contenedores:

- Contenedores de aplicaciones clientes en host como applets o aplicaciones stand-alone. Los elementos de aplicaciones J2EE pueden ser clientes para otros elementos de las aplicaciones.
- Los componentes host de contenedores web tales como servlets y JSP, proveen J2EE services. Los contenedores web que contienen applet que son ejecutados en clientes.
- Enterprise JavaBeans (EJB) contiene componentes tales como session beans, entity beans y message-driven beans.
- Los contenedores J2EE pueden personalizarse para naming, deployment, transaction, security, y messaging services.

18.3 Enterprise JavaBeans (EJB)

Es una arquitectura para desarrollo de aplicaciones transaccionales como componentes distribuidos de Java. EJB es una herramienta poderosa de desarrollo que usa una metodología para el desarrollo de aplicaciones distribuidas. Para desarrollar aplicaciones con enterprise beans, los desarrolladores de beans nunca conocen las necesidades de las aplicaciones cliente en lo que se refiere a detalles tales como transaction support, security, remote object acces y otros aspectos. Estos proporcionan transparencia para el desarrollador por el contenedor EJB.

EJB ofrece portabilidad. Un bean es desarrollado en un contenedor EJB que puede estar ejecutando otros EJB para conocer especificaciones EJB, brindadas por el servidor y el contenedor para los host de Enterprise JavaBeans.



Una Aplicación J2EE está constituida por múltiples componentes que interactúan. Un Componente J2EE es una unidad de software autocontenida escrita en Java que se ensambla en una aplicación J2EE junto con sus clases y archivos relacionados y que se comunica con otros componentes. Se clasifican en:

- Componentes Cliente: Clientes Web, Applets y Aplicaciones Cliente; se ejecutan en la máquina cliente.
- Componentes Web: Servlets y JavaServer Pages (JSP); se ejecutan en una máquina servidor HTTP (o servidor de aplicaciones J2EE)
- Componentes Empresariales: Enterprise JavaBeans; se ejecutan en una máquina servidor de aplicaciones J2EE.

Los componentes J2EE se escriben en JAVA y se compilan de la misma manera que un programa cualquiera JAVA. La diferencia entre componentes y clases estándares es que los componentes se ensamblan en una aplicación J2EE, se ejecutan y administran por un contenedor.

Existen 4 modelos de componentes, que a su vez puede combinar para construir otras aplicaciones:

Aplicaciones clientes:

- Aplicaciones Java puras, similares a las aplicaciones nativas.
- Se ejecutan en el cliente.
- Normalmente son un GUI.

Las aplicaciones que se basan en aplicaciones J2EE usan un navegador web como componente del cliente. El navegador web descarga paginas estáticas o dinámicas desde la web hacia la maquina del cliente.Las paginas dinámicas son generadas por servlets y JSPs desde la web. Las paginas JSP no requieren Java plug-ins o archivos de seguridad para ser descargados a la maquina del cliente.

Al descargar una página web puede contener un applet que se ejecute en Java Virtual Machine (JVM) en el navegador web. Un applet requiere un plug-in Java y la seguridad de archivo para ejecutarse con éxito por el usuario en el navegador web.

Los clientes web, típicamente no necesitan operaciones que consuman mucho rendimiento como ejecutar los parámetros para establecer comunicación entre entidades o conectar y ejecutar consultas en bases de datos. Estas operaciones generalmente son hechas por business-tier components.}

Applets:
Componentes GUI que normalmente ejecutan en Navegadores.
Permiten proveer GUI más poderosas que el simple HTML

Los applets son pequeños programas que requieren de un navegador para ejecutarse (Netscape, Microsoft Explorer,Mozilla,HotJava).

Las Applets están disponibles mediante páginas web descriptas en el lenguaje HTML (HyperText Markup Language). La etiqueta (tag) <APPLET> provee al navegador información acerca del applet(tamaño, parámetros de entrada). cuando se accede a una página web que contiene un Applet, éste se descarga automáticamente desde un servidor HTTP y, se ejecuta en la computadora del cliente. tanto los applets como las Aplicaciones requieren de la plataforma JAVA para ejecutarse. Sin embargo, applets requieren que la plataforma JAVA esté embebida en el navegador mientras que las Aplicaciones requieren que la plataforma JAVA esté disponible a través del sistema o como un programa separado.
En términos generales, las Applets y las Aplicaciones acceden a las mismas API´s.

Las Aplicaciones tienen acceso ilimitado a los recursos del sistema local: pueden leer y escribir en el FileSystemlocal y remoto. Sin embargo, los applets al ser descargados a través de la red están restringidos a acceder solamente al FileSystem del servidor de origen (remoto). Es la política de seguridad de sandbox.

Incluir un applet en una página HTML:

```
<HTML>
<HEAD>
<TITLE>Un programa Simple</TITLE>
</HEAD>
<BODY>
<img src="http://163.10.10.5/dibu.gif" align=bottom>
<APPLET CODE="HolaMundo.class"  WIDTH=150 HEIGHT=25>
</APPLET>
</BODY>
</HTML>
```

Algunos atributos del tag <APPLET>:

CODEBASE	Especifica el directorio o URL que contiene las applets
CODE	Especifica el nombre del archivo .class que contiene el applet
WIDTH	Especifica el ancho de la ventana del applet
HEIGHT	Especifica el alto de la ventana del applet
ALT	Especifica el texto que se mostrará en un browser
NAME	Especifica un nombre para la instancia applet
PARAM	Especifica parámetros para el applet

Servlets: Un servlet es un programa Java que se ejecuta en un servidor como lo es Oracle 9i Application Server, y produce paginas dinámicas como respuestas a peticiones cliente HTTP. Las paginas son enviadas de regreso al cliente a través del navegador web. Las peticiones pueden ser como una URL HTTP. Un servlet puede enviar una petición a otro servlet, JSP o un EJB, típicamente los servlets dan una respuesta como resultado HTML, pero varios tipos de datos pueden ser usados incluyendo XML.

JavaServer Pages (JSP): JSP puede ser HTML u otro lenguaje de marcado de páginas que este dentro del código Java. El código de Java genera HTML y se coloca entre etiquetas <%= and %> o <% and%> que se evalúan en tiempo de ejecución. JSP es compilado dentro de servlets por el compilador JSP. El servidor web invoca este servlet y devuelve la respuesta desde el servlet para el cliente.
Componentes Web:
Pueden ser páginas JSP, Servlets.
Normalmente ejecutan en un servidor Web y responden a pedidos HTTP realizados por clientes Web.

Generalmente se utilizan para generar páginas HTML dinámicamente que sirven de GUI. También se pueden utilizar para generar XML u otros formatos que sean consumidos por otros componentes.
Ej.: Apache SOAP, Apache AXIS

J2EE Web-tier components puede ser cada pagina JSP, paginas XML, WML, o servlets que generen HTML dinámico o estático. Java servlets proporciona una simple pero poderosa API para generar páginas web, al procesamiento dinámico de las peticiones cliente. JSP simplifica los procesos de generación dinámica usando JAVA y scripting language baja paginas HTML.

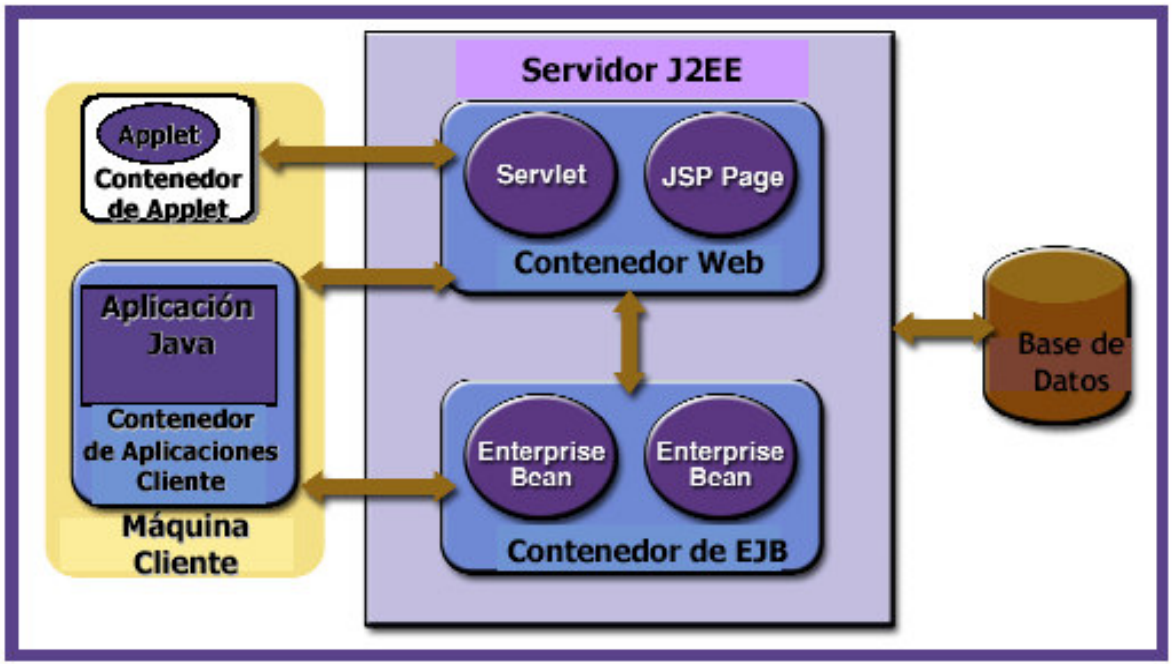
Los componentes web pueden usar Java Database Connectivity (JDBC) para accesar a los registros en las tablas de una base de datos, entre las ventajas de usar componentes web están las siguientes:
La interfaz HTML es generada por componentes web en contraste a apllets que requiere descargas para los clientes en tiempo de ejecución. También la interfaz HTML no requiere una preinstalación en las máquinas cliente.
El protocolo HTTP que es usado por clientes para requerir y recibir páginas web puede atravesar firewall sin ningún problema, en contraste a otros protocolos que usan el método de invocación remota (RMI).

Componentes EJB:
Típicamente contienen la lógica de negocio de aplicaciones J2EE
Componentes que se ejecutan en un ambiente controlado con soporte transaccional y de persistencia EJB Container
Normalmente ubicados en la capa media de una aplicación
NO son componentes con GUI
Existen tres tipos de componentes
Session Bean
Entity Bean
Message Driven Bean

Enterprise JavaBeans son componentes que son ejecutados como business tier. Estos componentes distribuidos contienen lógica de negocios que resuelven o conocen las necesidades de un dominio en particular como un banco, orden de entradas de sistema o recursos humanos. Los componentes Business-tier pueden recibir datos desde programas clientes, datos de procesos y enviar datos procesados a servidores de bases de datos para ser almacenados. Los componentes Business-tier también reciben información desde bases de datos, procesos y envían resultados a programas clientes.

Los componentes Business-tier pueden invocar otros componentes business-tier donde otros procesos lógicos estén en espera, ejemplo un servlet puede invocar un entity enterprise bean para insertar un nuevo detalle de cliente y devolver algún dato procesado como resultado a un programa cliente.

Contenedores



Los contenedores interactúan con los componentes invocando los métodos de administración o métodos call-back. Estos métodos definen la interface de comunicación entre el Contenedor y los componentes.

En la arquitectura J2EE todos los componentes se ejecutan en el entorno de un contenedor. Un Contenedor es un ambiente de ejecución que provee de servicios a las componentes tales como conectividad con la red, administración del ciclo de vida de los componentes, seguridad, y servicios especiales como transaccionalidad, persistencia, pooling,etc. Los contenedores proveen un ambiente estándar de ejecución, vista federada de APIs J2EE y Servicios específicos para los componentes.

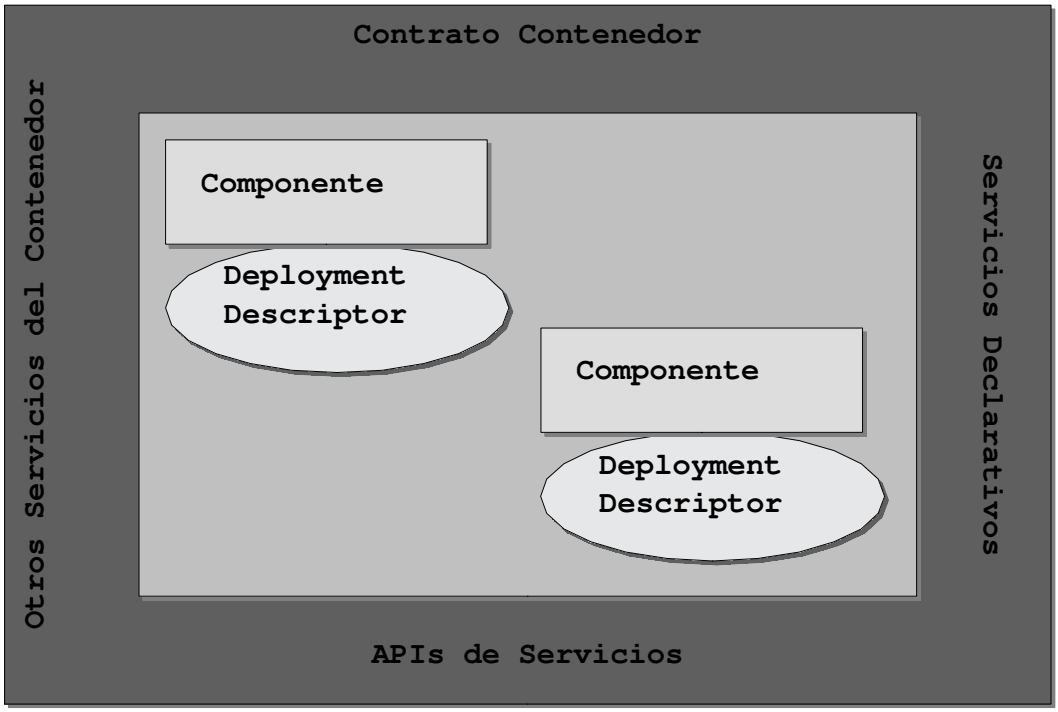
Los componentes pueden asumir que el contenedor, junto a sus servicios y APIs van a estar disponibles en cualquier implementación de la Plataforma J2EE. (Portabilidad)

La arquitectura J2EE proporciona los siguientes contenedores:

- Contenedores de aplicaciones clientes en host como applets o aplicaciones stand-alone. Los elementos de aplicaciones J2EE pueden ser clientes para otros elementos de las aplicaciones.
- Los componentes host de contenedores web tales como servlets y JSP, proveen J2EE services. Los contenedores web que contienen applet que son ejecutados en clientes.
- Enterprise JavaBeans (EJB) contiene componentes tales como session beans, entity beans y message-driven beans.

- Los contenedores J2EE pueden personalizarse para naming, deployment, transaction, security, y messaging services.

La arquitectura del contenedor consta de cuatro partes:



- Contrato con el Componente
- APIs de Servicios
- Servicios Declarativos
- Otros Servicios del Contenedor

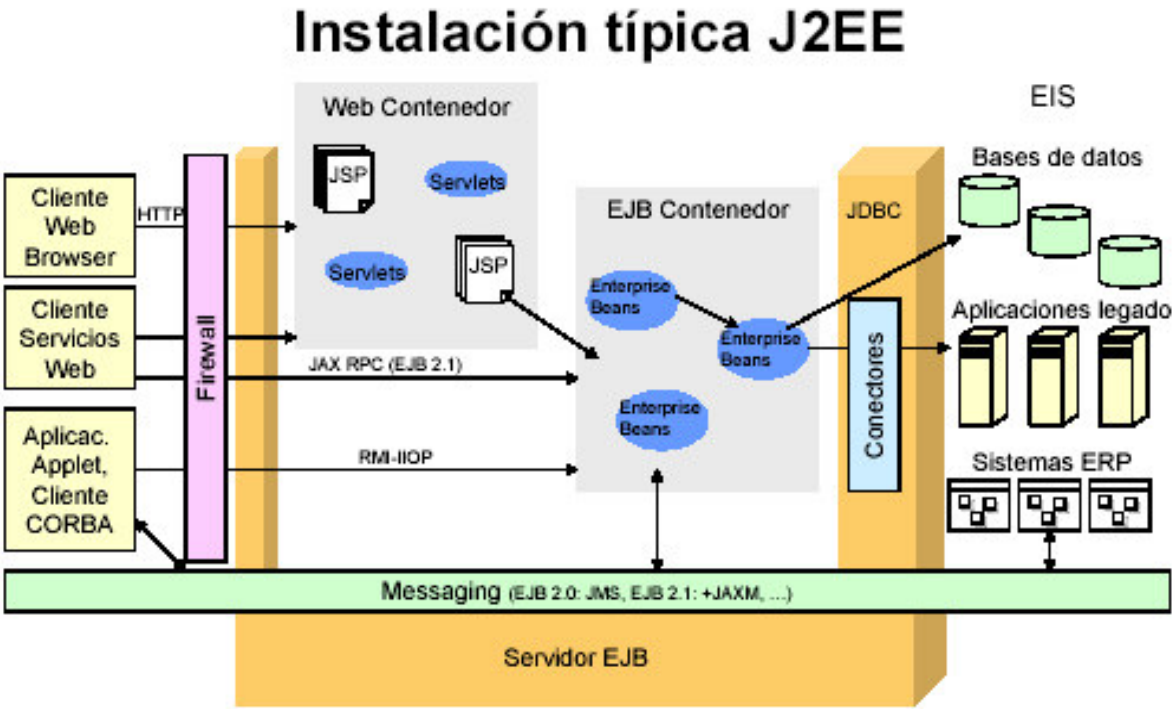
El Contrato con el Componente especifica un conjunto de APIs que los componentes que residen en dicho contenedor deben implementar. Dichas APIs definen el mecanismo de comunicación entre el contenedor y el componente.

Las APIs de Servicios proveen acceso a las extensiones estándar definidas por J2EE.
Ejemplo: JDBC, JTA, JNDI, JMS, etc

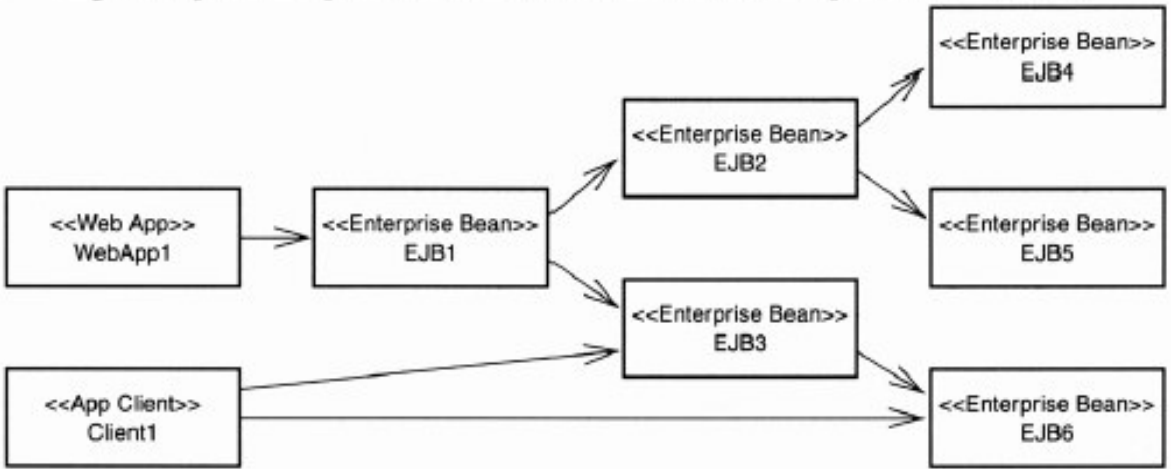
Los Servicios Declarativos permiten especificar el uso de servicios o comportamientos de un componente declarativamente por fuera de la implementación del mismo. Ej. de estos servicios pueden ser: Manejo transaccional, seguridad, etc

Los Otros Servicios que provee el contenedor son: manejo del ciclo de vida de un componente, pooling de recursos (Ej. BDs), inicializar el servicio JNDI registrando los componentes EJB o servicios de Clustering.

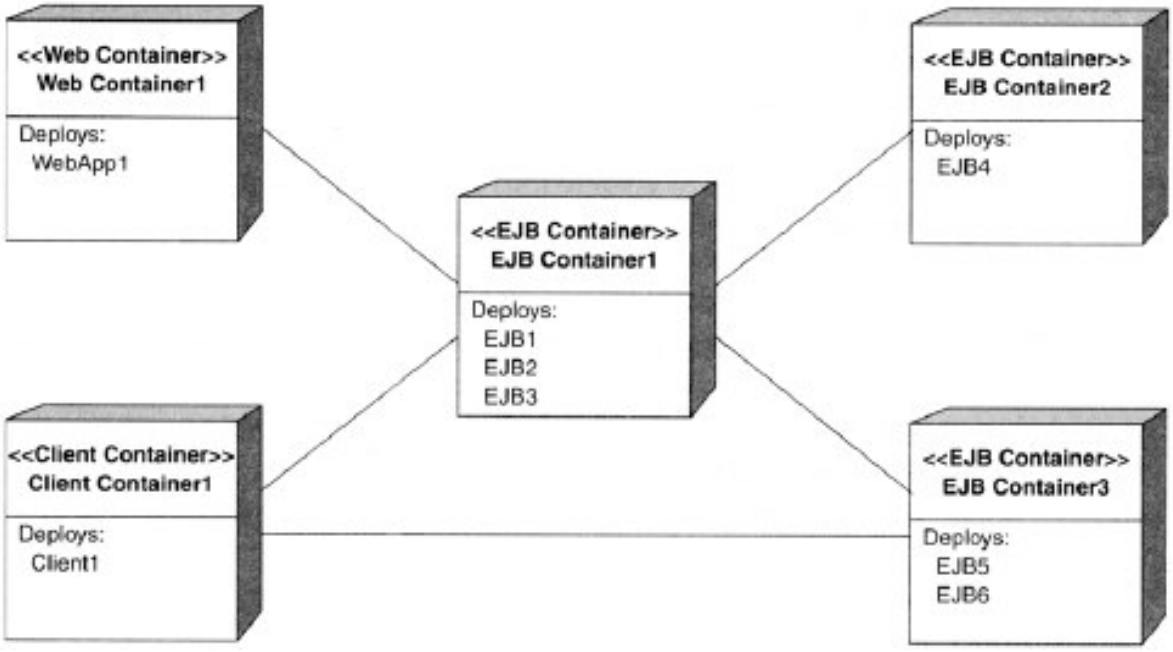
EJB son los componentes usados como parte de aplicaciones corporativas distribuidas, cada bean encapsula parte de la lógica de negocio, comunicándose con gestores de recursos, y con otros; accedido por distintos tipos de clientes: EJBs, servlets, clientes de aplicación, etc.
En tiempo de ejecución, reside en un contenedor EJB: servicios de seguridad, transacción, instalación (deployment), concurrencia, y gestión del ciclo de una aplicación puede tener uno o varios EJBs en uno o varios contenedores EJB.

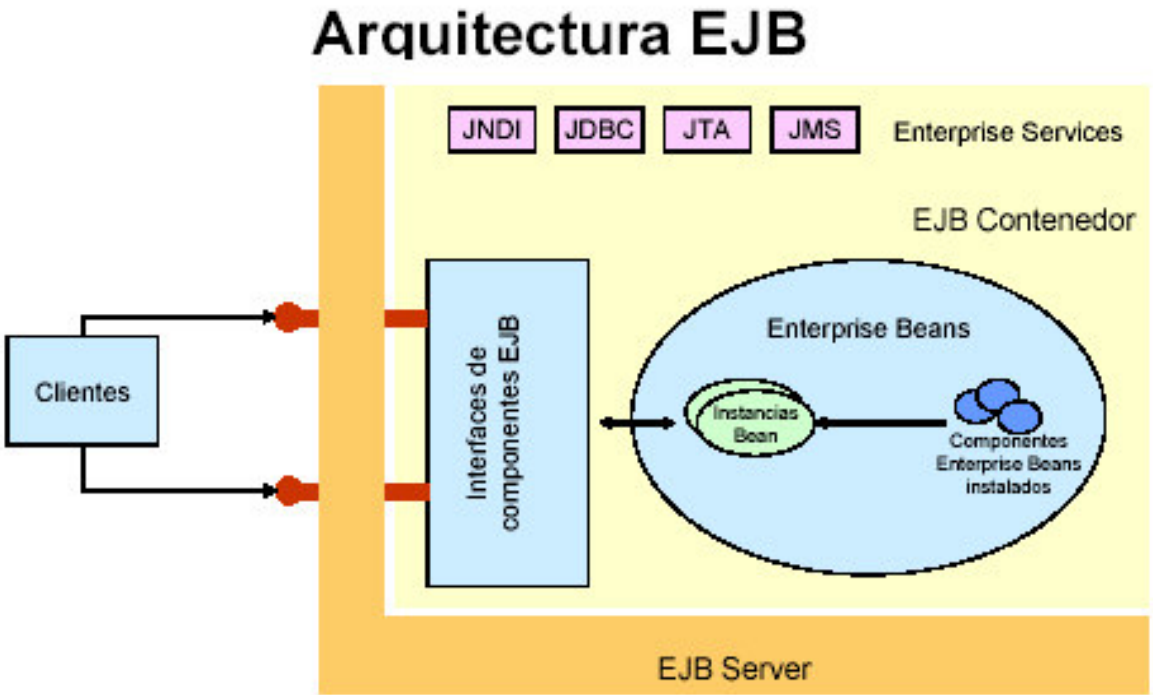


Ejemplo aplicación con múltiples EJBs



Instalación en múltiples contenedores





18.9 Elementos de la arquitectura EJB

EJB: gestión de conexiones, errores, disponibilidad, balance de carga y escalabilidad. Conexión con la estructura de comunicación y vigilancia de los procesos y threads que corren en el contenedor.

Contenedor EJB: ofrece un entorno de ejecución a los componentes instalados: manejo de las instancias-Bean, control del ciclo de vida y acceso a los servicios Enterprise estándar
El conjunto Servidor EJB y Container EJB proporciona un servidor de aplicaciones

Enterprise Beans: son los componentes de servidor en los que está encapsulada la lógica de negocio. Hay tres tipos:
– Session Beans representan procesos de negocio
– Entity Beans representan datos de negocio
– Message-Driven Beans procesan mensajes asíncronos
Clientes: a EJB pueden acceder clientes basados en browser, clientes Java, Servlets, clientes CORBA u otros Enterprise Beans

Entidades de negocio
Objeto de negocio que representa información mantenida por la empresa
– Estado, mantenido en la BD (persistente)
– Ejemplos: cliente, pedido, cuenta, empleado
– Las reglas de negocio asociadas con una entidad
• Restringen los valores de su estado (Ej.: código postal 5

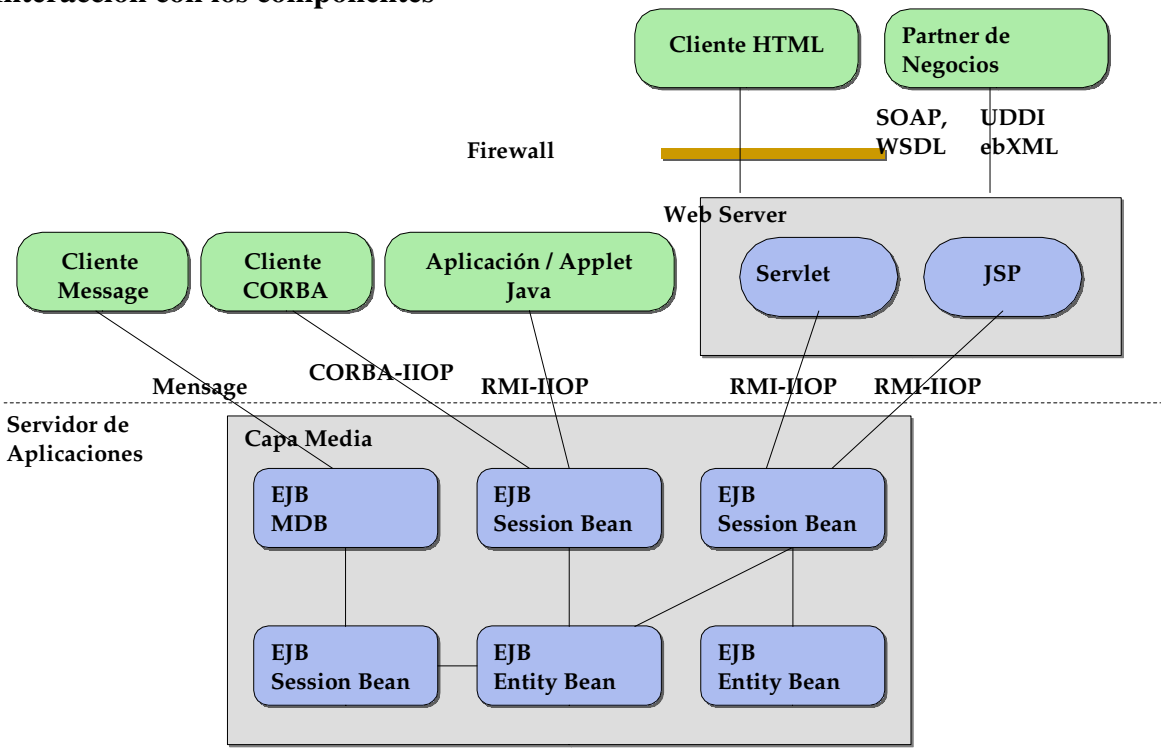
cifras)
• Mantienen relaciones entre entidades (Ej.: un cliente con varios pedidos)

Procesos de negocio
Objeto de negocio que encapsula una interacción de un usuario con una entidad de negocio
– Actualiza el estado de una entidad de negocio
– Puede tener su propio estado
• Estado persistente: proceso en varios pasos realizado por distintos actores: proceso de negocio colaborativo
– Ejemplo: procesamiento de una solicitud de préstamo
• Estado transitorio: proceso de negocio completado por un actor durante una conversación: proceso de negocio conversacional
– Ejemplo: sacar dinero de un cajero automático

Reglas de negocio
Distribuidas entre los componentes que implementan las entidades y procesos de negocio
En función de si la regla aplica a la entidad o al proceso
Ejemplo entidad: el balance de una cuenta no puede ser negativo (es independiente del proceso que lo cause)
– Ejemplo proceso: no se pueden sacar más de $500 de un cajero automático (independiente de la entidad Cuenta)

Parte de un EJB
Bean Class: Clase común que implementa un conjunto de operaciones bien definidas. Implementa: operaciones para interactuar con el contenedor y los servicios que el componente exporta
Remote Interface: Interfaz que exporta todos los métodos de negocio que un componente exporta y que pueden ser invocados desde fuera
Home Interface: Permite manejar la creación, destrucción y búsqueda (solo "entities") de instancias del componente. También permite que la ubicación de los componentes se maneje transparentemente
Local Interface: Similar a Remote, pero evita el overhead de las invocaciones remotas ya que no pasa por los stubs, la red, el marshaling y unmarshaling de parámetros. Pensado para las invocaciones entre Beans en el mismo servidor de aplicaciones
Local Home Interface: La función es la misma que la de la Home interface remota. Las invocaciones a la misma deben ser locales.

Interacción con los componentes



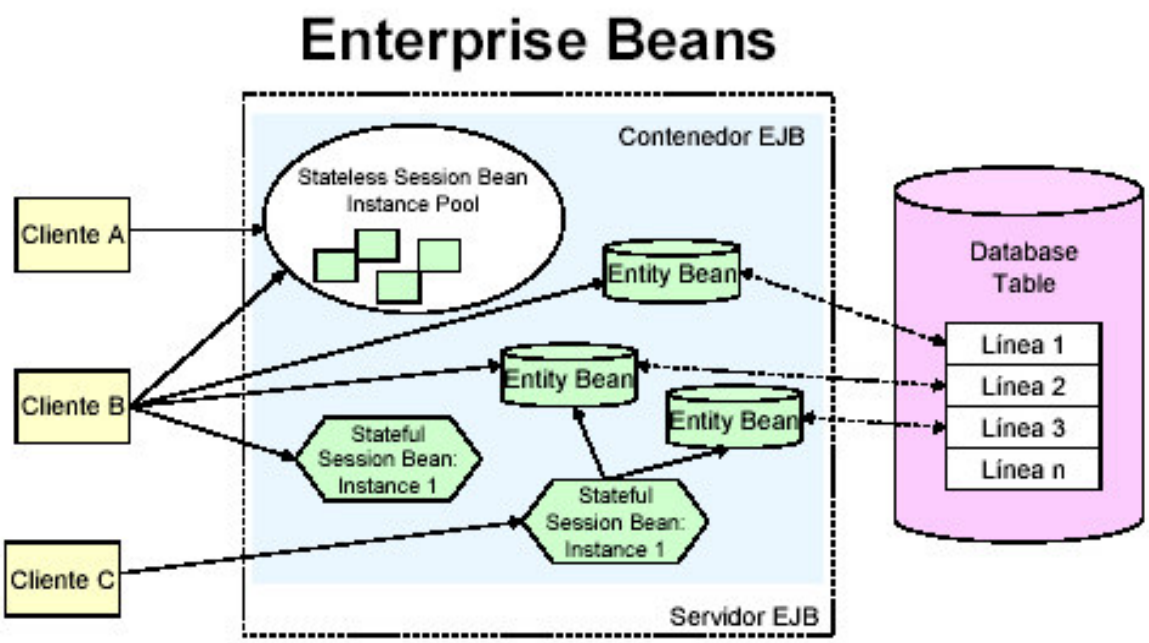
18.10 Enterprise Beans

Componentes de servidor (especificación EJB), implementa la lógica de negocio
Mantienen una identidad única durante todo su ciclo de vida

Entity Beans
– modelan conceptos de negocio como objetos persistentes asociados a datos
– Soporte pasivo de información que ofrece métodos para las operaciones sobre los datos (Ej. Cuenta bancaria, producto, pedido…)
Llamada síncrona

Session Beans:
Representan procesos ejecutados en respuesta a una solicitud del cliente (Ej. Transacciones bancarias, cálculos, realización de pedidos…)
– Típicamente usan Entity Beans para el procesado de datos
– Llamada síncrona

Message-Driven Beans:
Representan procesos ejecutados como respuesta a la recepción de un mensaje
Llamada síncrona



Entity Beans

Los entity beans saben como persistirse en almacenamiento secundario y permiten visualizar una representación orientada a objetos de la información.

Normalmente un EB representa un subconjunto del dominio del problema, es general información con alta cohesión.

Ejemplo: podría representar una línea de una factura, una factura entera o todas las facturas realizadas en un día.

Existen 2 tipos de EB, difieren en quien es responsable de la persistencia:
Bean Managed Persistent (BMP): el desarrollador del Bean es responsable de escribir el código para persistirlo.
Container Managed Persistent (CMP): el contenedor es responsable de manejar la persistencia.

Es posible establecer relaciones entre las entidades del sistema, estas relaciones pueden ser de dos tipos:

Bean Managed Relationship: puede relacionar BMP y CMP
Container Managed Relationship: solo puede relacionar EB CMP
Las relaciones pueden ser de tres tipos:
Uno a Uno
Uno a Muchos
Muchos a Muchos

Session Beans
La diferencia fundamental de los Session Beans con los Entity Beans: es el tiempo de vida, ya que los primeros solo duran el tiempo que permanece la sesión del cliente. En definitiva son objetos en memoria.

Existen 2 tipos:

Stateful Session Beans: El estado conversacional se mantiene mientras la sesión con el cliente esté viva. Se utilizan fundamentalmente para representar procesos de negocio que se expanden a través de varias invocaciones a métodos.

Ej. carrito de compras del Web

Stateless Session Beans: El estado conversacional se mantiene por el tiempo que dura la invocación a un método. Luego de cada invocación en contenedor puede optar por: destruir el Bean, recrearlo o inicializarlo. Estos normalmente deben recibir del cliente toda la información necesaria para realizar su tarea u obtenerla de una fuente externa como una base de datos

Elección de Entity Bean o Session Bean
Las entidades de negocio se implementan típicamente como Entity Beans, los procesos de negocio conversacionales se implementan típicamente como Session beans, procesos de negocio colaborativos se implementan típicamente como Entity Beans
– El estado representa los pasos intermedios realizados

Message Driven Beans
Es una solución para introducir asincronismo al modelo de componentes de EJB. No tienen estado conversacional
Los MDB son un tipo especial de EJB que pueden recibir mensajes de colas o tópicos generados por un cliente JMS, el cliente está completamente desacoplado del MDB, los MDB están suscritos a una cola o tópico, cuando llega un mensaje el contenedor despierta al MDB para que procese el mensaje. Son receptores de mensajes intermediario entre cliente emisor y Message-driven
Bean (servicio de mensajería), la diferencia con Session y Entity Beans es la comunicación asíncrona en lugar de llamada síncrona a métodos (el cliente bloquea hasta el fin de la llamada).

Características de los MDB:

- No tienen home, remote, local o local home interface
- No es posible invocarlo directamente
- Se invoca enviando un mensaje JMS
- Solamente tienen un método de negocio
- Es el invocado por el contenedor cuando llega un mensaje
- El método se llama onMessage( ) que recibe un mensaje

- No retornan un valor al cliente
- No pueden levantar excepciones al cliente
- Los MDB están suscritos a una cola o tópico
- Cuando llega un mensaje el contenedor despierta al MDB para que procese el mensaje

Estructura Enterprise Beans

Clase Enterprise Bean: implementa
– métodos de negocio
métodos de ciclo de vida (llamados directamente por el contenedor)

API vista-cliente: define
– Home interface
• métodos create, remove y find
– Remote interface
• Métodos de negocio

Deployment descriptor: Documento XML declara:
– Información de los EJB y su entorno
• Nombre del EJB
• Nombre de interfaces Home y Remote
• Nombre de la clase EJB
• Tipo del EJB
• Servicios que el EJB espera de su contenedor, como
transacciones
• Entradas del entorno EJB (dependencias con otros EJBs
y gestores de recursos)

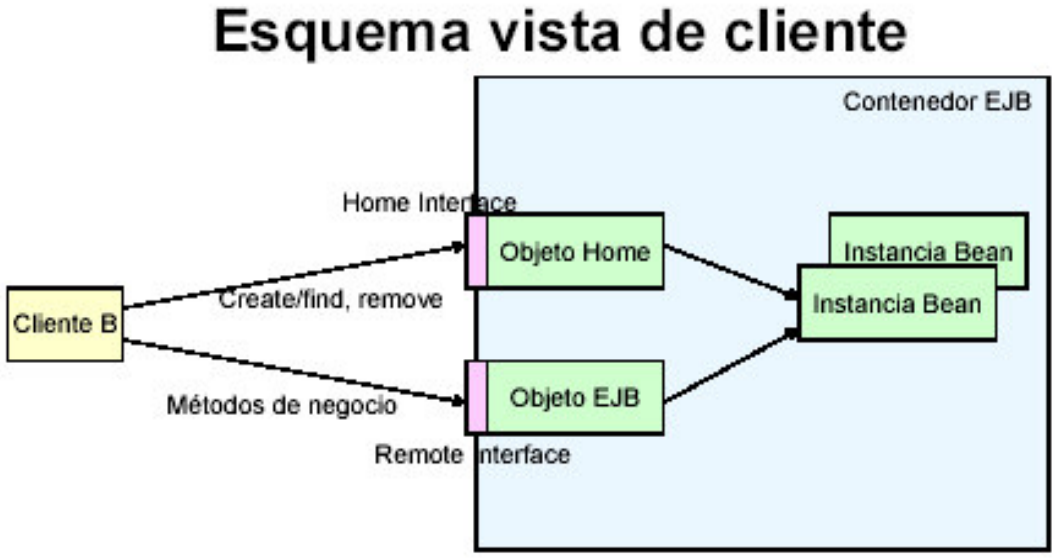
Vista cliente sobre Session y Entity Beans
Session y entity beans ofrecen distintas interfaces a los clientes
messege-driven beans no tienen clientes propiamente dichos sino que reaccionan a mensajes,
por lo que no ofrecen interfaces sino que anuncian su canal de mensajes (JMS-Topic o JMS-Queue)
con distinta forma de programación y comportamiento que session y entity beans

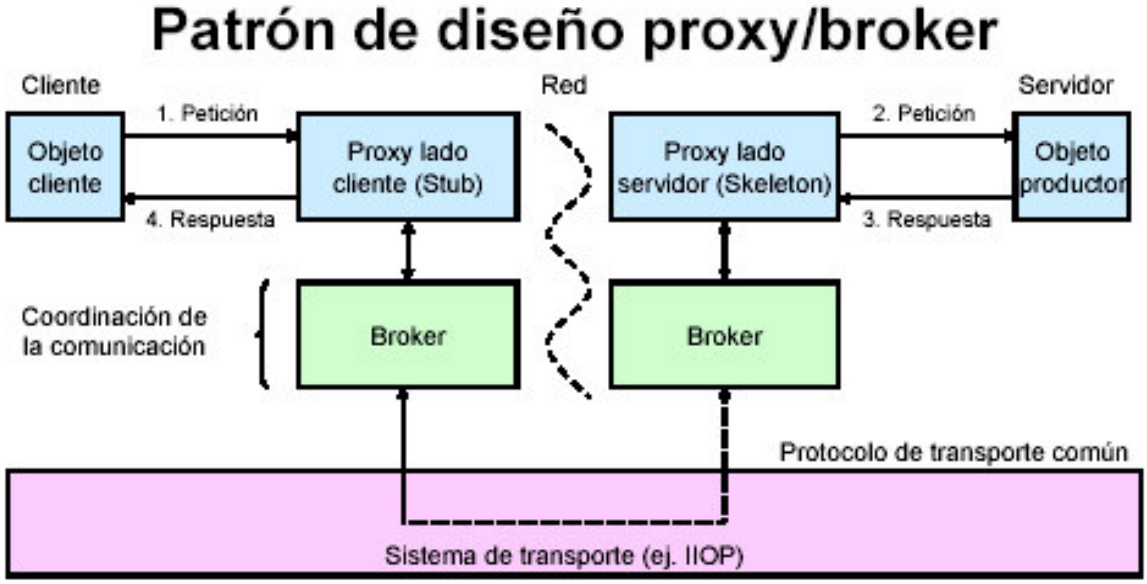
Importante para los clientes:
– Servicio de nombres y directorios a través de JNDI para localización de EJBs
– Protocolo RMI-IIOP para llamadas remotas a métodos

Características de la vista del cliente
Home Interface: Crear y borrar beans (session y entity beans), encontrar (entity beans)
(Remote) Component Interface: Exporta los métodos de negocio
Las interfaces son implementadas por Objetos-Home y Objetos-EJB:

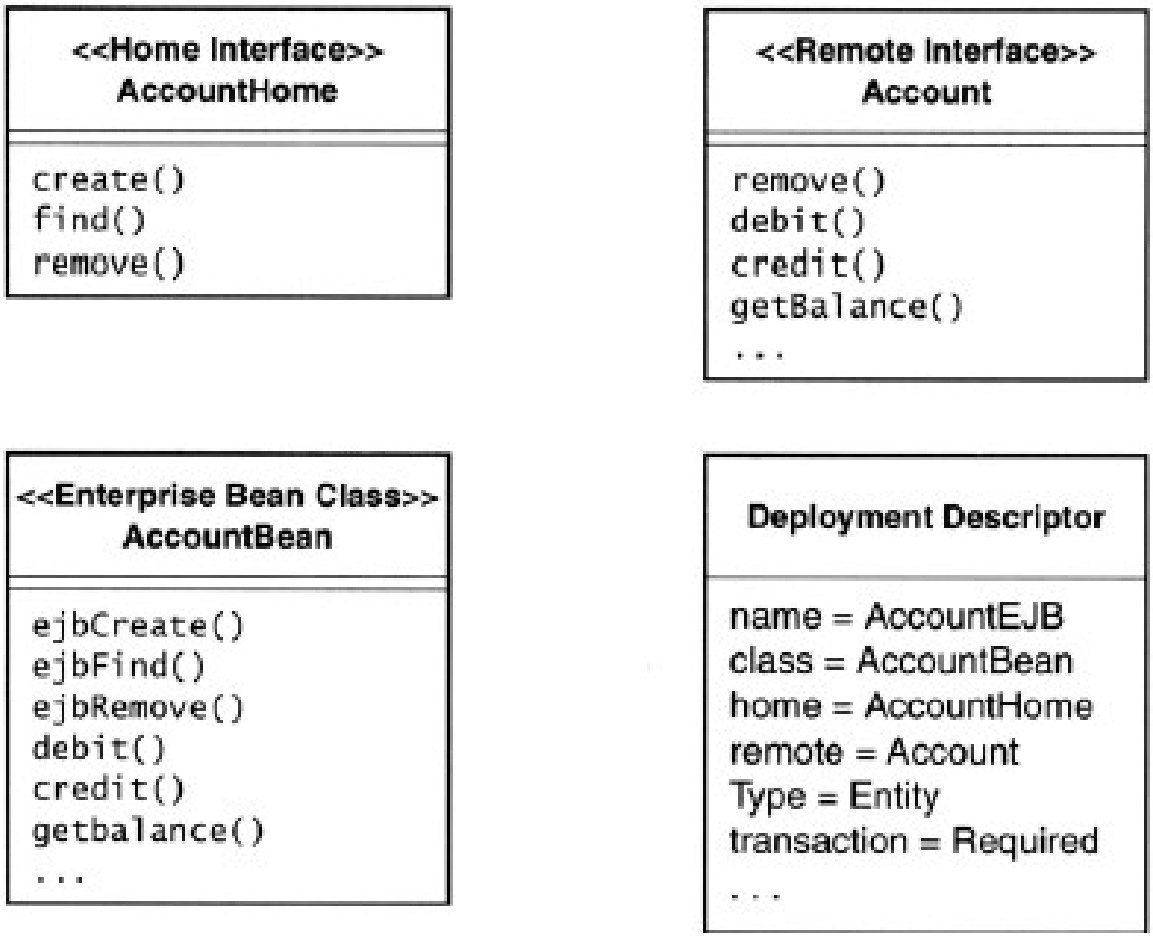
- Interfaces creados por el diseñador
- Durante la instalación, el contenedor genera las clases que implementan las interfaces
- Intercepción: el cliente nunca opera directamente sobre las instancias de beans sino sobre objetos generados por el contenedor
- El contenedor controla las llamadas y realiza pre y post-procesado
- Proporciona el entorno de ejecución a los EJBs llamados

Ej.: Control de permisos, enlace dinámico del objeto-bean a las instancias-bean, generación de instancias-bean si es necesario (Just-in-Time-Activation), o creación previa y gestión (Pooling)





Ejemplo: Account EJB



Contenedor EJB

El contenedor EJB y EJB server pueden ser considerados como el mismo. Este provee niveles de bajo nivel a servicios en tiempo de ejecución, EJB es accesando directamente por aplicaciones cliente y otras especificadas por el servidor. Los contenedores crean nuevos beans, dándole seguridad a lo que se ha creado manteniendo presente una interfaz entre bean y el servidor. Todas las peticiones que genera el bean desde esta infraestructura son directamente del contenedor. El contenedor administra por completo el ciclo de vida de un bean, el cual es creación, destrucción, passivation, distribución y otras infraestructuras relacionadas. A continuación se muestran algunos de los servicios que brinda un EJB:

Administración de instancias:
– Gestión del ciclo de vida de instancias
– Los estados y procesos del ciclo de vida dependen del tipo de Bean

Acceso remoto:
– El servidor EJB proporciona protocolos de comunicación para el acceso remoto a objetos distribuidos (RMI-IIOP)
– La interfaz y semántica de llamada deben seguir las convenciones de Java RMI
– El protocolo de comunicación debe soportar IIOP (según especificación CORBA)
– La gestión de la distribución de las llamadas remotas es tarea del contenedor

Seguridad:
– Autorización de acceso a componentes: concepto declarativo basado en roles: En Deployment Descriptor (descriptor de instalación) se definen roles de usuario y sus derechos de acceso a los métodos de los componentes
– El instalador (deployer) asigna roles a los usuarios, la gestión de usuarios y roles la realiza el servidor EJB, el control de acceso el contenedor

Persistencia:
– Las instancias de Entity Beans en la memoria de trabajo pueden estar enlazadas con datos de negocio de cualquier EIS. El contenedor garantiza la consistencia de datos (carga y almacenamiento periódicos)
• Persistencia gestionada por Beans: la instancia usa las conexiones JDBC a bases de datos proporcionadas por el contendor
• Persistencia gestionada por contenedor: en general se soportan bases de datos relacionales
• El medio de persistencia depende del contenedor y es independiente del Bean

Servicio de nombres y directorios:
– Asociación de referencias a objetos con nombres en una estructura de directorios jerárquica (JNDI API: Jave Naminig and Directory Interface)
– el contenedor asocia los Beans de forma automática, y proporciona además diversa información a los Beans en un directorio privado (Entorno)

Mensajería:
– El contendor proporciona a los beans acceso al servicio de mensajería mediante el API JMS (Java Messaging Service): comunicación asíncrona de mensajes entre dos o más participante mediante un sistema de colas de mensajes
– Los receptores de mensajes son los "Message -Driven Beans", cualquier Enterprise Bean puede ser emisor

Herramientas de contenedor
Las herramientas del contenedor leen el deployment descriptor y generan clases adicionales llamadas artefactos de contenedor (container artifacts), los artefactos de contenedor permiten al contenedor inyectar los servicios de nivel de sistema (intercepción)

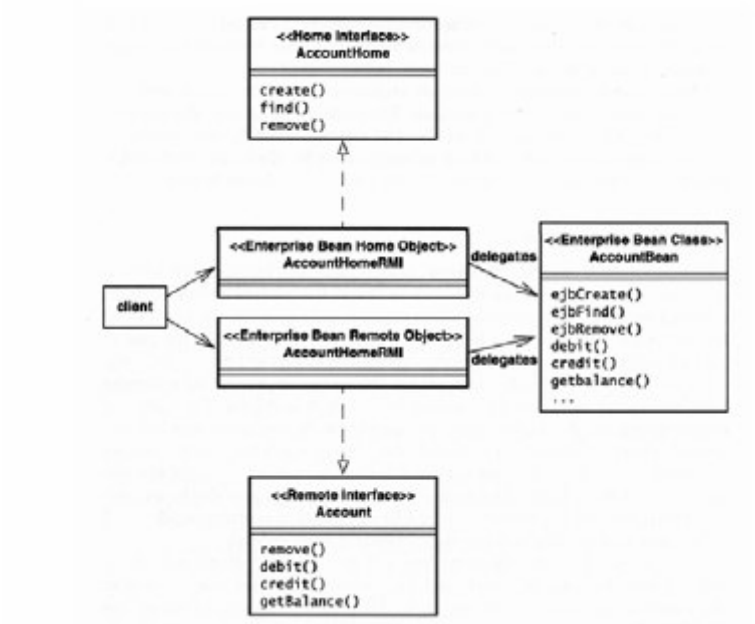
Elementos de una aplicación operativa:

– EJBs (lógica de negocio)

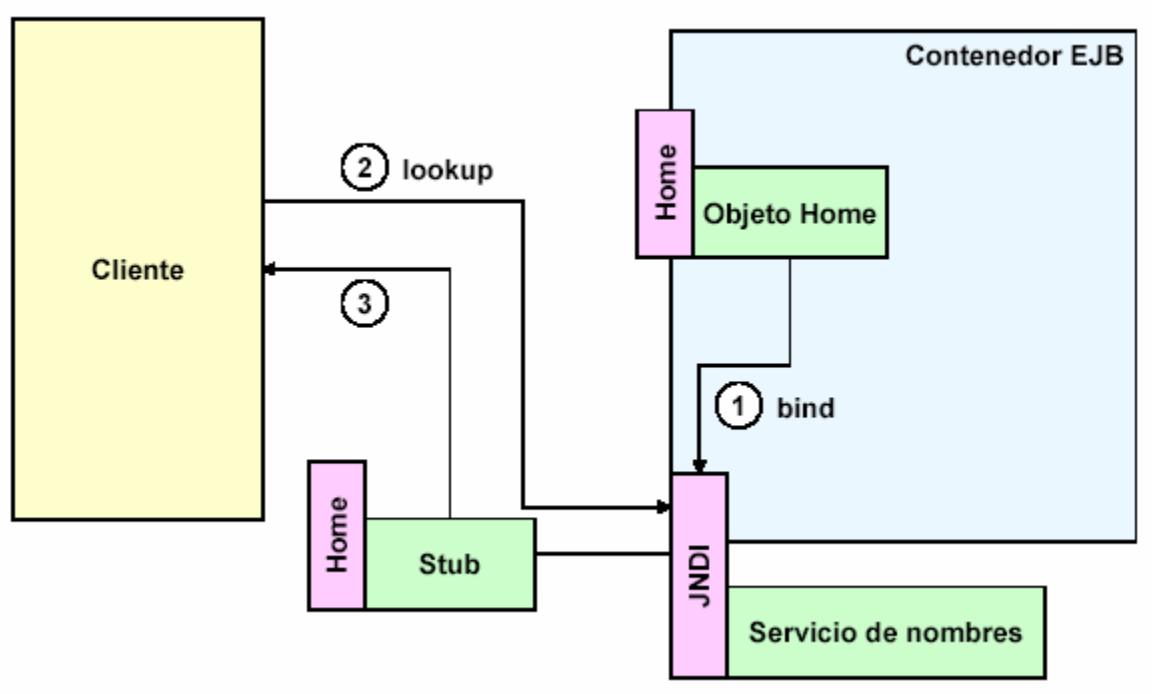
– Contenedor (implementación de los servicios de nivel de sistema)

– Artefactos de contenedor:

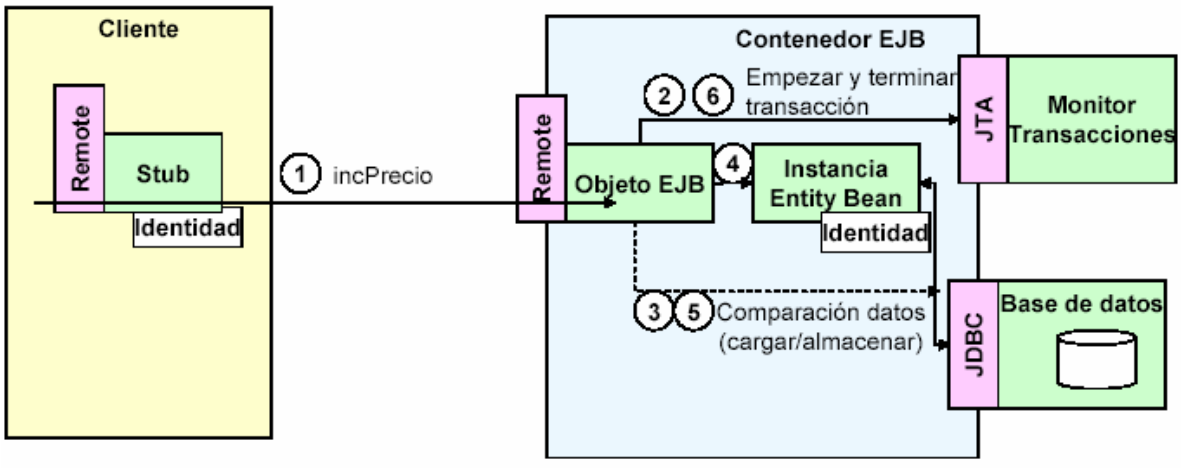
Artefactos del contenedor:



Llamada de un cliente: servicio de nombres:



Llamada a un método de negocio:



CONCLUSIONES

Se han establecido las bases imprescindibles para aumentar el conocimiento sobre las cambiantes tecnologías del mundo moderno en el área de la informática, en este caso el de los web services XML motivando así el comienzo de la integración web tanto con aplicaciones que tengan las compañías como para efectuar el comercio electrónico entre ellas, una eficacia mejorada, un mayor ahorro en gastos en el desarrollo de aplicaciones por medio de una mayor rapidez en dicho desarrollo y relaciones mas fuertes entre los usuarios y los proveedores. Diversos tipos de aparatos, desde ordenadores personales y distintos servidores hasta varios tipos de aparatos inteligentes, colaboraran a la perfección. Estimulando a las empresas salvadoreñas a una serie de nuevas oportunidades empresariales y tecnologías, llevando a Internet a un nuevo nivel.

Las formas de aplicar el contenido del estudio serán inimaginables, un ejemplo de estas es el implantar un sistema basado en web services para integrar la inscripción en línea de asignaturas que hace un estudiante en el sistema de registro académico de la Universidad Don Bosco que se encuentra desarrollado en Visual Fox con el pago que se hace en el sistema de un banco el cual puede estar desarrollado en Power Builder, Oracle, Java o Visual Basic entre otros.

Internet basado en los servicios será un lugar en el que la comunicación entre equipos sea común, así como el desarrollo de aplicaciones basadas en los web services en Internet, surgirán nuevas oportunidades para una colaboración más rica entre las personas y empresas salvadoreñas.

BIBLIOGRAFÍA

Vínculos de la Web

Información consultada el día miércoles 26 de Noviembre del año 2003 en el horario de 10:30 a.m. a 2:47 p.m.

- Índice técnico de los informes de W3C - http://www.w3.org/TR/
- Arquitectura de los Web services - http://www.w3.org/TR/wsa-reqs
- Implementación de SOAP4J - http://www.alphaworks.ibm.com/tech/soap4j/
- Seguridad de los Web services conceptos, estándares y requisitos - http://wwws.sun.com/software/whitepapers/webservices/securing_webservices.pdf

Información consultada el día lunes 29 de Diciembre del año 2003 en el horario de 08:00 p.m. a 10:33 p.m.

- Web Services XML-RPC, SOAP, sobre PHP, Perl, y otros conceptos - http://web-services.bankhacker.com/
- Web Services - http://homepages.mty.itesm.mx/al450951/

Información consultada el día jueves 15 de enero del año 2004 en el horario de 02:47 p.m. a 07:04 p.m:

- Análisis Web Services - http://www.activalink.net/index.php/An%E1lisis%20Web%20Services
- Implementaciones en SOAP - http://www.soapware.org/directory/4/implementations
- Aspectos básicos de XML Web Services - http://www.microsoft.com/spanish/msdn/articulos/archivo/280202/voices/webservbasics.asp

Información consultada el día viernes 30 de enero del año 2004 en el horario de 07:17 a.m. a 11:10 a.m:

- Arquitectura de Web Services - www.optimisa.cl/PDF/AlfredoPiquer.pdf
- Web Services Evolución histórica del Web - www.fing.edu.uy/inco/cursos/tsi2/TeoricoClase2.pdf

GLOSARIO

AB	abbr. Equipo Asesor (Advisory Board)
AC	abbr.Comité Asesor (Advisory Committee)
Actividad	n. Cualquiera de las áreas de trabajo formal del W3C, de acuerdo a la definición en el Documento de Procesos.
Equipo Asesor	n. Grupo de 9 personas elegidas por el Comité Asesor con responsabilidades varias.
Comité Asesor	n. Grupo constituido por los representantes oficiales de cada uno de los Miembros de la Organización W3C.
Bot	n. Agente de software encontrado en canales IRC. e.g. Zakim
Bridge	n. Puente para teleconferencias. Se trata de un dispositivo que conecta múltiples líneas de teléfono con el fin de mantener reuniones.
Chair	n. (1) Persona designada por el Director y que dirige un Grupo de Trabajo. (2) Líder o moderador de cualquier reunión del W3C.
Chairman	n. Término obsoleto actualmente reemplazado por "Chair".
Comm	n. Comunicaciones, entendiéndose hacia el Público y los Miembros. Primariamente es la responsabilidad del Equipo de Comunicaciones (Comm Team), grupo del Equipo W3C.
COO	n. "Chief Operating Officer". Nuevo puesto ejecutivo en el W3C, responsable de operaciones y de la ejecución del Proceso W3C (W3C Process).
Date Space (Espacio para Fecha)	Área del sitio W3C (W3C Web Site) para el cual la parte alta de jerarquía tras www.w3.org es un número de año.
Director	n. (1) Persona designada por el Acuerdo de Miembro que tiene la autoridad ejecutiva en el W3C. Tradicionalmente el Director se ocupa principalmente de las cuestiones técnicas.
Interino (Fellow)	n. Cualquier miembro del Equipo W3C que sea empleado de una organización Miembro del W3C , en calidad de préstamo al W3.
FTF	abbr: Cara a Cara: Reunión en la que la mayoría de los participantes acuden físicamente a la misma sala.
Host (Anfitrión)	n. Cualquiera de las organizaciones patrocinadoras del W3C, actualmente MIT, ERCIM y Keio. Los miembros del Equipo W3C que no son Interinos son empleados o contratados por los Anfitriones.
IRC	Internet Relay Chat: Un protocolo [RFC 1459] para la comunicación por texto en tiempo real por Internet usado habitualmente por el W3C para comunicaciones internas y para proseguir reuniones.

Longfellow	(1) Henry Wadsworth Longfellow (1807-1882), poeta americano, maestro y lingüista a cuya memoria se dio nombre a un puente de carretera que une Cambridge con Bostonun. (2) Puente de teleconferencias del W3C con capacidad para 24 líneas.
Member (Miembro)	n. (1) Organización (ocasionalmente individuo) que tiene un contrato mediante el Acuerdo de Miembro (Member Agreement) para participar en las actividades del W3C. (2) Persona empleada por o designada para participar en las actividades del W3C por un Miembro (en el sentido 1 de la definición).
Mystic	(1) Río que separa Chelsea y Charlestown, Massachusetts (2) Nombre original del Puente de carreteras Tobin. (3) Puente de teleconferencias con capacidad para 6 líneas.
Note (Nota)	n. Publicación en el sitio web del W3C, dentro del área de Recomendaciones Técnicas "Technical Recommendations", que no constituye una Recomendación Técnica, pero que por razones de trabajo merece ser publicada.
Scribe (Escribano)	n. Persona designada para archivar y publicar las actas de una reunión
SysWeb	n. Grupo del Equipo W3C responsable de las operaciones técnicas de las infraestructuras de ordenadores del W3C y página web. El contenido público de la página web es gestionado por el Equipo de Comunicaciones (Comm Team).
TAG	abbr.Grupo de Arquitectura Técnica (Technical Architecture Group)
Team (Equipo)	n. Conjunto de personas consistente en empleados del W3C, contratantes e Interinos.
Grupo de Arquitectura Técnica	n. Grupo de 9 personas que trabajan en aspectos varios de la Arquitectura de Red.
Informe Técnico	n. Documento publicado de un modo Oficial por el W3C. Incluye específicamente documentos de seguimiento de la Recomendación y Notas. En el pasado las Entregas se consideraban Informes Técnicos.
Tobin	(1)Maurice J. Tobin (1901-1953), Político de Boston, a cuya Memoria se dedicó un Puente de carretera que une Chelsea con Charlestown (MA). (2)Puente de teleconferencias del W3C con capacidad para 18 líneas.

W3C	n. Consorcio World Wide Web: Entidad no corporativa creada por contratos entre los Anfitriones con el propósito de llevar la Red a su potencial máximo.
Zakim	(1) Leonard P. Zakim (1953-1999), defensor de los derechos civiles de Boston, en cuyo honor se dio nombre a un Nuevo Puente de carreteras que une Boston con Charlestown. (2) Puente de teleconferencias del W3C con capacidad para 120 líneas. (3) Bot de IRC que ayuda en la gestión de una reunión.

Anexos

PRESUPUESTO

CANTIDAD	DESCRIPCIÓN	PRECIO UNITARIO	TOTAL
1	Ordenador AMD DUROM 1.8 Ghz., 128 Mb RAM, 40 Gb. de disco duro, Unidad lectora de CD. y CD RW	$600.00	$600.00
300	Horas de Internet para investigación	$1.00	$300.00
1	Gastos de papelería (impresiones, fotocopias, anillados, empastados, CD, libros, papel, etc.).	$150.00	$150.00
1	Transporte	$100.00	$100.00
300	Horas hombre para investigación*	$6.40	$1920.00
1	Gastos indirectos	$100.00	$100.00
Costo total del proyecto			**$3,170.00**

Salario que obtiene el profesional UDB contratado por horas.

CRONOGRAMA DE ACTIVIDADES

ACTIVIDAD	ENERO				FEBRERO				MARZO				ABRIL				MAYO				JUNIO			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Recopilación de información sobre arquitectura y plataformas de web services.	■	■																						
Clasificación, selección y análisis de la información.			■	■																				
Desarrollo analítico de web services, pruebas de plataforma instalación y prueba de servidores					■	■	■	■																
Primera defensa									■	■														
Recopilación de información sobre principales protocolos de comunicaciones, lenguajes de programación y marcado de web services.											■	■	■											
Clasificación, selección y análisis de la información														■	■	■								
Realizar pruebas demostrativas sobre funcionamiento y configuración de protocolos de comunicaciones.																	■	■	■	■	■	■		
Segunda defensa.																							■	■