ESTUDIO Y ELABORACIÓN DE UNA GUÍA METODOLÓGICA PARA IMPLEMENTAR LOS SISTEMAS DISTRIBUIDOS MEDIANTE DE WEB SERVICES Y NET REMOTING

EDUARDO JOSÉ GONZÁLEZ ALCÁZAR

CÓDIGO: 200105038

DIRECTOR

MOISÉS QUINTANA

MSC. CIENCIAS COMPUTACIONALES

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

PROGRAMA DE INGENIERÍA DE SISTEMAS

FACULTAD DE INGENIERÍA

CARTAGENA

2007

ESTUDIO Y ELABORACIÓN DE UNA GUÍA METODOLÓGICA PARA IMPLEMENTAR LOS SISTEMAS DISTRIBUIDOS MEDIANTE DE WEB SERVICES Y NET REMOTING

Presentado por:

EDUARDO JOSÉ GONZÁLEZ ALCÁZAR

Estudiante de Ingeniería de Sistemas

Universidad Tecnológica de Bolívar

Revisado por:

MOISÉS QUINTANA

Director de Monografía

Tabla de Contenido

Tabla de Ilustraciones

Objetivos

Ilustrar en qué consiste, las principales características y la mejor manera de implementar las principales tecnologías de invocación de objetos remotos, tales como los Web Service y .Net Remoting para el caso del .Net Framework.

Objetivos Específicos

- Explicar las características principales de las diferentes plataformas sobre la cual funcionan estas tecnologías de invocación de objetos remotos.
- Definir los Servicios Web, enumerando sus principales características, y arquitectura.
- Identificar los principales problemas de seguridad que plantean los Servicios Web y las mejores técnicas para corregirlos.
- Explicar la implementación de un Servicio Web en un Servidor de Aplicaciones, en este caso Internet Information Service.
- Definir .Net Remoting, enumerando sus principales características, y arquitectura.
- Conocer las principales similitudes y diferencias entre .Net Remoting y otras tecnologías de invocación de objetos remotos.

Introducción

Los Servicios Web son una prometiente solución a viejas necesidades que existían en el mundo de la informática; estos presentan características que se han venido buscando y trabajando desde hace tiempo, como son: la reutilización de código y la necesidad de distribuir las aplicaciones. También presentan ciertas características importantes como son: compartir información fácil y flexiblemente entre los usuarios y los negocios. Estos habilitan el acceso a datos que previamente habían sido escondidos dentro de las compañías y accesible solo usando un software personalizado. Junto con los beneficios vienen serios riesgos: información importante podría ser expuesta a gente que no debería verla. Pero nunca obtendrán su mayor potencial si no aprendemos a manejar estas situaciones

Estos representan la siguiente fase de la computación distribuida. La extensión de la computación distribuida comenzó con los protocolos conocidos como TCP/IP. Usar TCP/IP para construir aplicaciones distribuidas era mucho trabajo para los diferentes programadores, que solo querían construir aplicaciones de negocio. Para aminorar la carga de trabajo de los programadores distribuidos la industria de la computación desarrolló el Entorno de Desarrollo Distribuido conocido como (DCE), basado en los paradigmas de cliente/servidor, seguido por CORBA. Casi al mismo tiempo, Microsoft introdujo el COM, seguido por DCOM usando tecnología DCE como una base, y COM+. Sun, construyendo en su lenguaje Java, introdujo la plataforma Java 2, Edición Empresarial (J2EE), con su popular Enterprise Java Beans (EJBs), usando muchos conceptos e ideas de las tecnologías previas. Cada paso hizo de la computación distribuida mucho más fácil,

pero generalmente cada tecnología todavía vive en su propio mundo, haciendo la interoperabilidad entre las capas medias de las diferentes tecnologías muy difíciles.

Ahora los Servicios Web han salido a escena. Las dos metas son hacer la programación distribuida mucho más sencilla y realzar la interoperabilidad. Estas metas son ayudadas por:

- Poco acoplamiento entre el programa que realiza la petición y el servicio proveedor.
- El uso de XML, cuya plataforma y lenguaje son neutrales.

Cuando todos los anteriores modelos distribuidos fueron implementados se considero en primer plano la tecnología, y la seguridad siempre fue dejada de ultimo. El mantra era: “Hagamos funcionar el modelo, después nos preocupamos por la seguridad”. Inevitablemente, esto resultó en un pobre desempeño y un difícil uso de la seguridad.

El reto de la seguridad de los Servicios Web radica en entender y determinar los riesgos involucrados en la seguridad de un servicio basado en web. Todo esto teniendo en cuenta la tecnología de seguridad existente y al mismo tiempo emergiendo los estándares, y entender como ellas serán usadas para compensar el riesgo en los Servicios Web. Cualquier modelo de seguridad debe ilustrar cómo los datos deben fluir a través una aplicación y la topología de la red para conocer los requerimientos definidos por los negocios, sin exponer los datos a un riesgo indebido.

Microsoft, además de las mencionadas anteriormente, ha desarrollada la tecnología distribuida .NET Remoting, la cual es la sucesora de DCOM. Dicha tecnología nos permite invocar Métodos y pasar objetos sin importar en que dominio se encuentren las aplicaciones que intervienen.

Esta Monografía está compuesta de 6 capítulos, en los cuales se explicarán las plataformas que soportan a estas tecnologías distribuidas sobre las que hemos estado hablando. También explicaremos en qué consiste los Servicios Web, su funcionalidad, ventajas y desventajas, las mejores formas de hacerlos seguros de tal manera que los datos no sean expuestos. Mostraremos como implementar dichos servicios con las tecnologías más importantes y las diferencias que hay en cada una de ellas. Después pasaremos a hablar de .NET Remoting y de sus diferencias con los Servicios Web y la desarrollada por Sun llamada RMI.

CAPITULO I

Plataformas y Tecnologías Necesarias para soportar los Servicios Web y .NET Remoting

En este capítulo veremos las distintas plataformas y tecnologías necesarias para desarrollar sistemas que implementen Servicios Web y .NET Remoting y qué papel desempeñan.

.NET Framework

.NET Framework es la plataforma implementada por Microsoft para desarrollo sobre Sistemas Operativos Windows. Esta opera como una máquina virtual que procesa todo el código generado por los compiladores como una capa intermedia, para después sí llevarlo a un código que pueda entender el Sistema Operativo. Esto hace el código mucho más seguro y libre de errores indeseables que puedan dañar a dicho Sistema.

Esta plataforma provee una serie de clases o Api desarrolladas a partir de diferentes patrones de diseños, y por medio de ellas los desarrolladores pueden implementar los diferentes software. Este Framework de Microsoft presenta varios lenguajes de programación como son: C# - Su lenguaje bandera -, J# que es la versión de Microsoft para .NET de la plataforma de desarrollo de Sun Microsystems “Java”; también nos trae otros lenguajes que nos había presentado en el pasado como Visual Basic .NET, el cual ahora se

presenta totalmente orientado a objeto, y también trae el Visual C++ que es la versión del popular C++ para .NET. Esta variedad de lenguajes permite al desarrollador escoger el que más le guste y al que más se acomode, y a diferencia de lo que muchos pueden pensar no presenta ninguna dificultad o ningún problema para utilizar las diferentes API que pertenece al Framework.

[Microsoft .NET es una plataforma de desarrollo y ejecución de aplicaciones. Esto quiere decir que no sólo nos brinda todas las herramientas y servicios que se necesitan para desarrollar modernas aplicaciones empresariales y de misión crítica, sino que también nos provee de mecanismos robustos, seguros y eficientes para asegurar que la ejecución de las mismas sea óptima. Los componentes principales de la plataforma .NET son:

- Un entorno de ejecución de aplicaciones, también llamado “Runtime”, que es un componente de software cuya función es la de ejecutar las aplicaciones .NET e interactuar con el sistema operativo ofreciendo sus servicios y recursos.
- Un conjunto de bibliotecas de funcionalidades y controles reutilizables, con una enorme cantidad de componentes ya programados listos para ser consumidos por otras aplicaciones.
- Un conjunto de lenguajes de programación de alto nivel, junto con sus compiladores y linkers que permitirán el desarrollo de aplicaciones sobre la plataforma .NET.
- Un conjunto de utilitarios y herramientas de desarrollo para simplificar las tareas más comunes del proceso de desarrollo de aplicaciones

- Documentación y guías de arquitectura, que describen las mejores prácticas de diseño, organización, desarrollo, prueba e instalación de aplicaciones .NET][1]

La arquitectura del Framework de .NET es muy sencilla y se maneja en forma de pirámide. En el primer nivel – arriba hacia abajo – nos encontramos con les lenguajes de alto nivel (J#, C#, VB .NET, C++), seguidos por el CTS que es la es la especificación de lenguajes común "El sistema común de tipos, conocido como CTS, provee una definición común de los tipos de datos básicos que utiliza el CLR. El CTS posibilita, entre otras cosas, que todos los lenguajes de alto nivel que compilan contra una plataforma CLI compartan el mismo sistema de tipos de datos, permitiendo lograr una mejor interoperabilidad"[2]. En tercera línea nos encontramos con las librerías que nos trae .NET, entre las cuales encontramos las clases para los Win Forms, ASP.NET, ADO.NET, y las clases base y de utilidades que este Framework nos trae. En la parte de abajo vemos que esta el CLR o Common Language Runtime (Lenguaje Común en Tiempo de Ejecución). Este es el que se encarga de llevar las clases de la plataforma a un código que pueda entender el Sistema Operativo que en este caso es Windows. Lo anterior es lo que comúnmente se llama "Maquina Virtual".

[1] (MSDN, 2006), Programa Microsoft Desarrollador Cinco Estrellas
[2] (MSDN, 2006), Programa Microsoft Desarrollador Cinco Estrellas

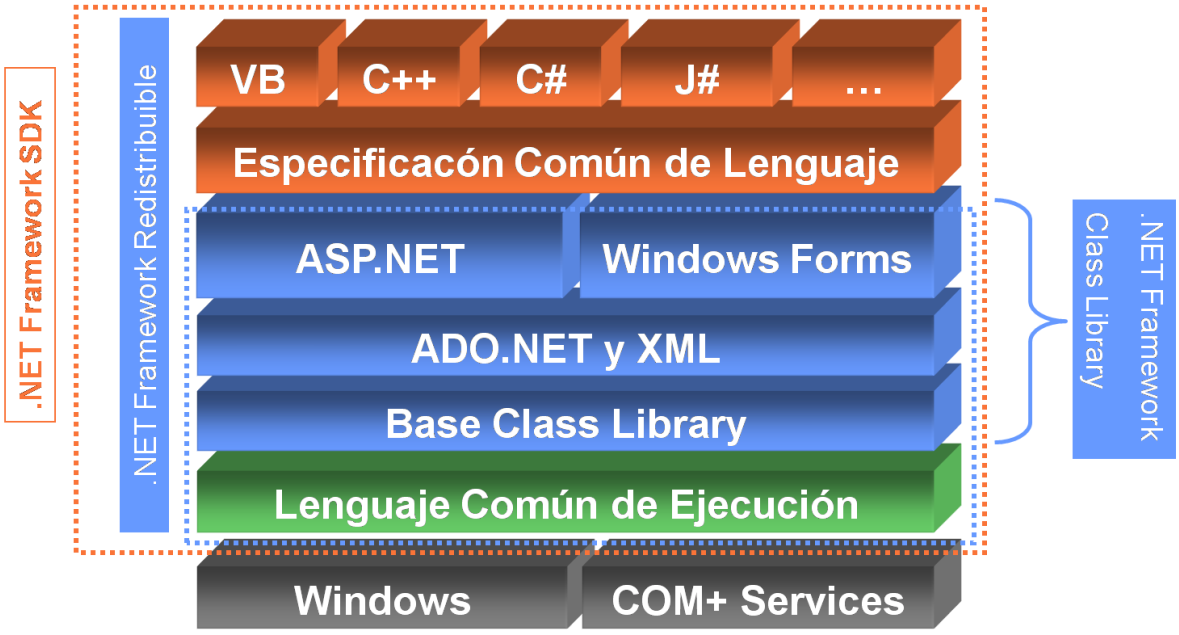


Ilustración 1: Arquitectura del .NET Framework (MSDN, 2006)

HTTP

El protocolo de transferencia de hipertexto (Hypertext Transfer Protocol) se encarga de procesar como su nombre lo indica hipertexto, el cual consisten en texto con componentes y enlaces a otros textos. Pero en la actualidad no es usado solamente para eso, también se usa para transferencia de archivos, diferentes tipos de datos, trafico de red, etc.

Este protocolo tiene un funcionamiento básico. Cada vez que el cliente realiza una petición se ejecutan los siguientes pasos:

- El cliente Web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.
- Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente.

  Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD,…), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada (casi siempre HTTP/1.0) y un conjunto variable de información, que incluye datos sobre las capacidades del navegador, datos opcionales para el servidor,…
- El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.
- Se cierra la conexión TCP.

Este proceso se repite en cada acceso al servidor HTTP. Por ejemplo, si se recoge un documento HTML en cuyo interior están insertadas cuatro imágenes, el proceso anterior se repite cinco veces, una para el documento HTML y cuatro para las imágenes.

HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores. Al finalizar la transacción todos los datos se pierden. Por esto se popularizaron las cookies, que son pequeños ficheros guardados en el propio ordenador que puede leer un sitio web al establecer conexión con él, y de esta forma reconocer a un visitante que ya estuvo en ese sitio anteriormente. Gracias a esta identificación, el sitio

web puede almacenar gran número de información sobre cada visitante, ofreciéndole así un mejor servicio.[3]

Hay páginas que prestan el servicio de realizar una petición a una página web en específico y muestran el encabezado que se trasmite a través del protocolo. Para realizar un ejemplo utilizaremos la página http://www.programacionweb.net/utilidades/headers.php y realizaremos una petición a la página de http://www.unitecnologica.edu.co y esto es lo que nos muestra la siguiente información:

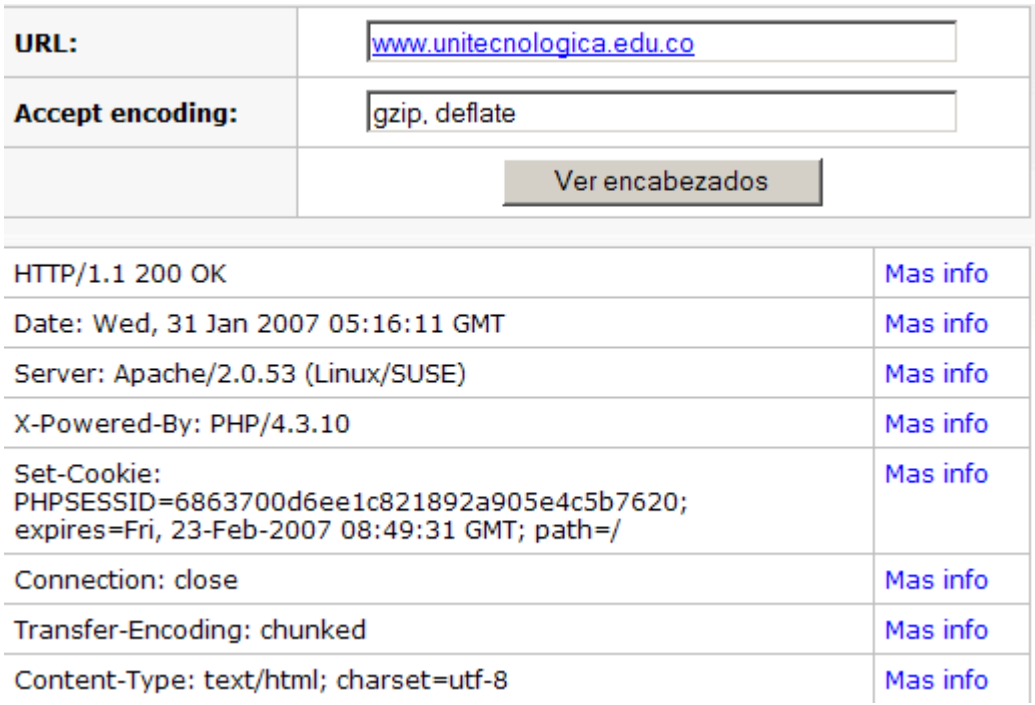


Ilustración 2: Encabezado Protocolo HTTP (http://www.programacionweb.net/, 2003 - 2007)

[3] (Wikipedia, 2007), http://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol

XML

SGML fue tomado como un estándar internacional en 1986. Herramientas y procesos para trabajar con SGML, pero esta no fue tan ampliamente adoptada como se esperaba. A mediados de 1990, la capacidad de SGML para hacer conjuntos de elementos de etiquetas fue usada para crear HTML, el cual proveo con la capacidad de crear etiquetas y mostrarlo usando la tecnología de los navegadores. Pero HTML solo podía marcar datos como encabezados o listas de párrafos o cualquier otro simple formato orientado a contenido. Él no podía semánticamente describir en qué consistía el documento propiamente dicho. Por ende se necesitaba algo mucho más poderoso

El lenguaje HTML, a pesar de su sencillez, es sin duda un invento prodigioso. Es el más exitoso sistema de presentación de documentos de la historia. Desde que apareció el WWW, gracias al HTML hemos podido publicar y acceder a más información de la que jamás hemos podido imaginar.

Pero a su vez, el HTML ha sido víctima de su propio éxito. El gran crecimiento de Internet, los intereses comerciales y la necesidad de poder realizar páginas Web vistosas, ha dado lugar a que en poco tiempo este lenguaje haya evolucionado muy rápidamente y, por desgracia, no siempre por el camino más adecuado. Actualmente estamos en la versión 4.0 y, sin embargo, sigue siendo igual de rígido e inflexible como era en un principio. Y es que es un lenguaje limitado en cuanto que no nos permite realizar sobre Internet todas las aplicaciones o cosas que nos gustaría.

Estas razones han obligado a los miembros del W3 Consortium a, en lugar de desarrollar nuevas versiones de HTML desarrollar un nuevo lenguaje (mejor dicho metalenguaje) que han denominado XML (Extensible Markup Language) que aproveche las innegables ventajas del HTML pero que a su vez permita realizar muchas cosas más. Esto no significa, al menos por el momento, el fin del HTML. Existen demasiadas páginas en HTML y resulta muy sencillo crearlas. Además los navegadores no soportarán todavía en toda su potencia el XML y tecnologías asociadas, pero es evidente una reformulación del HTML como una aplicación XML y un cambio radical en la forma de elaborar las páginas WEB en los próximos años.

La idea que subyace bajo el XML es la de crear un lenguaje muy general que sirva para muchas cosas. El HTML está diseñado para presentar información directamente a los humanos, y esto sin duda es algo bueno, pero es un lenguaje complicado de procesar para los programas informáticos. El HTML no es bueno porque no indica lo que está representando, se preocupa principalmente de que eso tiene que ir en azul, o con un tipo de letra determinada, pero no te dice que lo que está mostrando es el título de un libro o el precio de un artículo. El XML hace precisamente esto: describe el contenido de lo que etiqueta.

“XML (Extensible Markup Language) no es, como su nombre pudiera sugerir, un lenguaje marcado. XML es un meta-lenguaje que nos permite definir lenguajes de marcado adecuado a usos determinados. “XML es un lenguaje que permite realizar las etiquetas

que nos son necesarias. Estas etiquetas deben estar organizadas de acuerdo a ciertos principios generales"[4].

XML fue diseñado con objetivos muchos más pequeños que SGML, para que de esa manera procesadores de poco poder pudieran utilizarlos para servir su contenido a través del internet. El W3C en su especificación "Recomendaciones XML 1.0"describió las metas iníciales del XML. Estas metas de diseñas están publicada en la página de dicho organismo, y son:

- ~~XML~~ debía ser directamente usable en el Internet.
- ~~XML~~ debía soportar una amplia variedad de aplicaciones
- ~~XML~~ debía ser compatible con SGML
- Debía ser fácil escribir programas que procesen documentos XML.
- El número de características opcionales en XML debe ser guardado al mínimo absoluto, idealmente cero.
- Los Documentos XML debían ser Legibles y razonablemente claro.
- El diseño de XML debía ser preparado rápidamente.
- El diseño de XML debía ser formal y conciso
- Los Documentos XML debían ser fácil de crear
- La Brevedad en la etiquetas XML es de una mínima importancia.

XML ha transcendido mucho más allá, al ser un estándar y no "casarse" con ningún sistema operativo o lenguaje de programación, y dado lo flexible que es por el hecho de ser un meta-lenguaje, está siendo utilizado por todo el mundo, inclusive para la estandarización del HTML (Código en que se escriben la páginas web) el cual se llama

[4] (Rusty Harold, 2004), XML 1.1 Bible

XHTML y se encuentra en la versión 1.0, XML se ha convertido en el pan de cada día, se usa para guardar datos con un formato especifico, pasar información a través de la web, configuración de sistemas, etc.

SOAP

SOAP es un protocolo elaborado para facilitar la llamada remota de funciones a través de Internet, permitiendo que dos programas se comuniquen de una manera muy similar técnicamente a la invocación de páginas Web. Las solicitudes SOAP se pueden hacer en tres estándares: GET, POST y SOAP. Los estándares GET y POST son idénticos a las solicitudes hechas por navegadores de Internet. SOAP es un estándar similar a POST, pero las solicitudes se hacen en XML y permiten recursos más sofisticados, como pasar estructuras y arreglos ("arrays").

SOAP provee dos formas para representar invocaciones de operaciones de Servicios Web: RPC llamada de procedimiento remoto (Remote Procedure Call) mensajería y mensajería de estilo de documento. RPC mensajería provee una forma de representar métodos de invocación en mensajes SOAP. Sin embargo la estructura de los mensajes que representan operaciones de invocación es altamente rígida. Mensaje de Estilo de Documento, por otra parte proveen una mayor flexibilidad; este permite mensajes contener arbitrariamente elementos de datos. Aunque validar estos mensajes es mucho más complicado.[5]

[5] (Apple Computer, Inc., 2002, 2005)

Independientemente de cómo se haga la solicitud, las respuestas siempre son en XML. XML describe perfectamente los datos en tiempo de ejecución y evita los problemas ocasionados por cambios inadvertidos en las funciones, ya que los objetos llamados tienen la posibilidad de validar siempre los argumentos de las funciones, haciendo que el protocolo sea muy sólido.

Este protocolo se puede implementar fácilmente en casi cualquier ambiente de programación. Actualmente, existen diversos paquetes de desarrollo SOAP para diversos sistemas operativos y lenguajes de alto nivel. Microsoft tiene un paquete para Visual Studio 6 en http://msdn.microsoft.com/soap/default.asp. Este es una parte importante de la arquitectura .NET de Microsoft y cuenta con amplio soporte en Visual Studio.NET.

La estatura de un mensaje SOAP sería algo parecido a esto.

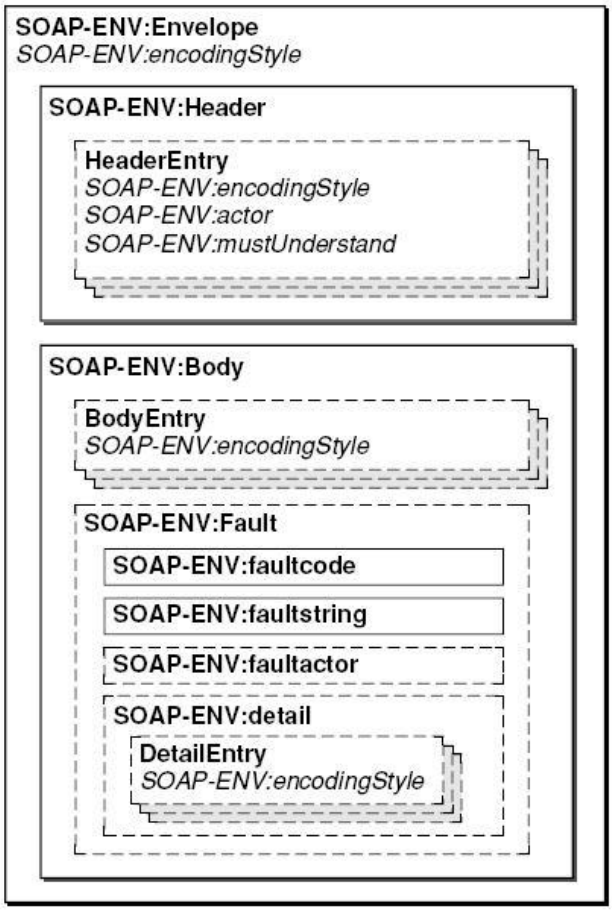


Ilustración 3: Estructura SOAP, (Apple Computer, Inc., 2002, 2005)

El protocolo SOAP tiene diversas ventajas sobre otras maneras de llamar funciones de manera remota como DCOM, CORBA o directamente en TCP/IP:

- Es sencillo de implementar, probar y usar.
- Es un estándar de la industria, creado por un consorcio del cual Microsoft forma parte, adoptado por W3C (http://www.w3.org/TR/SOAP/) y por varias otras empresas.

- Utiliza los mismos estándares de la Web para casi todo: la comunicación se hace mediante HTTP con paquetes virtualmente idénticos; los protocolos de autenticación y encriptación son los mismos; el mantenimiento de estado se hace de la misma forma; se implementa normalmente por el propio servidor Web.
- Atraviesa "firewalls" y routers, que "piensan" que es una comunicación HTTP.
- Tanto los datos como las funciones se describen en XML, lo que permite que el protocolo no sólo sea más fácil de utilizar sino que también sea muy sólido.
- Es independiente del sistema operativo y procesador.
- Se puede utilizar tanto de forma anónima como con autenticación (nombre/clave).

UDDI

La iniciativa UDDI surgió como respuesta a ciertas preguntas de varias empresas, incluidas Microsoft, IBM, Sun, Oracle, Compaq, Hewlett Packard, Intel, SAP y unas trescientas más, unieron sus esfuerzos para desarrollar una especificación basada en estándares abiertos y tecnologías no propietarias que permitiera resolver los retos anteriores. El resultado, cuya versión beta se lanzó en diciembre de 2000 y estaba en producción en mayo de 2001, fue un registro empresarial global alojado por varios nodos de operadores en el que los usuarios podían realizar búsquedas y publicaciones sin costo alguno.

A partir de la creación de esta infraestructura para servicios Web, los datos sobre estos servicios se pueden encontrar de forma sistemática y confiable en una capacidad universal totalmente independiente de proveedores. Se pueden llevar a cabo búsquedas categóricas precisas utilizando sistemas de identificación y taxonómicos extensibles. La

integración de UDDI en tiempo de ejecución se puede incorporar a las aplicaciones. Como resultado, se fomenta el desarrollo de un entorno de software de servicios Web.

La información de UDDI se aloja en nodos de operador, empresas que se han comprometido a ejecutar un nodo público conforme a la especificación que rige el consorcio UDDI.org. En la actualidad existen dos nodos públicos que se ajustan a la versión 1 de la especificación UDDI: Microsoft aloja uno e IBM el otro. Hewlett Packard se ha comprometido a alojar un nodo bajo la versión 2 de la especificación. Los operadores del host deben replicar datos entre ellos a través de un canal seguro, para conseguir la redundancia de la información en el registro UDDI. Se pueden publicar los datos en un nodo y descubrirlos en otro tras la réplica. Actualmente, la réplica se produce cada 24 horas. En el futuro, este intervalo entre réplicas se reducirá, ya que habrá más aplicaciones que dependan de los datos de UDDI.

Resulta importante observar que no existen requisitos de propietario respecto al modo en que el operador del host implementa su nodo. El nodo sólo se debe ajustar a la especificación UDDI. El nodo de Microsoft (http://uddi.microsoft.com/default.aspx [en inglés]), por ejemplo, se ha escrito por completo en C# y se ejecuta en producción en tiempo de ejecución en lenguaje común .NET. El código de base se beneficia claramente de la compatibilidad nativa con SOAP y de la serialización que ofrecen las clases de sistema .NET. En el lado del servidor, el nodo del operador Microsoft utiliza Microsoft® SQL Server 2000 como almacén de datos. Creo que basta con mencionar que IBM utiliza tecnologías diferentes para ejecutar su nodo, ¿verdad? No obstante, los dos nodos se comportan exactamente igual, ya que se ajustan al mismo conjunto de llamadas a API XML basadas en SOAP. Las herramientas de los clientes pueden interoperar con ambos

nodos sin problemas. Por eso, el nodo público UDDI constituye un claro ejemplo de que el modelo de servicios Web XML funciona en entornos heterogéneos.

El próximo paso para comprender la iniciativa UDDI consiste en ver qué datos se almacenan en UDDI y cómo se estructuran. UDDI es relativamente ligero; se ha diseñado como *registro*, no como *depósito*. La diferencia, aunque sutil, resulta esencial. Un registro redirige al usuario a recursos, mientras que un depósito sólo almacena información. El registro Microsoft® Windows® puede servir de ejemplo: contiene las configuraciones y parámetros básicos pero, en última instancia, su función es la de dirigir la aplicación a un recurso o binario. Buscar un componente COM basándonos en su Identificador de programa nos conducirá a un Identificador de clase, que a su vez nos dirigirá a la ubicación del binario.

UDDI se comporta de forma similar: como el registro de Windows, se basa en identificadores únicos globales (GUID) para garantizar la capacidad de búsquedas y determinar la ubicación de recursos. En última instancia, las consultas a UDDI conducen a una interfaz (un archivo .WSDL, .XSD, .DTD, etc.) o a una implementación (como un archivo .ASMX o .ASP) ubicadas en otro servidor.

WSDL

WSDL es un formato XML que describe los servicios de red como un conjunto de puntos finales que procesan mensajes contenedores de información orientada tanto a documentos como a procedimientos. Las operaciones y los mensajes se describen de

forma abstracta y después se enlazan a un protocolo de red y a un formato de mensaje concreto para definir un punto final de red. Los puntos finales concretos relacionados se combinan en puntos finales abstractos (servicios). WSDL es extensible, lo que permite la descripción de puntos finales de red y sus mensajes, independientemente de los formatos de los mensajes o protocolos de red utilizados para comunicarse. Sin embargo, los únicos enlaces que se muestran en este documento describen cómo utilizar WSDL junto con SOAP 1.1, HTTP GET/POST y MIME.[6]

WSDL se ha convertido en una pieza clave de la pila de protocolos de los servicios Web. Por eso, es importante saber cómo colaboran UDDI y WSDL y por qué la idea de interfaces frente implementaciones forma parte de cada protocolo. WSDL y UDDI se diseñaron para diferenciar claramente los metadatos abstractos y las implementaciones concretas.

WSDL distingue claramente los mensajes de los puertos: los mensajes (la sintaxis y semántica que necesita un servicio Web) son siempre abstractos, mientras que los puertos (las direcciones de red en las que se invoca al servicio Web) son siempre concretos. No es necesario que un archivo WSDL incluya información sobre el puerto. Un archivo WSDL puede contener simplemente información abstracta de interfaz, sin facilitar datos de implementación concretos, y ser válido. De este modo, los archivos WSDL se separan de las implementaciones.

[WSDL definen los **servicios** como colecciones de puntos finales de red o **puertos**. En WSDL, la definición abstracta de puntos finales y de mensajes se separa de la instalación

[6] (Christensen, y otros, 2000)

concreta de red o de los enlaces del formato de datos. Esto permite la reutilización de definiciones abstractas: **mensajes**, que son descripciones abstractas de los datos que se están intercambiando y **tipos de puertos**, que son colecciones abstractas de **operaciones**. Las especificaciones concretas del protocolo y del formato de datos para un tipo de puerto determinado constituyen un **enlace** reutilizable. Un puerto se define por la asociación de una dirección de red y un enlace reutilizable; una colección de puertos define un servicio. Por esta razón, un documento WSDL utiliza los siguientes elementos en la definición de servicios de red:

- **Types**: Contenedor de definiciones del tipo de datos que utiliza algún sistema de tipos (por ejemplo XSD).
- **Message**: definición abstracta y escrita de los datos que se están comunicando.
- **Operation**: descripción abstracta de una acción admitida por el servicio.
- **Port Type**: conjunto abstracto de operaciones admitidas por uno o más puntos finales.
- **Binding**: Especificación del protocolo y del formato de datos para un tipo de puerto determinado.
- **Port**: punto final único que se define como la combinación de un enlace y una dirección de red.
- **Service**: colección de puntos finales relacionados.][7]

Una de las consecuencias más interesantes de esto es que pueden existir varias implementaciones de una única interfaz WSDL. Este diseño permite que sistemas dispares escriban implementaciones de la misma interfaz, para garantizar así la comunicación entre ellos. Si tres empresas diferentes implementan el mismo archivo WSDL y una parte del software de cliente crea el código auxiliar/proxy a partir de esa interfaz, dicho

[7] (Christensen, y otros, 2000)

software se podrá comunicar con las tres implementaciones con el mismo código de base, cambiando simplemente el punto de acceso.

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri"?>

    <import namespace="uri" location="uri"/>*

    <wsdl:documentation.../> ?

    <wsdl:types> ?
        <wsdl:documentation.../>?
        <xsd:schema.../>*
        <-- extensibility element  --> *
    </wsdl:types>

    <wsdl:message name="nmtoken> *
        <wsdl:documentation.../>?
        <part name="nmtoken" element="qname"? type="qname"?/> *
    </wsdl:message>

    <wsdl:portType name="nmtoken">*
        <wsdl:documentation.../>?
        <wsdl:operation name="nmtoken">*
            <wsdl:documentation.../> ?
            <wsdl:input name="nmtoken"? message="qname">?
                <wsdl:documentation.../> ?
            </wsdl:input>
            <wsdl:output name="nmtoken"? message="qname">?
                <wsdl:documentation.../> ?
            </wsdl:output>
            <wsdl:fault name="nmtoken" message="qname"> *
                <wsdl:documentation.../> ?
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name="nmtoken" type="qname">*
        <wsdl:documentation.../>?
        <-- extensibility element  --> *
        <wsdl:operation name="nmtoken">*
            <wsdl:documentation.../> ?
            <-- extensibility element  --> *
            <wsdl:input name="nmtoken"?> ?
                <wsdl:documentation.../> ?
                <-- extensibility element  -->
            </wsdl:input>
            <wsdl:output name="nmtoken"?> ?
                <wsdl:documentation.../> ?
                <-- extensibility element  --> *
            </wsdl:output>
            <wsdl:fault name="nmtoken"> *
                <wsdl:documentation.../> ?
                <-- extensibility element  --> *
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="nmtoken"> *
        <wsdl:documentation.../>?
        <wsdl:port name="nmtoken" binding="qname"> *
            <wsdl:documentation.../> ?
            <-- extensibility element  -->
        </wsdl:port>
        <-- extensibility element  -->
    </wsdl:service>

    <-- extensibility element  --> *

</wsdl:definitions>
```

Ilustración 4: Documento WSDL, (Christensen, y otros, 2000)

CAPITULO II

Servicios Web

Los servicios WEB son una nueva clase de aplicaciones WEB. Son aplicaciones modulares auto-contenidas, auto-descriptivas que pueden ser publicadas, localizadas e invocadas a través del ambiente WEB. Los cuales pueden realizar funciones que van desde requerimientos simples a procesos empresariales complejos.

Como resultado de una combinación de las mejores aspectos de la programación orientada a componentes y la programación Web, surgen los servicios Web que presentan forma de módulos, los cuales pueden ser reutilizables sin que la implementación o el lenguaje, sistema operativo o modelo de componente utilizado en su generación sean un obstáculo. Lo que implica que el usuario no tiene necesariamente que saber qué se tiene instalado o cómo funciona para poder utilizar su funcionalidad, logrando así que su acceso se realice a través de protocolos de Internet basados en XML (HTTP o SMTP).

Por otro lado han surgido diferentes tecnologías para su construcción entre los que los desarrolladores pueden utilizar, lenguajes como Java, VB, C, C++, C#, Perl, PHP, Smalltalk, and Python para desarrollar Servicios Web consumados. Además de las dos plataformas de servicios web más populares: Java 2 Enterprise Edition (J2EE) and Microsoft .NET. Más adelante veremos cómo se implementa un Servicio Web desde cada una de estas plataformas y utilizándolo desde la otra para demostrar su interoperabilidad.

Definición

Según la W3C (el organismo que se encarga de desarrollar gran parte de los estándares de internet), se define un Servicio Web de la siguiente forma: “Un servicio Web es una aplicación software identificada mediante una URI (Uniform Resource Identifier), cuya interfaz y uso es capaz de ser definido, descrito y descubierto mediante artefactos XML, y soportar interacciones directas con otras aplicaciones de software, usando mensajes basados en XML y protocolos basados en Internet”. Qué, Además de ser una definición un tanto complicada, uno llega a la conclusión de que es tan genérica que millones de cosas pueden ser un Servicio Web. Sin embargo, cuando los desarrolladores hablamos de Servicio Web nos estamos refiriendo a tecnologías muy concretas, al menos la gran mayoría de las veces.

Servicios Web es un estándar de comunicación entre procesos y/o componentes, diseñado para ser multiplataforma y multilenguaje, es decir, no importa en qué lenguaje este programado ya sea Visual Basic, Java, C# o en que plataforma este corriendo, ya sea Linux, Unix o Windows éstos serán accesibles y utilizables por otras aplicaciones desarrolladas en otras plataformas o lenguajes de programación. Antiguamente se utilizaban estándares como DCOM (Distributed Component Object Model) introducido por Microsoft e implementado por otras plataformas, y CORBA (Common Object Request Broker Architecture) introducido por el OMG (Object Management Group) e implementado en distintas plataforma incluido Windows. Estos estándares tenían bastantes problemas de configuración, especialmente en entornos en que se encontraban firewalls de por medio en los cuales era imposible habilitar ciertos puertos de comunicación para que estos componentes funcionaran. De esta manera la preferencia

por utilizar el puerto 80 de HTTP, que normalmente se encuentra habilitado en la mayoría de los servidores y firewalls debido al uso de navegadores y servidores Web, no traería mayores complicaciones el uso de una tecnología que use este protocolo y puerto de TCP/IP.

[Servicios Web son una Interfaz de Internet basada en protocolos para el procesamiento. Servicios Web por lo general no tienen estado y siguen un modelo de interacción de petición/respuesta. El estándar de Servicios Web provee una forma para localizar interface e intercambiar datos en una forma entendible. Para aplicaciones de Intranet, Los mensajes SOAP pueden ser recibidos directamente por aplicaciones de servidor. Por Aplicaciones Extranet, por cuestiones de seguridad, SOAP mensajes son recibidos por Servidores Web o integración de servidores que pasan los mensajes en la apropiada aplicación. Por otro lado está la interfaz hacia el servicio, no hay requerimiento de cómo los servicios son proveídos. Por ejemplo, un Servicios Web][8]

Ahora veremos un ejemplo de cómo los Servicios Web pueden ser provistos.

[8] (Hartman, y otros, 2003)

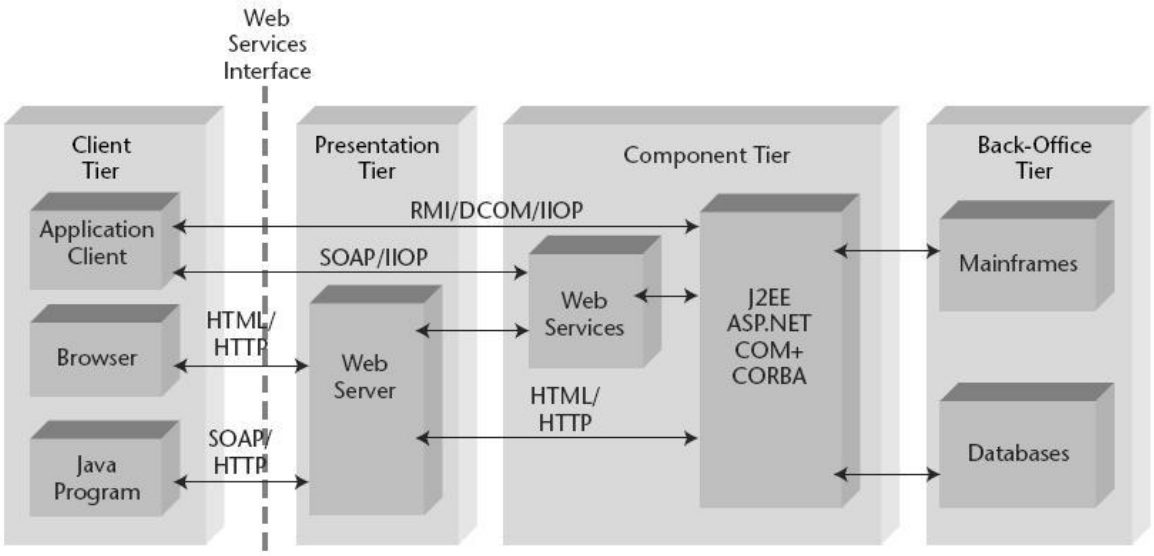


Ilustración 5: Ambiente Típico de un Servicios Web, (Hartman, y otros, 2003)

Características

Los servicios Web tienen la habilidad de solucionar una gran cantidad de problemas a nivel empresarial que reducen costos de programación, operacionales y de respuesta. Los servicios web han surgido dada su naturaleza como una forma muy practica de solucionar dichos inconvenientes.

Muchos negocios sufren de problemas de integración por la rápida evolución y los constantes cambios en las diferentes tecnologías, condiciones del mercado hasta por las diferentes relaciones empresariales. Por ende es de vital importancia que los sistemas internos de su empresa se puedan comunicar con los sistemas de sus compañías asociadas y bases de datos. La rapidez y fácil integración facilita y hace más poderoso los procesos de su negocio. Aún así los negocios experimentan muchos problemas en esta

área, por los diferentes lenguajes de las base de datos, diferentes protocolos de comunicación, y diferentes formas de expresas problemas en lenguajes entendidos por computadoras, integras sistemas es extremadamente difícil.

Así que, tenemos problemas de integración, pero queremos solucionarlos de manera rápida y eficiente. Las personas quieren ver los resultados de sus investigaciones lo más rápido posible, ¿Cómo se puede replantear los activos existentes sin que sea nocivo para los procesos de la compañía? ¿Qué tan rápido se pueden cambiar dadas las condiciones del mercado? El problema con las soluciones en el pasado es que los esfuerzos integración tomaban demasiado tiempo, y se habían creado nuevas, inflexibles y arquitecturas difícil de cambiar.

Aunque parezca una paradoja, una de las razones principales para que los negocios adoptaran tan rápidos los Servicios Web fue que los otros negocios los había adoptados. Este “acuerdo” solo se llevó a cabo por la tecnología puede solucionar los problemas de integración, proveyendo un lenguaje común que se pudiera usar en la integración entre las empresas. Sin acuerdo entre un lenguaje común, no hubiera interoperabilidad.

En el pasado, ha habido batallas sobre que protocolos y que lenguajes usar. En un tiempo, hubo un debate sobre si TCP/IP sería el protocolo de red dominante. Cuando se convirtió en el protocolo dominante para el internet, otros protocolos usados como fundamentos para el transporte, incluyendo el HTTP, el cual se convirtió en el protocolo para ser usado sobre la Web. Http se convirtió ampliamente soportado por aplicaciones de protocolos de capa, y SOAP fue desarrollado usando HTTP como fundamento. Ahora la mayoría de los

negocios han adoptado SOAP como medio de comunicación entre las aplicaciones y servidores, esto asegura que cada una de las aplicaciones tenga la oportunidad de hablar un lenguaje común. Los Servicios Web están basados en SOAP y representan nuestro estado de evolución y en estándar de comunicación.

Porque hay tan amplio acuerdo y aceptación de los protocolos de los Servicios Web, ahora es posible potenciar el trabajo de las aplicaciones existentes y convertirlas en un Servicio Web usando los protocolos estándares de los servicios Web que todo el mundo entiende. Los servicios Web te permiten cambiar las interfaces de las aplicaciones, sin necesidad de reescribirlas, usando una capa intermedia. Un ejemplo del impacto de lo sencillo que es usar una capa intermedia, los clientes y servidores de .Net pueden hablar con servidores de J2EE usando SOAP. La implementación de las capas inferiores no es relevante, solo importa los medios de comunicación.

Arquitectura

La arquitectura de los Servicios Web se puede decir que consiste en siete partes como dicen en el libro *Servicios web Architecture and Its Specifications: Essentials for Understanding WS* de *Luis Felipe Cabrera y Chris Kurt*. Las cuales son: MetaDatos, Seguridad, Transacciones, Confiabilidad en la mensajería, y la mensajería misma, la siguiente capa son los estándares de XML y SOAP, y por ultimo son los protocolos que permiten esta tipo de comunicación. Ahora trataremos de profundizar en cada uno de ellos.

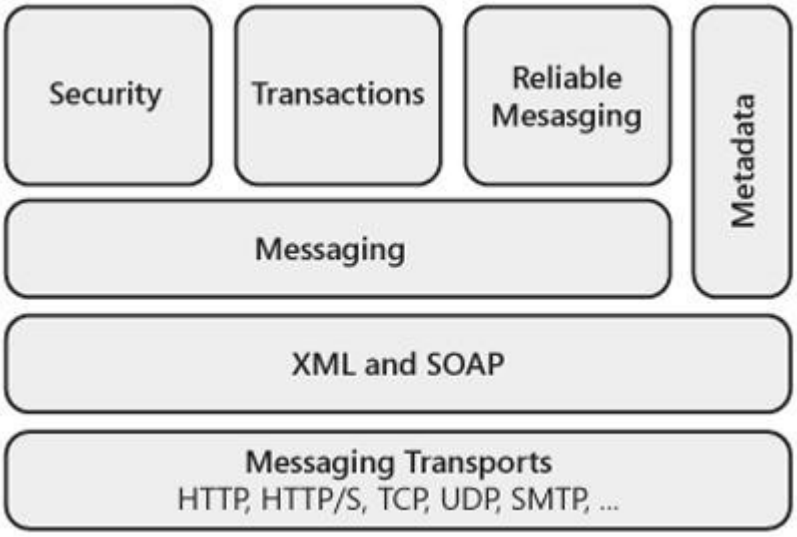


Ilustración 6: Arquitectura de los Servicios Web, (Cabrera, y otros, 2005)

Mensajería

Como ya hemos dicho el tipo de mensajería que se maneja en los Servicios Web son los antes mencionados documentos XML definidos por un modelo abstracto de datos el cual es compatible con texto basado en XML 1.0 y cada una de sus diferentes especificaciones, como son XML Schemas, XML Query, XSLT 2.0 el cual se llama XML Information Set. El XML Information Set o Conjunto de Información tiene unos artículos de información los cuales son: Document, Element, Attribute, Namespace, Character, Comment.

Los servicios de mensajería de los Servicios Web son mensajes en SOAP de tal manera que pudiera tomar completa ventaja del XML Information Set. El sistema de artículos posibles de la información traza generalmente a las varias características en un documento de XML, tal como elementos, cualidades, namespaces, y comentarios. Cada artículo de la

información tiene un sistema asociado de las características que proporcionan una descripción más completa del artículo. El hecho de que la carga útil del mensaje y los encabezados del protocolo emplean el mismo modelo se puede utilizar para asegurar la integridad de los encabezados de la infraestructura así como cuerpos del uso. La infraestructura y los usos de la mensajería pueden encaminar los mensajes basados en el contenido de los encabezados y de los datos dentro del mensaje. Las herramientas que se han desarrollado para el modelo de los datos de XML se pueden utilizar para examinar y construir mensajes completos. Estas ventajas no estaban disponibles en arquitecturas tales como DCOM, CORBA, y RMI, donde detalles como los encabezados del protocolo infraestructurales opacaban la aplicación.

La flexibilidad de la mensajería proporcionada por SOAP permite que los servicios se comuniquen con una variedad de patrones en el intercambio de los mensajes, satisfaciendo los requisitos de usos distribuidos. El uso de llamadas de procedimiento remoto, por ejemplo, popularizó el patrón síncrono del intercambio del mensaje de petición / respuesta. Cuando los estados latentes de la entrega del mensaje son incontrolados, la mensajería asincrónica es necesaria. Cuando se utiliza el patrón asincrónico de petición / respuesta, la correlación explícita del mensaje llega a ser obligatoria.

SOAP es definido independientemente del mecanismo subyacente del transporte de la mensajería en funcionamiento. Permite el uso de muchos transportes alternativos para el intercambio de mensajes y permite transferencia síncrona y asincrónica y el proceso del mensaje.

Puesto que los protocolos del servicio del Web se diseñan para ser totalmente independiente del transporte subyacente, la selección del mecanismo apropiado se puede diferir hasta el tiempo de ejecución. Esto no prohíbe a usos del servicio del Web la flexibilidad de determinar el transporte apropiado mientras que se envía el mensaje. Además, el transporte subyacente puede cambiar mientras que el mensaje se encamina entre los nodos, y, otra vez, el mecanismo seleccionado para cada salto puede variar como sea necesario.

Para los mensajes que se encaminarán y los servicios que se tratarán en este ambiente de la mensajería del multitransporte, se requiere un mecanismo de dirección común. La especificación de direccionamiento de los Servicios Web define el modelo y la sintaxis para un encaminamiento y tratar el jefe de la información de mensaje. El encabezado de la información de mensaje para tratar se describe en términos de su Conjunto de Información. Un atascamiento del SOAP también se define. Al usar el atascamiento del SOAP, la dirección del mensaje se hace con los bloques del encabezado del SOAP. Tratando la información se asume para ser inmutable. Los intermediarios del SOAP pueden no modificar la dirección de la información a lo largo de la trayectoria del mensaje

Metadatos

Todas las interacciones del Servicios Web son realizadas intercambiando mensajes del SOAP. Para prever un desarrollo robusto y un ambiente operacional, los servicios se describen usando meta datos legibles por la máquina. Los metadatos proporcionan la infraestructura del sistema que se puede utilizar en una variedad de maneras. Los metadatos apoyan la automatización y las herramientas, permiten descubrimiento

automatizado del servicio, promueven la automatización de la comunicación entre los servicios, y aumentan interoperabilidad entre los servicios. Por diseño, los meta datos del Servicios Web responden a varios propósitos. Los tres propósitos más importantes de metadatos se contornean aquí. Se utiliza para describir los formatos de intercambio del mensaje que el servicio puede apoyar y describir los patrones válidos del intercambio del mensaje de un servicio. Los metadatos también se utilizan para describir las capacidades y los requisitos de un servicio. Esta última forma de metadatos se le llama política de un servicio. Los formatos del intercambio de mensaje y los patrones del intercambio de mensaje se expresan en el idioma descriptivo del servicio del Web (WSDL). Se expresan las políticas usando WS-Política. Se expresan los contratos usando las tres clases de metadatos apenas descritos. Los contratos son las abstracciones que aíslan usos de los detalles internos de la puesta en práctica de los servicios que confían.

Representa el lenguaje basado en XML usado para las descripciones de los metadatos en el Servicios Web. WSDL describe los patrones del intercambio del mensaje, la Política de los Servicios Web describe capacidades y requisitos, y “otros métodos” son representados por el área fuera de los óvalos más pequeños. El hecho de que WSDL está contenido totalmente en contrato significa que toda su información está utilizada en el contrato de un servicio.

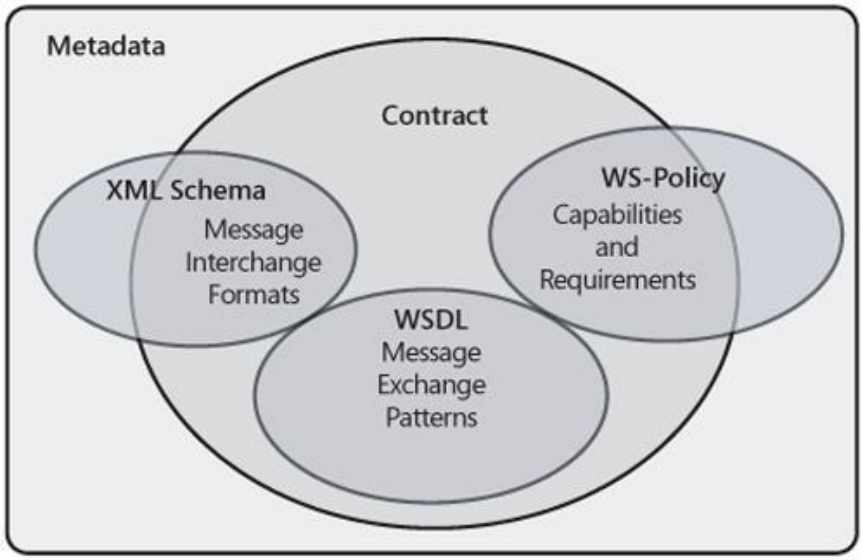


Ilustración 7: Componentes de los Metadato, (Cabrera, y otros, 2005)

WSDL era el primer mecanismo extensamente adoptado para describir las características básicas de un servicio del Web. Los mensajes descritos en WSDL se agrupan en las operaciones que definen los patrones básicos del mensaje. Las operaciones se agrupan en las interfaces llamados de puertos que especifican un contrato abstracto para un servicio. Finalmente, los puertos y los atascamientos se utilizan para asociar tipos de puertos a transportes concretos y a la información física del despliegue. Una descripción de WSDL es un primer paso para la identificación automática de todas las características del servicio y para permitir las herramientas de desarrollo del software.

WSDL especifica lo que debe contener un mensaje de la petición y lo que parecerá el mensaje de respuesta en una notación inequívoca. La notación que un archivo de WSDL utiliza para describir formatos del mensaje se basa en esquema de XML. Esto significa que es tanto un lenguaje de programación neutral como una base estándar, que se hace conveniente para describir las interfaces del servicio que son accesibles de una variedad amplia de plataformas y de lenguajes de programación. Además de describir contenido

del mensaje, WSDL puede definir donde está disponible el servicio y qué protocolo de comunicaciones se utiliza para hablar con el servicio. Esto significa que el archivo de WSDL puede especificar los elementos bajos requeridos escribir un programa para obrar recíprocamente con un Servicios Web. Varias herramientas están disponibles leer un archivo WSDL y para generar el código requerido para producir los mensajes sintácticamente correctos para un Servicios Web.

Seguridad

La seguridad es un aspecto fundamental de los sistemas informáticos, especialmente esos sistemas abarcados con Servicios Web. La seguridad tiene que ser robusta y eficaz, porque los sistemas pueden hacer solamente asunciones de conexiones sobre el formato de mensajes y de los intercambios legales del mensaje, toda la seguridad tiene que ser construida basada en mecanismos y asunciones explícitas acordadas. La infraestructura de la seguridad debe ser bastante flexible, apoyar la variedad amplia de políticas de la seguridad requeridas por diversas organizaciones.

La seguridad de los Servicios Web se basa en el requisito que los mensajes entrantes permiten probar un sistema de aserciones hechas sobre un remitente, un servicio, o un cierto otro recurso. Llamamos estas demandas aserciones de la seguridad. Los ejemplos de las demandas de la seguridad incluyen identidad, cualidades, la posesión dominante, permisos, y capacidades. Todas las aserciones de la seguridad se codifican en símbolo de seguridad. A menudo, el símbolo de la seguridad es símbolo binario envuelto en XML. El símbolo de la seguridad se utiliza para la autentificación y la autorización así como para mensajes de firma y que cifran. En terminología tradicional de la seguridad, este símbolo

de la seguridad representa una mezcla de capacidades y de controles de acceso. Mensajes que no contienen o referencian por lo menos un símbolo de la seguridad no puede ser autenticada, autorizado, o asegurado con las firmas y el cifrado.

En general el modelo de seguridad de los Servicios Web apoya varios modelos más específicos de la seguridad, tales como autorización identidad-basada, listas del control de acceso, y autorización basada en capacidad. Permite el uso de tecnologías existentes tales como certificados públicos dominantes X.509, símbolos basados en XML. En el modelo de la seguridad es suficiente construir los sistemas que utilizan acercamientos más sofisticados para el intercambio dominante de alto nivel, la autentificación, el control de acceso política y las relaciones complejas de la confianza. Los poderes y los servicios de confianza pueden también ser utilizados. Por ejemplo, un servicio de confianza se puede construir para hacer cumplir una política de la seguridad en un límite de la confianza; los mensajes que van fuera del límite se cifran mientras que los que permanecen dentro del límite son desencriptados. Esta flexibilidad y grado de sofisticación no está presentes en soluciones anteriores. La forma de Implementar los diferentes sistemas de seguridad en los Servicios Web las veremos más adelante en el capítulo 4.

Sistema de Identificación

El Sistema de Identificación de un Servicio Web es un habilitador dominante para automatizar conexiones a los servicios sin la intervención humana. El acercamiento del servicio del Web al sistema de identificación refleja los dos acercamientos más comunes a encontrar la información en un sistema informático distribuido: el mirar en una localización bien conocida o difundir una petición a todos los oyentes disponibles. El

servicio de los registros de UDDI como el directorio, y los protocolos del sistema de identificación se utilizan para difundir peticiones.

UDDI especifica un protocolo para preguntar y poner al día un directorio común de la información de Servicios Web. El directorio incluye la información sobre abastecedores de servicio, los servicios que reciben, y los protocolos instrumento de esos servicios. El directorio también proporciona mecanismos para agregar meta datos a cualquier información registrada. El acercamiento del directorio de UDDI puede ser utilizado cuando la información del Servicios Web se almacena en localizaciones bien conocidas. Una vez que se localice el directorio, una serie de peticiones de la pregunta se puede enviar para obtener la información deseada. Las localizaciones del directorio de UDDI se obtienen fuera de banda, generalmente con datos de la configuración de sistema.

Los abastecedores de Servicios Web tienen varias opciones para desplegar registros de UDDI. Los panoramas del despliegue caen en una de tres categorías: público, adicional-empresa e infra-empresa. Para apoyar a despliegues públicos, a un grupo de vendedores conducidos por Microsoft, IBM, y anfitrión de SAP el registro del negocio de UDDI [UBR]. El UBR es un registro del público UDDI que se repliega a través de organizaciones anfitrionas múltiples, sirviendo como un recurso para los Servicios Web basados en internet y una prueba para los desarrolladores de Servicios Web. Mientras que la puesta en práctica del público UDDI ha recibido la mayoría de la atención hasta la fecha, los adeptos tempranamente utilizan el acercamiento del suplemento y de la infra-empresa más a menudo. En estos dos panoramas del despliegue, un registro privado es desplegado por una organización y un control mucho más apretado sobre los tipos de información registrados es posible. Estos registros privados se pueden dedicar a solamente una

organización o a los grupos de socios de negocio. UDDI también define los protocolos para la réplica entre los registros y para la federación de la confianza a través de despliegues.

Para todos los panoramas del despliegue, los directorios de UDDI contienen la información detallada sobre Servicios Web y donde se reciben. Una entrada en la guía de UDDI tiene tres porciones primarias: el abastecedor de servicio, Servicios Web ofrecidos, y atascamientos a las puestas en práctica. Cada uno de estas piezas proporciona una información progresivamente más detallada sobre el Servicios Web. La información de carácter general describe el abastecedor de servicio. Esta información no se apunta en el software de los Servicios Web sino en un revelador o un ejecutor que necesite entrar en contacto con a alguien responsable del servicio directamente. La información del abastecedor de servicio incluye nombres, direcciones, contactos, y otros detalles administrativos. Todas las entradas de UDDI tienen elementos múltiples para las descripciones del multilenguaje.

La lista de los servicios disponibles del Servicios Web se almacena dentro de una entrada del abastecedor de servicio. Estos servicios se pueden organizar, dependiendo de su uso previsto; pueden ser agrupados en área de aplicación, la geografía, o cualquier otro esquema que sea apropiado. La información de servicio almacenada en un registro de UDDI incluye simplemente una descripción del servicio y un indicador a las puestas en práctica que el Servicios Web contiene. Los acoplamientos a los servicios recibidos por otros abastecedores, llamados Service Projections, pueden también ser colocados. La parte final de una entrada del abastecedor de servicio de UDDI es el atascamiento a una puesta en práctica. Este atascamiento asocia la entrada del Servicios Web al URI exacto para identificar donde se despliega el servicio, especifica el protocolo para utilizar el

acceso, y contiene referencias a los protocolos exactos que se ponen en ejecución. Estos detalles son suficientes para que un desarrollador escriba una aplicación que invoque el Servicios Web. La definición detallada del protocolo se proporciona a través de una entidad de UDDI llamada un tipo modelo (o tModel). En muchos casos, el tModel se refiere a un archivo de WSDL que describe el SOAP de la interfaz del Servicios Web, pero los tModels son también bastante flexibles para describir casi cualquier clase de recurso.

El sistema de identificación dinámico del servicio del Web se proporciona de una manera diferente. Como alternativa a almacenar la información en un registro sabido, los Servicios Web dinámicamente descubiertos anuncian explícitamente su llegada y salida de la red. El sistema de identificación de un Servicios Web define protocolos para anunciar y para descubrir servicios del Web a través de mensajes de multicast. Cuando un Servicio Web se conecta con una red, anuncia su llegada enviando un mensaje hola. En el caso más simple, estos avisos se envían a través de la red usando llamada del protocolo del Servicio Web de multicast esto una red ad hoc. Este acercamiento también reduce al mínimo la necesidad de la interrogación en la red. Para limitar la cantidad de red traficar y optimizar el proceso del sistema de identificación, un sistema puede incluir un proxy del sistema de identificación. Un proxy de sistema de identificación substituye la necesidad de enviar mensajes del multicast con una localización bien conocida del servicio, transformando una red ad hoc en una red manejada. Usando la información de la configuración, las colecciones de servicios de proxy se pueden ligar juntas para escalar el servicio del sistema de identificación a los grupos de servidores, escalando a partir de una máquina a muchos. Porque los proxy son ellos mismos del sistema de identificación del Servicios Web, pueden anunciar su presencia con su propio mensaje especial de bienvenida. Los Servicios Web que reciben este mensaje pueden entonces aprovecharse de los servicios del proxy y se requieren no más para utilizar los protocolos del sistema de identificación.

Cuando un servicio sale de una red, el sistema de identificación del Servicios Web especifica un mensaje despedida que se enviará a la red o al proxy del sistema de identificación. Este mensaje informa a los otros servicios en la red que el Servicio Web que sale está no más disponible.

Manejo de la Confiabilidad de los mensajes y transacciones

Muchas condiciones pueden interrumpir un intercambio de mensajes entre dos servicios. Esto es especialmente un problema cuando no es fiable transportar por protocolos tales como HTTP 1.0 y el smtp se utiliza para la transmisión o cuando un intercambio del mensaje atraviesa conexiones múltiples de la capa de transporte. Los mensajes pueden ser perdidos, ser duplicados, o ser reordenados, y los Servicios Web pueden fallar y perder el estado volátil. La Confiabilidad de los mensajes son un protocolo que permite la entrega confiable "end-to-end" de los mensajes basados en características específicas del aseguramiento de la entrega. La especificación define tres diversos aseguramientos de la entrega del mensaje que se puedan utilizar en la combinación:

- Al menos un mensaje entregado.
- Máximo un mensaje entregado.
- Los mensajes se entregan en el orden en que se enviaron.

La combinación de al menos una vez y máximo una vez de esta forma nos aseguramos que los mensajes hayan llegado al lugar exacto de la forma más segura. Debido al diseño de transporte independiente de la arquitectura de los Servicios Web, todos los aseguramientos de la entrega están garantizados con independencia del transporte de la

comunicación o de la combinación de los transportes usados. Usar el manejo de confiabilidad simplifica el desarrollo del sistema debido a el número más pequeño de los modos de fallo potenciales de la entrega que un revelador debe anticipar al programar un servicio

Al usar el sistema de confiabilidad, los participantes deben reconocer el protocolo basado en la información enviada en encabezados de mensajes SOAP. El sistema de mensajes transmitidos como grupo se refiere como secuencia del mensaje. Una secuencia del mensaje es establecida por el iniciador o remitente, y tiene que ser aceptada por el último Servicio Web receptor. Al establecer una asociación a dos caras, el iniciador ofrece una secuencia, con su identidad, al recipiente. Cuando el iniciador procura establecer una secuencia, el último receptor es informado de esta petición por el uso de la operación de Create Sequence. Si el último receptor acuerda colaborar en este comportamiento, esto es indicado por un mensaje acertado de Create Sequence Response. Un rechazamiento significa que la secuencia no está establecida. La entrega confiable del mensaje es un comportamiento que no requiere un coordinador explícito. Una infraestructura de software que implementa la confiabilidad del protocolo de la mensajería simplifica el código de los desarrolladores que tienen que escribir a los mensajes de la transferencia bajo aseguramientos del transporte que varían. Cuando se utiliza una plataforma que apoya el protocolo, es la infraestructura subyacente que verifica que los mensajes se hayan transferido correctamente entre los puntos finales, retransmitiendo mensajes cuando son necesarios. Las aplicaciones no necesitan ninguna lógica adicional manejar las retransmisiones del mensaje, la eliminación duplicada del mensaje, o el reconocimiento del mensaje que se puede requerir para proporcionar los aseguramientos de la entrega.

La puesta en práctica de sistema de manejo de confiabilidad de mensajes se distribuye a través del iniciador y del servicio, según las indicaciones de la Figura 8. Esas características que no son visibles “on the wire,” por ejemplo orden de expedición del mensaje, son proporcionadas por la puesta en práctica de la especificación del sistema de manejo de confiabilidad. La figura representa la transmisión de un mensaje usando el protocolo confiable de la mensajería. El primer paso está para los usos del punto final que interactúa con la aplicación como REMITENTE y RECEPTOR para establecer las condiciones previas para intercambiar mensajes. El REMITENTE entonces transmite un mensaje en el paso 2. En el paso 3, la infraestructura de la mensajería del REMITENTE cambia formato el mensaje como apropiado para el transporte funcionando. El transporte de la mensajería la envía al recipiente apropiado en el paso 4. Finalmente, la capa de la mensajería en el RECEPTOR lo transforma para ser conveniente para el uso del RECEPTOR en el paso 5.

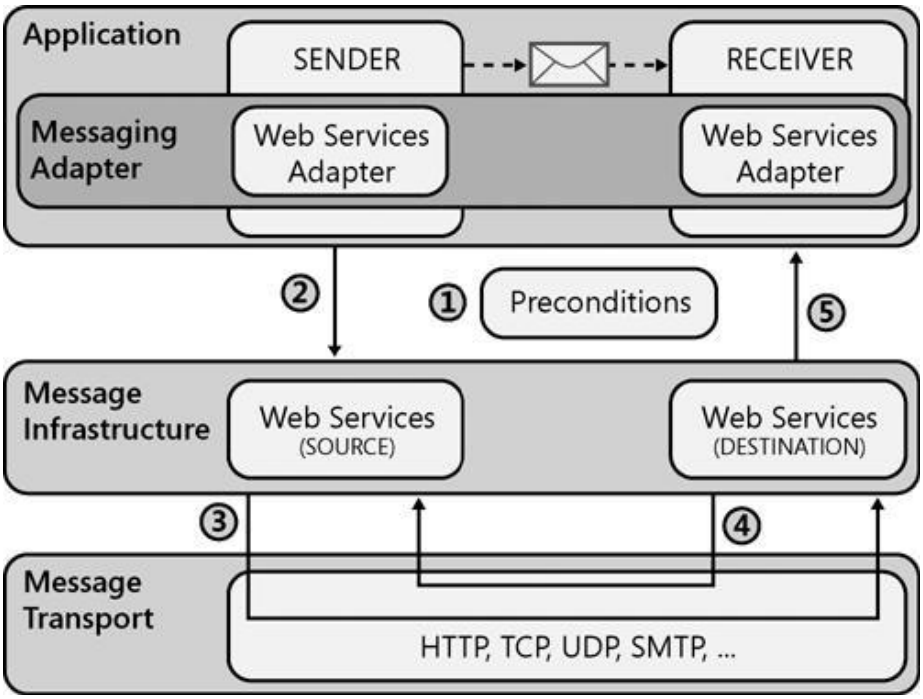


Ilustración 8: Capas de Implementación del Manejo de Confiabilidad de los Mensajes, (Cabrera, y otros, 2005)

La coordinación explícita se requiere en ciertas situaciones. La interacción básica coordinada está entre dos participantes, una coordinación bipartita. Cualquier coordinación multipartita se puede acopiar de la coordinación bipartita sucesivamente ensamblando en más participantes según lo necesitado. La coordinación bipartita puede ser espontánea, o puede ser que requiera a coordinador señalado. El patrón síncrono de la mensajería "solicitar-respuesta" es un ejemplo de un protocolo espontáneo popular de la coordinación y es una de las formas más simples de coordinación del acuerdo. Para cada petición del trabajo, el Servicio Web receptor debe terminar todo el trabajo previsto antes de volver cualesquiera datos al solicitante. Ambas partes siguen este patrón terminante sin necesidad de un servicio explícito de la coordinación.

Algunas familias de los protocolos de la coordinación de la N-manera requieren a coordinador señalado al pastor, una unidad del trabajo con un número de servicios de cooperación del participante. Un ejemplo es cuando las actividades se deben coordinar entre los servicios que no son totalmente esperados para ser conectado al mismo tiempo. Mientras cada participante y el coordinador se comuniquen en algún momento, la coordinación puede suceder y el acuerdo en el resultado puede ser alcanzado. La arquitectura de los Servicios Web define algunas operaciones simples para los coordinadores señalados. La especificación de la coordinación de los Servicios Web define un marco extensible de la coordinación para apoyar los panoramas en los cuales requieren a los coordinadores explícitos. Este protocolo introduce un bloque de encabezado SOAP, llamado un contexto de la coordinación, para identificar únicamente la parte del trabajo común que debe ser emprendido. Para iniciar una parte común de trabajo, un Servicios Web envía un contexto de la coordinación a unos o más servicios.

Poner el recibo de alarmas de un contexto de la coordinación a un servicio receptor que la colaboración común está solicitada. El contexto de la coordinación contiene bastante información para que el recipiente de la petición se determine si participar en el trabajo. La información exacta contenida dentro del contexto de la coordinación varía dependiendo de la clase de trabajo que se solicite. El sistema de tipos de la coordinación es ampliable. Los nuevos tipos se pueden definir por una puesta en práctica, mientras cada servicio que participa en el trabajo común tenga una comprensión común del comportamiento requerido. Por ejemplo, las transacciones atómicas son uno de algunos tipos iníciales de la coordinación de la piedra angular que se han definido en la arquitectura de los Servicios Web.

Si se entiende y se acepta el tipo solicitado de la coordinación, el Servicio Web utiliza el protocolo del registro de la coordinación del mismo para notificar al coordinador y para participar en el trabajo común. Un contexto de la coordinación incluye una referencia de punto final para el coordinador y los identificadores de los comportamientos posibles que pueden ser seleccionados. La operación de registro especifica el comportamiento apoyado por el Servicio Web que participa. Una vez que el mensaje del registro se envíe al coordinador, el Servicio Web participa en el trabajo según los protocolos que ha suscrito. El registro es la operación dominante en el marco de la coordinación. Permite el "cableado conjunto" de diversos Servicios Web que deseen coordinar para realizar una unidad común del trabajo.

Las Transacciones Atómicas de los Servicios Web especifican transacciones ACID tradicionales para los mismos. La ayuda establecida para este comportamiento ampliamente utilizado incluye un coordinador señalado que es independiente de todos

los participantes. Dentro del contexto del tipo atómico de la coordinación de la transacción, se definen tres protocolos: un protocolo de la terminación y dos variantes de un bifásico confían el protocolo (2PC). Un Servicio Web registrado para la terminación tiene la capacidad de decir al coordinador señalado cuando empezar a confiar el proceso. Este protocolo también define mensajes para comunicar el resultado final de la transacción al iniciador. Sin embargo, el protocolo no requiere que el coordinador se asegure de que el resultado fuera procesado por el iniciador. En cambio, otros comportamientos en las transacciones atómicas requieren un coordinador para asegurarse de que los participantes procesen los mensajes de la coordinación.

El protocolo 2PC trae a todos los participantes registrados a una decisión común de confianza o aborta la transacción, y se asegura de que todos los participantes sean informados del resultado final. Mientras que su nombre indica, utiliza dos fases de notificaciones para terminar la transacción. Dos variantes de este protocolo se definen: 2PC volátil y 2PC durable. Ambos protocolos utilizan los mismos mensajes en “wire corresponding” a las operaciones de “preparar, confiar y cancelar”, pero, 2PC volátil no tiene ningún requisito de la durabilidad. El protocolo volátil 2PC debe ser utilizado por los participantes que manejan recursos volátiles, tales como encargados del escondrijo o encargados de ventana. Al coordinador entran en contacto en una primera fase de notificaciones y no requieren a estos participantes en segunda fase de notificaciones. El protocolo durable 2PC debe ser utilizado por los participantes que manejan recursos durables tales como bases de datos y archivos. Cuando se ha iniciado un proceso de confianza, entran en contacto con estos participantes por primera vez después de que hayan entrado en contacto con todos los participantes volátiles 2PC. Los participantes durables 2PC requieren que las dos fases completas de notificaciones alcancen el comportamiento de todo o nada impuesto por el coordinador y terminar la transacción.

Estos comportamientos son los más apropiados para los panoramas en los cuales los recursos se pueden llevar a cabo para la duración de la transacción y las transacciones son típicamente de muy breve duración. Este protocolo garantiza que debajo del proceso normal del coordinador, entra en contacto con todos los participantes con el resultado de la primera fase.

Para las transacciones que se espera que requieran más hora de terminar, o cuando los recursos tales como cerraduras no pueden ser llevados a cabo, los comportamientos alternos son definidos por otros protocolos de la coordinación. Al poner a un coordinador en ejecución que apoye el comportamiento durable 2PC, es algo natural que sea un servicio que es independiente de todos los servicios del participante. Esto es porque el estado tiene que ser preservado hasta que han alcanzado a todos los participantes y haber contestado explícitamente. Así, si algún servicio del participante fuera del coordinador usa una transacción durable 2PC, tendría que estar disponible para procesar el protocolo hasta que el resto de los participantes hayan alcanzado la terminación. Este requisito puede ser inaceptable, dado que no hay hora a priori limitada en cuánto tiempo un participante puede tomar para responder.

Las transacciones atómicas de los Servicios Web tienen varias ventajas sobre los protocolos existentes de la coordinación para las transacciones atómicas tradicionales. Primero, el protocolo basado en SOAP, permitiendo su uso por los participantes que se han desarrollado en diversas plataformas. Este grado de interoperabilidad no se encuentra en protocolos anteriores. En segundo lugar, compone con la seguridad y la el manejo de Confianza de los Servicios Web, de tal modo proporcionando la operación end-to-end segura en topologías arbitrarias de la red. Tercero, es “cortafuego amistoso,”

permitiendo a Servicios Web utilizarlo aun cuando que se despliegan en diversos dominios de la confianza.

Un patrón que ha demostrado ser muy útil al construir sistemas distribuidos es el uso de colas durables transaccionales que proporcionan entrega asincrónica store-and-forward del mensaje. En este patrón, las transacciones atómicas se explotan en cada uno de los puntos finales de la transmisión. En el lado del remitente, el uso de envía-entrega de mensajes a una cola durable de una manera transaccional atómica de la cual el uso y el encargado de la cola ambos de Transacciones Atómica de los Servicios Web. El mensaje se considera entregado con éxito a la cola solamente si no hay excepción de proceso. Entonces, el subsistema de la cola asume el control de la entrega del mensaje entre la cola que origina y la cola que recibe. Este paso de la transmisión puede ser hecho a la vez que es diferente que cuando el mensaje fue puesto en la cola que originaba. Además, la localización de la cola que origina no necesita coincidir con la localización de uso de la cual el mensaje fue originado. Análogamente, el uso que recupera el mensaje de la cola receptora hace uso de las transacciones atómicas. De esa manera, un mensaje se puede recuperar de la cola solamente cuando no hay errores de proceso.

Las actividades de negocio en los Servicios Web especifican dos protocolos para las transacciones duraderas. En vez de sostener se detiene en los recursos hasta que la transacción está confiada, dichas actividades que en la especificación se basa en acciones que compensan. El modelo subyacente de la transacción es el supuesto de abrir la transacción jerarquizada. Estos protocolos codifican cómo los pares de servicios débilmente acoplados alcanzan el acuerdo que han terminado una tarea común. En un protocolo, el coordinador comunica explícitamente a participantes que no están

solicitando más trabajo en nombre de la tarea común. En el segundo protocolo, el participante es el que notifica al coordinador que el trabajo en nombre de la tarea común se ha terminado. El uso de acciones compensatorias proporciona un mecanismo para acabar operaciones tentativas sin dejar las cerraduras en ellas. Una operación de la remuneración debe ser publicada si, para cualquier razón, el sistema desea deshacer los efectos de la operación tentativa acabada. El uso de estos protocolos ha sido experimental hasta la fecha.

Enumeración, Transferencia y Eventos

Muchos escenarios requieren intercambio de datos usando más que un simple par de mensajes de petición/respuesta. Los tipos de aplicaciones que requieran estos intercambios de datos más largos incluyen preguntas de la base de datos, fluidez de datos, el trasversal de la información tales como namespaces, y enumeración de listas. La enumeración, particularmente, se alcanza estableciendo sin embargo una sesión entre la fuente de datos y el solicitante. Se establece esta sesión usando la operación de la enumeración, que proporciona un contexto de la enumeración que entonces se utilice en operaciones subsecuentes. Los mensajes sucesivos dentro de la sesión transportan la colección de elementos que son recuperados. No se hace ningunas asunciones en el acercamiento usado por el servicio para organizar los artículos que serán producidos. Qué espera bajo circunstancias de procesos normales, la enumeración producirá todos los datos subyacentes antes del final de la sesión.

La enumeración en los Servicios Web especifica protocolos para establecer una sesión de enumeración y para recuperar secuencias de datos. Los protocolos de la enumeración

permiten que la fuente de datos proporcione una abstracción de la sesión, llamada contexto de la enumeración, al servicio que se consume. Este contexto representa un cursor lógico con una secuencia de los artículos de datos. El solicitante entonces utiliza dicho contexto sobre un palmo de unos o más mensajes de SOAP para solicitar los datos. Los datos enumerados se representan como conjunto de información de XML. La especificación también permite que una fuente de datos proporcione un mecanismo de encargo para comenzar una nueva enumeración. Porque una sesión de la enumeración pudo requerir varios intercambios del mensaje, el estado de la sesión debe ser conservado.

Tres operaciones más de petición/respuesta se definen en la Enumeración de los Servicios Web: Renew, Get Status, y Release. La operación renew se utiliza para ampliar la validez de un contexto de la enumeración. La operación Get Status permite al invocador solicitar la información sobre un contexto dado de la enumeración. La operación release representa la parada y el abandono falta-libres de los recursos de la sesión. Finalmente, hay un mensaje consultivo enviado por fuentes de datos cuando una enumeración se termina de forma inesperada.

La información del estado con respecto al progreso de la iteración se puede mantener entre las peticiones por la fuente de datos o el servicio que se consume. La Enumeración de los Servicios Web permite que la fuente de datos decida, en base de solicitación-por-solicitación, qué parte será responsable de mantener el estado para la petición siguiente. Esta flexibilidad permite varias clases de optimizaciones. Un ejemplo de la optimización está permitiendo que un servidor evitara de ahorrar cualquier estado del cursor entre las invocaciones. Porque los estados latentes del mensaje pueden ser grandes para un

servicio que apoya varias enumeraciones simultáneas, no preservar el estado pudo rendir ahorros substanciales en la cantidad de información total que debe ser mantenida. Mantener las puestas en práctica en los dispositivos recurso-obligados, tales como teléfonos celulares, no pudo mantener ninguna información de estado en ellos. Además de enumerar las entidades de los datos presentes en un Servicio Web, es conveniente poder realizar varias operaciones básicas en ellas. Estas operaciones se introducen en las especificaciones de la Transferencia de los Servicio Web.

Las operaciones básicas requeridas manejar las entidades de datos que acceden con Servicio Web se definen en la especificación de los dichos servicios. Una comprensión en esta especificación requiere de dos nuevos términos para ser introducida: recurso y fábrica. Un recurso es cualquier entidad direccionadle por una referencia de punto final que pueda proporcionar su representación de XML. Una fábrica es un Servicio Web que puede crear un recurso de su representación de XML. La especificación introduce las operaciones que crean, ponen al día, recuperan, y suprimen recursos. Debe ser observado que el mantenimiento del estado para un recurso está, a lo más, conforme a los “mejores esfuerzos” del servidor de recibimiento.

Cuando un cliente recibe la aceptación del servidor de una petición de crear o de poner al día un recurso, puede razonablemente contar con que el recurso ahora exista en la localización confirmada y con la representación confirmada, pero esto no es una garantía, incluso en ausencia de terceros. El servidor pudo cambiar la representación de un recurso, pudo quitar un recurso enteramente, o pudo llegar detrás un recurso que fue suprimido. Esta carencia de garantías es constante con el modelo débilmente acoplado traído de la web. Si están deseados, los servicios pueden ofrecer las garantías adicionales que no son

requeridas por la arquitectura de los Servicios Web. La operación de recuperación es exactamente igual a la operación en el intercambio de Metadatos del Servicio Web. La petición de crear se envía a una fábrica. La fábrica después crea el recurso solicitado y determina su representación inicial. La fábrica se asume para ser distinta del recurso que es creado. El nuevo recurso se asigna una referencia servicio-determinada de punto final, que se vuelve en el mensaje de respuesta. La operación puesta pone al día un recurso proporcionando una representación del reemplazo. Una foto de una sola vez de la representación de un recurso, idéntica a la operación del conseguir en el Metadato-Intercambio del Servicio Web, se puede recuperar usando la operación de conseguir en la especificación de transferencia. Después de que una operación acertada de la cancelación, el recurso no esté más disponible con la referencia de punto final. Estas operaciones de gerencia de cuatro metadatos forman la base necesitada para construir la gerencia del estado en el Servicio Web.

En los sistemas compuestos de servicios que se comunican entre uno y otro, posiblemente usando mensajería asincrónica, muchos panoramas existen en qué información se produjo por un servicio y ésta es de interés a otro servicio. Debido a las características pobres del escalamiento, interrogación no es a menudo un mecanismo apropiado para obtener tal información; el riesgo es que demasiados mensajes innecesarios están siendo enviados a través de la red. En lugar, la arquitectura requiere un mecanismo para las notificaciones explícitas cuando ocurren los acontecimientos. Los Servicios Web mantiene ayudas de la arquitectura esto con un protocolo eventing ligero.

Los eventos en los Servicios Web especifican los mecanismos que le permiten al mismo, también designados un suscriptor, para colocar interés en los acontecimientos específicos

que son proporcionados por otro Servicio Web (la fuente del acontecimiento). Esta operación del registro se llama Subscription. Dichos eventos que definen las operaciones que un Servicio Web pueden proporcionar para las suscripciones que crean y de manejos. Las suscripciones de manejo del servicio, llamadas al encargado de la suscripción, pueden o no pueden ser el mismo servicio que la fuente del acontecimiento. Cuando el encargado de la suscripción y la fuente del acontecimiento son diferentes, todas las operaciones de la gerencia de la suscripción se dirigen al encargado de la suscripción. Cuando una fuente del acontecimiento se determina que ha ocurrido un acontecimiento, proporcionará esa información a todas las suscripciones que emparejan. Esto es similar a los asuntos que publican o los temas en un tradicional publican/suscriben el sistema de la notificación del acontecimiento. En contraste con el tradicional publicar/suscribir, los sistemas, la arquitectura que los Servicios Web proporcionan flexibilidad completa de la manera que los asuntos están definidos, organizados, y descubiertos; proporciona una infraestructura común para las suscripciones de manejo que pueden ser apalancadas en muchas diversas áreas del uso.

Un corredor de eventos se puede utilizar para agregar o para redistribuir notificaciones con diversas fuentes. Este acercamiento es apoyado por los eventos de los Servicios Web.

Los corredores pueden desempeñar varios papeles importantes en un sistema. Los asuntos se pueden organizar para uso de ciertas clases de usos. Los corredores pueden actuar como agregadores de la notificación que combinen la información del acontecimiento de fuentes múltiples. Pueden también actuar como filtros, recibiendo más mensajes que los que ellos utilizan para sus propias notificaciones. Se requiere esta flexibilidad de desplegar sistemas robustos y escalables de la notificación. Una anotación

de WSDL se utiliza para proporcionar un mecanismo bien conocido, estándar para descubrir las fuentes del acontecimiento proporcionadas por un Servicio Web. Estas anotaciones se especifican como cualidades puestas en elementos del port Type de WSDL. Identifican el Servicio Web como fuente del acontecimiento e identifican qué mensajes son notificaciones. Esta cualidad indica que las notificaciones y las operaciones de la solicitar-respuesta del port Type son acontecimientos expuestos por un servicio con un puerto limitado a este port Type. Se espera que los suscriptores del acontecimiento de la Solicitar-respuesta envíen respuestas a los avisos del acontecimiento que reciben. Una política se utiliza para especificar los modos de la entrega y los tipos apoyados del filtro.

Administración

Las Estructuras de la Administración de los Servicios Web se dividen en varios componentes de la arquitectura, proporcionando un sistema común de las operaciones que son requeridas por todas las soluciones de la gerencia de sistemas. Esto incluye la capacidad de descubrir la presencia de los recursos y de la navegación de la gerencia entre ellos. Los recursos individuales de la gerencia, tales como ajustes y valores dinámicos, se pueden recuperar, sistema, creación, y supresión. El contenido de envases y de colecciones, tales como tablas y registros grandes, pueden ser enumerados. Finalmente, se definen las suscripciones del acontecimiento y las operaciones específicas de la gerencia. En cada uno de estas áreas, la administración define solamente requisitos mínimos de la puesta en práctica.

Se ha tomado el cuidado para poder desplegar puestas en práctica conforme de la administración de los Servicios Web a los dispositivos pequeños. Al mismo tiempo, la

especificación se ha diseñado para escalar hasta datacenter grandes e instalaciones distribuidas. Además, los mecanismos son independientemente definidos de cualquier modelo implicado de los datos o de modelos de la salud del sistema. Esta independencia apoya el uso de la especificación a todas las clases de los Servicios Web.

La administración de Servicios Web requiere que los recursos manejados estén referidos usando referencias de punto final con la información adicional específica. Esta información incluye el URL del agente que proporciona el acceso al recurso, el URI único del identificador del tipo del recurso que el recurso pertenece a cero o más llave que identifique el recurso. Estas llaves se asumen para ser pares conocidos/valor. El "mapping" de esta información a una referencia de direccionamiento de Servicios Web de punto final es como sigue: el URL del recurso tras a la propiedad de la dirección, el tipo identificador del recurso de trazado a una referencia específica característica nombrado Resource URI (en el namespace apropiado de XML), y cada llave traza a un parámetro de la referencia nombrado Key con una cualidad llamada Name.

Para acomodar las necesidades de la mensajería de los servicios de la gerencia, tres calificadores se definen para las operaciones. La representación del SOAP para estos calificadores está en elementos del encabezado. Un "descanso de la operación" especifica un plazo después de lo cual la operación no necesita ser mantenida. Se utiliza cuando las traducciones de la información subyacente son necesarias o espera el elemento del "locale". Finalmente, un calificador de la "freshness" se proporciona para solicitar valores actualizados y para prohibirlos el volver de datos añejos.

Para tener acceso a los datos usando las operaciones de manejo de transferencias de los Servicios Web, la administración específica tres calificadores más. La operación de conseguir se puede calificar con el encabezado de Summary Permitted y el encabezado de No Caché. El calificador de Summary Permitted permite la transmisión de representaciones abreviadas, cuando está disponible. El calificador de No Caché requiere la transmisión de datos frescos, rechazando depositar de la información. Para haber puesto y crear las operaciones, el calificador de Return Resource asigna que por mandato la vuelta del servicio la nueva representación de un recurso en la respuesta del mensaje. Return Resource permite que los recursos obligados del Servicio Web eviten conservar la información del estado al poner al día un recurso.

La administración de los Servicios Web define tres modos de encargo de la entrega para la notificación del acontecimiento: por lote, tirar, y atrapar. Cada uno de estos modos es identificado por un URI. Este se utiliza URIs al establecer suscripciones. El modo por lote de la entrega permite a un suscriptor recibir los mensajes múltiples del acontecimiento liados en solos mensajes SOAP. El suscriptor puede también solicitar un número máximo de los acontecimientos incluidos en el paquete, una cantidad de tiempo máxima que el servicio deba tomar al acumular acontecimientos, y una cantidad máxima de datos que deban ser devueltos. El modo de la entrega del tirón permite que el servicio de dato que se produce mantenga una cola lógica de acontecimientos de modo que el suscriptor pueda votar para las notificaciones a pedido. Se hace esta interrogación usando Enumeración de Servicios Web con el contexto de la enumeración vuelto con el mensaje de respuesta de la suscripción. Finalmente, cuando el multicast del UDP, el cual es un mecanismo apropiado de la mensajería, el modo de la entrega de la trampa permite que una fuente del acontecimiento la utilice. En modo de la trampa, la fuente del

acontecimiento puede enviar sus notificaciones a una dirección predeterminada del multicast del UDP.

CAPITULO III

Seguridad

Para diseñar, desarrollar e implementar servicios web seguros, arquitectos y desarrolladores deben aprender nuevas tecnologías y considerar nuevas amenazas asociadas con la exposición de su funcionalidad en potencialmente inseguras redes. Estos arquitectos y diseñadores responsables por la seguridad de los servicios web tienen un gran número de opciones disponibles para utilizar. Pero estas opciones son mucho más complicada por el hecho que diferentes proyectos y diferentes organizaciones tienen diferentes requerimientos de seguridad. Veremos los aspectos generales necesarios para hacer seguro un servicio web en cualquier lenguaje.

Hay un conjunto de tecnologías de seguridad que se presentan en varias ocasiones en diferentes corporaciones cuando identifican sus requerimientos de seguridad. Hay un número de servicios encargados de la seguridad que normal mente se utilizan para cumplir con estos requerimientos.

Criptografía

La primera tecnología de seguridad importante que necesitarás para asegurar un servicio Web es la manera de proteger los datos sensibles como viajan en las redes abiertas. La criptografía puede utilizarse para cifrar mensajes para protegerlos contra el acceso

indebido; es decir, para evitar que alguien lea los datos del mensaje cuando pasa cerca en el alambre. De la criptografía se puede también asegurar la integridad de los mensajes; es decir, evitar que alguien los modifique, suprimiendo, o insertando pedacitos en los datos del mensaje sin que este sea detectado por el recipiente legítimo del mensaje. La mayoría de la gente piensa en criptografía como manera de hacer la información ilegible, o cifrado, y, más adelante, de invertir la operación de modo que la información sea otra vez comprensible, o descifrado. Se utiliza como manera de proteger la información, generalmente cuando la comunicación es a partir de un punto a otro. Las llaves criptográficas, que no son más nada que números al azar muy grandes, controlado por un proceso.

En la criptografía tradicional, la misma llave criptográfica se utiliza para cifrar y para descifrar información. Esto se conoce como llave secreta porque los dos los partidos que desean comunicarse con seguridad utilizan la misma llave para cifrar y para descifrar mensajes. Ambos consiguen sus llaves con medios seguros y deben proteger sus llaves para cerciorarse de que solamente los individuos autorizados pueden utilizar la información. El no perder de vista todas estas llaves es difícil. También, desde que ambos utilizan la misma llave, un lado de la comunicación no puede ser distinguida de la otra, así que no es posible probar quién originó un mensaje. Los algoritmos dominantes secretos comunes son el estándar de cifrado de datos (DES) (NIST 1988) Triple DES (3DES), y Advanced Encryption Standard (AES) (NIST 2001), estandarizado por el National Institute of Standards and Technology de los E.E.U.U. (NIST).

Otro acercamiento de la criptografía se llama llave pública o criptografía asimétrica. Esta forma de criptografía utiliza dos llaves diferentes pero matemáticamente relacionadas.

Una llave no se puede utilizar para descubrir la otra. Con la criptografía dominante pública, la llave pública se puede hacer público a cualquier persona que desea conducir transacciones con el sostenedor de la llave privada. La distribución de la llave pública es fácil. La llave privada se debe mantener privada y llevar a cabo solamente por su dueño. Un algoritmo dominante público popular es RSA, inventado por Ron Rivest, Adi Shamir, y Leonard Adleman. Cuando la criptografía dominante pública se utiliza para el cifrado, la llave pública se utiliza a cifrar los datos y la llave privada se utiliza para descifrar datos. Los sostenedores de las llaves públicas, usa la llave pública para cifrar los datos significativos para ir al dueño de la llave privada, pero solamente el dueño de la llave privada puede descifrar la información.

Autenticación

La criptografía es una herramienta necesaria para la protección de datos pues atraviesa redes excesivas, pero para la mayoría de los servicios web la criptografía por sí mismo es inadecuada. Para mantener a servicio web seguro, también necesitas saber las identidades de los actores que están estableciendo una conexión con servicio web. La autentificación proporciona tal servicio, y puede darte la confianza que el solicitante de un servicio web es quién él o ella dice ser y no un impostor.

Los mecanismos de autentificación caen en dos categorías básicas: contraseña y respuesta a la petición. Aunque la autentificación contraseña-basada es popular, tiene algunas limitaciones inherentes. La autentificación basada respuesta a la petición puede ser más compleja instalar, pero proporciona un muy perceptiblemente alto nivel de seguridad.

La clase más simple de autentificación orientada a la conexión utiliza un secreto compartido bajo la forma de la contraseña, número de identificación personal (PIN). La característica más significativa de sistemas basados en contraseña es que la autentificación no depende de la información enviada por el lado que realiza la verificación de la autentificación. Todos estamos familiarizados con los sistemas basados en contraseña. Las conexiones del sistema operativo son basadas en contraseña. La autentificación básica del HTTP es otro ejemplo de tal sistema. Si las peticiones de usuario a un recurso protegido y no proporcionan la información de la autentificación, el servidor web rechazan la petición. Este rechazamiento hace el navegador del usuario solicitar la identificación y la contraseña de usuario. El navegador reedita la petición con el user-id y la contraseña. Entonces, si el user-id y la contraseña son aceptables, el servidor web entrega el contenido solicitado. Una desventaja de este sistema es que la contraseña está enviada sobre la red y potencialmente se expone. La criptografía se debe utilizar para proteger la contraseña cuando atraviesa la red abierta. La autentificación más común del cliente para aplicaciones web establece una sesión cifrada usando el SSL, y entonces utiliza una contraseña convencional protegida por el cifrado del SSL para autenticar al usuario.

En sistemas basados en respuesta a la petición, el lado que realiza la verificación de la autentificación primero envió un desafío. El sistema del cliente que intenta probar la identidad del usuario realiza una cierta función en el desafío basado en la información solamente disponible para el usuario/el cliente y devuelve el resultado. Si el resultado está según lo esperado, autentican al usuario. La característica notable de estos sistemas es que la respuesta depende del desafío. La autentificación del resumen del HTTP es un ejemplo de un sistema basados en respuesta a la petición. En este acercamiento, el cliente envía una respuesta al servidor que se deriva de un valor al azar proporcionado por el

servidor así como la contraseña. Estos sistemas son más seguros que sistemas de contraseña porque no exponen la información de autenticidad sobre la red. La información de autenticidad proporcionada originalmente por el usuario nunca sale del sitio de trabajo del cliente. Otra ventaja es que desde el desafío, que es elegido por el lado de autenticidad, varía con cada autentificación, la respuesta variará también, de este modo eliminan los ataques de respuesta. Estos no pueden apoyar generalmente la personificación del lado de autenticado por la aplicación del servidor, que este sistema mucho más seguro pero menos flexible que el de contraseñas. Los sistemas de la Desafiar la respuesta realizan la autentificación sin enviar una contraseña sobre una red. Sin embargo, este acercamiento no elimina contraseñas. El abrir la información de la autentificación almacenada en el cliente requiere generalmente una contraseña. En contraste con la autentificación de la contraseña, sin embargo, ni la contraseña ni la información de la autentificación siempre del sitio de trabajo del usuario en un sistema de desafiar la respuesta.

Protocolos de uso Criptográfico

La mayor parte de los sistemas populares de autentificación basados en desafiar la respuesta son el implementar el uso de protocolos criptográficos. La estructura de estos protocolos está basada sobre los mecanismos, llamada criptografía de llave secreta y de público. Los protocolos definen cómo los clientes y los servidores deben intercambiar la información dominante para establecer una sesión autenticada segura. Un cliente y un servidor entonces utilizan la sesión establecida para intercambiar los datos que se protegen contra el acceso y/o la modificación. Los protocolos criptográficos más comunes de autentificación son SSL y Kerberos.

Algoritmos de llaves públicas, el actor principal mantiene su llave privada. Por esa razón, la firma de un actor principal en un mensaje que usa la llave privada del actor constituye una prueba de la identidad. Cuando la tecnología de llaves públicas es utilizada para autenticar el actor principal a un servidor web, él envía al servidor su certificado de llave pública después de que el servidor de autentificación haya establecido la llave de la sesión. El actor también envía su firma en una combinación de servidor y información proporcionada por el cliente. El servidor verifica la firma en el certificado de llave pública del actor y verifica la firma del cliente en los datos combinados. Si se verifica la firma, autentican al cliente, y la sesión cifrada puede comenzar a usar una llave secreta compartida de la sesión. El uso más amplio de un esquema de protocolo de llave pública de autentificación para la Web tener acceso es el protocolo seguro de la capa de sockets (SSL). El SSL ahora oficialmente se llama transporte Seguridad de capa (TLS) (IETF 1999). El SSL es un protocolo de seguridad del transporte colocado sobre el TCP pero debajo de la capa de aplicación de una pila de protocolos de comunicación. Fue desarrollado originalmente por Netscape para proporcionar la seguridad en las transacciones web del HTTP. Además de HTTP, el SSL puede también estar proporcionaban el transporte seguro para el telnet, el ftp, y otros protocolos de aplicación. El SSL generalmente es usado reforzar la confiablidad e integridad para los usuarios finales que tienen acceso a la web, y debe ser basado en el estándar de cifrado de triple Data Encryption Standard (3DES) or RC4 llaves de sesión. Para la protección adecuada, las llaves públicas del servidor web deben ser por lo menos 1.024 bits.

En algoritmos de llaves privadas, la identidad de un actor principal es verificada probando si él puede cifrar correctamente un mensaje usando una llave que se comparte solamente entre el verificador y el actor. Así, el verificador debe también poseer la llave para realizar la verificación. A diferencia de protocolos de llaves públicos, los protocolos de llaves

privadas pueden ser difíciles de escalar a las aplicaciones grandes porque cada actor debe tener una llave secreta diferente para cada otro autor que quisiera autenticar. Para ocuparse del problema en parejas de las llaves para todas las aplicaciones, las versiones prácticas de los protocolos de llaves secretas de autentificación mantienen una política de confianza sobre un tercer ente que mantienen las llaves para una colección de actores. Todos los actores en esa colección tienen que confiar solamente en los terceros para proteger sus llaves secretas.

El representante más popular de los protocolos de llaves secretos de autentificación es el Kerberos (IETF 1993, Neumann 1994), que fue desarrollado en el MIT. Después de que un cliente y un servidor han utilizado Kerberos para probar su identidad, pueden también cifrar todas sus comunicaciones para asegurar confiablidad e integridad de los datos. Kerberos se ha incorporado en el Distributed Computing Environment (el DCE) (Gittler 1995, OSF 1996) y adoptado con extensiones por Microsoft para los ambientes del Windows 2000. Kerberos generalmente se usa en la capa intermedia dentro de las redes corporativas. Kerberos permite a un actor pruebe su identidad a un servidor sin enviar los datos de la autentificación (tales como una contraseña) que pueden permitir que un atacante personifique posteriormente al actor. La aplicación del cliente proporciona una llave secreta que se derive de una contraseña como la base de la autentificación. La llave secreta se puede potencialmente almacenar en un símbolo del hardware (tarjeta del DES) para una autentificación más fuerte y puede también derivar de un certificado de llave pública. Para utilizar un servicio de seguridad de Kerberos, el cliente primero envía la identidad del actor al servidor de autentificación, que envía detrás una credencial llamada un boleto aprobador (TGT – Ticket Granting Ticket). Se ha cifrado El TGT de modo que solamente el actor legítimo que posee la contraseña correcta pueda descifrarla y utilizarla en un futuro. Cuando el cliente desea tener acceso a un servidor de aplicación usando

Kerberos, el cliente envía el TGT al centro de distribución de llaves (KDC). El KDC vuelve un ticket de sesión, que contiene un identificador secreto de la llave y del cliente de la sesión. El cliente utiliza el ticket de la sesión para establecer una sesión autenticada y segura del protocolo TCP/IP con el servidor de aplicación. El ticket de la sesión es protegido por el cifrado y no es expuesto sobre la red. Kerberos proporciona opcionalmente los servicios de cifrado y de integridad para los mensajes intercambiados entre los clientes y las aplicaciones de servidor. Kerberos utiliza DES para el cifrado y RSA MD4/MD5 para la integridad. El Kerberos es capaz de soportar llaves de 128 bits, cuál es la longitud dominante recomendada actual para la mayoría de las aplicaciones. DCE amplía la funcionalidad de Kerberos, y se ha utilizado extensivamente en redes corporativas en las cuales su sistema rico de características que hacen que sea un sistema de alto rendimiento en el manejo de llaves secretas de autenticación. La seguridad del DCE 1.1 es un producto maduro que proporciona una poderosa ayuda y flexible para todos los aspectos de la seguridad: conexión, autentificación, protección del mensaje, autorización, delegación, intervención, y manejo de llaves. DCE comienza con el Kerberos V5, que proporciona la autentificación de llaves secretas básica (DES) y protección del mensaje. El DCE entonces agrega los servidores de registro y los servidores de privilegio para proporcionar servicios adicionales. Se saben y se prueban la autentificación de llave secreta usando Kerberos y la seguridad de tecnologías DCE con buen funcionamiento en redes corporativas. Sin embargo, está generalmente aceptado que la distribución y manejo de llaves secretas no es manejable para una gran cantidad de usuarios. Incluso con el diseño basado Kerberos del Passport de Microsoft, el protocolo todavía recibe la crítica para que su inhabilidad maneje la escala y la naturaleza libremente federada de los servicios Internet.

Sistemas de Autenticación

Hay diferentes sistemas de autenticación para los servicios web que usan los diferentes mecanismos que hemos mencionado.

- **Sistemas Operativos basados en autenticación:** Los servicios web son usualmente requeridos y entregados vía HTTP. Sin embargo, estos sistemas generalmente tienen servidores web como sistema de soporte. Algunos servidores web, realizan la autenticación usando las facilidades del sistema operativo. De esta manera el Internet Information Server (IIS) de Microsoft realiza la autenticación. El IIS ofrece una variedad de métodos de autenticación. Esto incluye nombre de usuario, contraseña con la autenticación básica de HTTP, NT LAN Manager (NTLM) y encabezados HTTP, ambos son basados en el método de desafiar la respuesta, autenticación SSL y Kerberos. Como resultado de una finalización de la autenticación satisfactoria por parte del IIS, el usuario es conocido por el sistema operativo y las facilidades de este puede ser usadas para la autenticación. Esta es una manera muy poderosa de hacer autorización, pero no los implementadores de sistema quieren establecer las cuentas del sistema operativo Windows para los usuarios Web.
- **Servidores Web basados en autenticación:** Servidores web generalmente vienen con la capacidad de construir un sistema capaz de realizar autenticación manejando los diferentes requerimientos de autenticación para HTTP. Dependiendo del servidor Web, la información usada para la autenticación puede ser la misma usada por el sistema operativo, guardada en un archivo contenedor separado pero en la misma plataforma que la del servidor web, o guardado en separado pero en un repositorio de usuario (Lightweight Directory Access Protocol [LDAP]). La clave para este tipo de autenticación contra la basada en un sistema

operativo es que el usuario que haya logrado autenticar no será conocido para el sistema operativo.

- **Autenticación Basada en *Tokens*:** Con estos sistemas, el usuario de poseer un token físico que juegue alguna parte del proceso de autenticación, lo cual lo hace mucho más fuerte que las mismas contraseñas. Tokens frecuentemente son usados para accesos remotos a sistemas con privilegios, dado que ellos proveen dos factores para la autenticación (posesión física de la tarjeta del token y conocimiento de un PIN). Los Tokens, sin embargo, son mucho más costosos y complejos de implementar que los de Identificadores y contraseñas. Algunas veces los valores de los token deben ser verificados por un servidor de autenticación. Los token también podrían tener una conexión física a la estación de trabajo, de tal manera que el desafía sea ingresado automáticamente dentro del token y la respuesta sea automáticamente envidada desde el mismo hacia la estación de trabajo durante el protocolo de autenticación.
- **Una sola autenticación Web:** Se necesita la autenticación tanto para las transacciones HTTP como para cualquier otro tipo de transacción electrónica. La dificultad con HTTP es el no tener estado y la inhabilidad de seguir la pista a la sesión de un usuario. Cada petición a un servidor web es tratada como una nueva petición y el usuario debe, teóricamente, se autenticado otra vez. Una solución a este problema es tener la información de la autenticación del navegador en caché y presentarla con cada petición al servidor. Esto evita al usuario a tener que reingresar la información múltiples veces. Sin embargo, esto crea el problema de una contraseña plana que atraviesa el Internet y crea el problema de una contraseña clara del texto que atraviesa el Internet con cada acceso al servidor web. Otros problemas, tales como usando la misma autenticación con diferentes servidores web en el mismo clúster del servidor, aún requeriría ser autenticado de vuelta. En respuesta a eso se creó este tipo de autenticación con sistemas como

Netegrity SiteMinder, Microsoft Passport, Entrust getAccess, y RSA ClearTrust. Los sistemas que manejan este tipo de autenticación mantienen una sesión entre peticiones HTTP al mismo servidor o clúster. El usuario ingresa una vez y puede tener acceso a cualquier otro servidor web en el clúster sin necesidad de ingresar otra vez mientras dure la sesión. Si la verificación de la autenticación pasa, el servidor de seguridad crea un cookie encriptada, la cual contiene el nombre de usuario, IP, dirección, navegador, hora, y tiempo de expiración. El servidor web devuelve la cookie al navegador del cliente, el cual la usa para las siguientes peticiones HTTP autenticadas.

- **Biométrica:** Cuando una empresa necesita una fuerte evidencia de la identidad del usuario, ella frecuentemente recurre a este método. La biométrica incluye mecanismos tales como escáner de retina, reconocimiento de voz, y lectores de palma y huellas digitales. Esto provee una fuerte autenticación, pero puede ser costosa y necesita ser evaluada con respecto a las negativas como la aceptación social. Biométrica son frecuentemente usada en edificios controlados o aplicaciones críticas con usuarios limitados. Cuando ha sido usada con servidores de autenticación web, biométrica generalmente requiera una integración personalizada.

Autorización

La criptografía y la autentificación se requieren para virtualmente cualesquiera asegurar cualquiera aplicación de servicio Web, no importa que tan simple los requisitos de la seguridad puedan ser. La última tecnología de seguridad que es generalmente necesaria es la autorización. La autorización concede el permiso a los actores accesar a los recursos de los servicios del Web, proporcionando la base para el control de acceso. La

autentificación se realiza principalmente para apoyar la autorización. La razón primaria para que un servidor autentique una identidad es para que el servidor pueda tomar una decisión, basada en esa identidad, para conceder el acceso al recurso del servicio Web solicitado.

Las políticas de autorización restringen el acceso a diversas colecciones de recursos: anfitriones, archivos, páginas web, interfaces de las aplicaciones, los métodos, instancias, y registros de la base de datos, para nombrar algunos. Las políticas de autorización pueden también restringir el acceso en un contexto global (tal como tiempo del día), contexto transaccional (tal como no más de tres retiros por día), o valores de los datos (tales como negociar no más de un millón partes de acción). Este tipo de políticas de *grano fino* de la autorización pueden ser muy complejas de definir y manejar, y están ciertamente bien más allá de las necesidades básicas de la seguridad de una típica aplicación.

Seguridad en tecnologías de capa intermedia

Las tecnologías de mitad de capa son el software de los sistemas de la computación moderna empresaria, el cual procesa las peticiones provenientes a través de las puertas de enlace de los servicios web.

Si examinas una selección de las tecnologías de capa intermedia, encontrarás que hay muchas semejanzas, tales como grupos, roles, y otras cualidades del usuario, así como funciones o características únicas, tales como derechos requeridos de CORBA. Otras

funciones o características, tales como cualidades del método-permiso de EJB y atributos de permiso del cliente de .NET, pueden parecer diferentes pero son términos realmente diferentes con significados similares. Hay también términos que parecen equivocadamente similares, por ejemplo el actor en JAAS y DCOM+, pero que tienen muy diferente semántica. Esta sección comienza definiendo los elementos y los principios del la seguridad de capa intermedia en términos estándares para las tecnología.

Paradigma Cliente – Servidor

La característica dominante de la mayoría de los sistemas distribuidos comerciales es hoy el paradigma cliente/el servidor, que es la base para el modelo del Remote Procedure Call (RPC) que permite un programa del cliente a enviar una petición de llamada de un procedimiento a un servidor. En cada interacción, el cliente inicia la petición, y un servidor recibe y procesa la petición. Algunos modelos del RPC apoyan la invocación “fire-and-forget”, donde el cliente envía una petición pero no espera ninguna respuesta a partir de esta. Un ejemplo en funciones unidireccionales es CORBA, por que el cliente no cuenta con ninguna respuesta del servidor e incluso no está garantizado que su petición será procesada en todos. Éste es también el caso para el mundo de los servicios Web basados en SOAP, donde, si un método no devuelve cualquier cosa, no se envía ningún mensaje de respuesta al cliente.

Seguridad en el Paradigma Programación Orientada a Objetos

CORBA, COM+, .NET, y J2EE son todos orientados a objeto. Actualmente, el mundo de la computación da por sentado que cualquier tecnología computacional moderna, distribuida o no, tiene soporte inherente de los objetos.

Cuando se agrega objetos a un modelo cliente/ servidor, hay un significativo efectos sobre la infraestructura total de la seguridad de la empresa. Los objetos tienden a ser de granularidad fino, es decir, encapsulan cantidades pequeñas de datos y proporcionan diverso métodos para manipular esos datos. Los sistemas basados en objetos tienen generalmente muchos más objetos de diversas variedades, aumentando el número de recursos en tus sistemas que necesitas proteger, comparado a los sistemas convencionales, procesales. La cantidad de recursos necesarios para una operación también se incrementa súbitamente.

Una arquitectura que basa su seguridad orientada a objetos debe apoyar una gran cantidad recursos protegidos. Tradicionalmente, esto se ha hecho vía a las agrupaciones del recurso. Los objetos son agrupados, y las políticas se definen en esos grupos. Objetos con nombres similares, o los que residen en la misma localización, no se deben requerir para pertenecer igual grupo, puesto que las políticas no siguen necesariamente la topología o el nombramiento de la política de tu aplicación. Igual es verdad para los objetos que se asignarán al mismo grupo; semejanza conocida y la co-localización no se debe requerir para ser gobernada por políticas similares. Hay también objetos transitorios y de breve duración.

Tales objetos son probables a ser sin nombre y creado sin el control del administrador. Manualmente asignar el transeúnte los objetos a las políticas de la seguridad sería poco realista. Además, los sistemas basados en objetos tienden para tener jerarquías de nombramiento menos rígidas, y el nombramiento del mecanismo puede permitir que un objeto tenga más de un nombre. También una preocupación es la identidad del objeto en algunos sistemas, en los cuales dos referencias opacas del objeto pueden estar reutilizado por el sistema de capa intermedia, haciéndola difícil de determinarse si los objetos ellos señalan a ser diferente o igual. Cualquier arquitectura de la seguridad del middleware está en lugar, debe permitir que los administradores de la seguridad definan políticas de la seguridad sin asumir que ellas saben el nombre de cada objeto en el sistema. Aunque un objeto tiene más de un nombre, la misma política se debe aplicar a ella no importa qué se utiliza el nombre.

De seguridad adicional la preocupación es que los métodos en objetos están limitados no más a apenas dos o tres de la "cancelación" del universal, "leído" y "escribir operaciones".

Los métodos podrían tener la misma complejidad y potencialmente implicar muchas actividades diversas. Por lo tanto, los administradores de la seguridad no deben tener que entender la semántica de los métodos en los objetos para asegurarlos. Debido a la encapsulación y a otras técnicas tales como atascamiento dinámico y la reencarnación a pedido de las puestas en práctica que del objeto. Hace los sistemas distribuidos grandes escalables y más simples de diseñar, es difícil entender qué real es la manipulación de recursos detrás de la interfaz del objeto. Mientras que el trabajo de los reveladores del cliente es simplificado usando objetos, la encapsulación hace difícil de determinar qué

políticas de la seguridad deben gobernar el acceso a el cual la aplicación se opone. La seguridad de la arquitectura de sistemas basados en objetos debe también tener algunos mecanismos en lugar eso ayuda con este problema.

CAPITULO IV

Asegurando un Servicio Web desde IIS y .Net

Una parte en la construcción soluciones basadas en Servicios Web de Microsoft de edificio, IIS juega un papel crítico en la protección de los servicios web que están residiendo en el. Los mecanismos de la seguridad que IIS proporciona se pueden clasificar en los grupos de base siguientes: autentificación, protección del mensaje, control de acceso, registración, y aislamiento de fallas

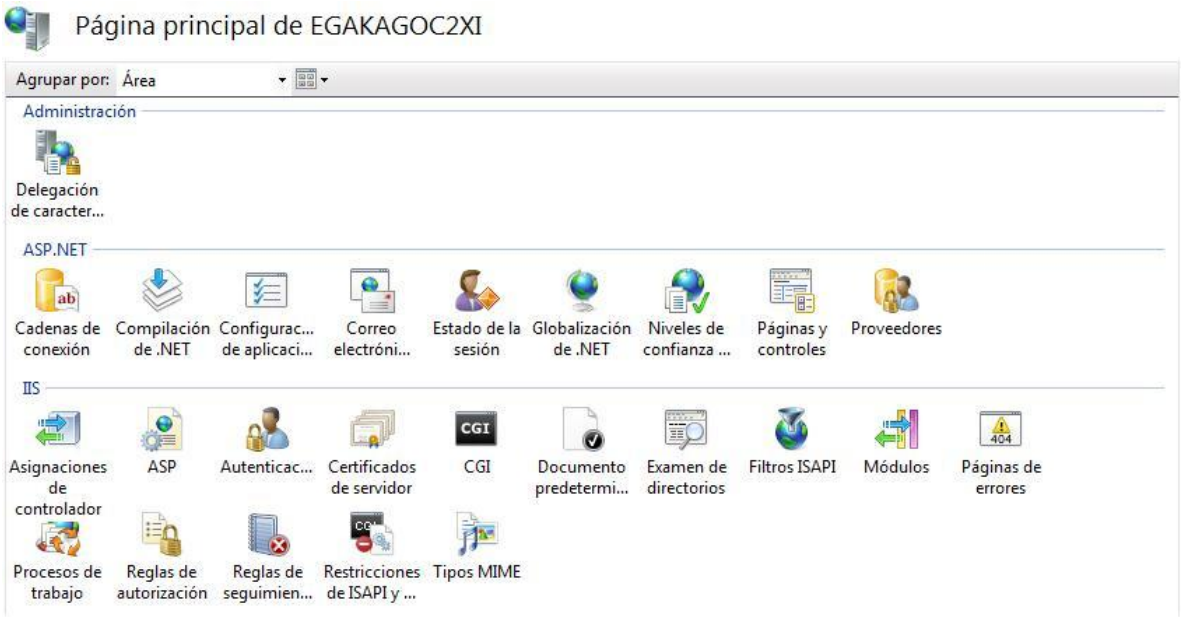


Ilustración 9: Página Inicial de Administración IIS, Windows Vista, Microsoft Corporation

Autenticación

Los mecanismos de la autentificación de IIS no son muy diferentes de los que la mayoría de los otros servidores web proporcionan. Para cada directorio, unas o más opciones de la autentificación, demostradas en la siguiente figura, pueden ser seleccionadas.

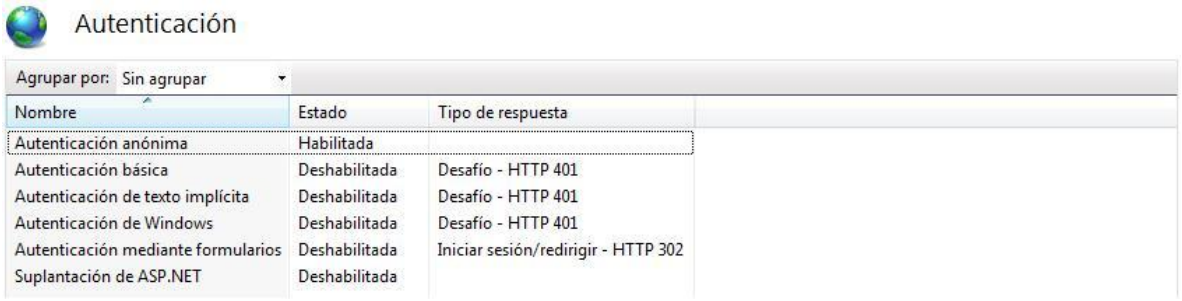


Ilustración 10: Página Inicial de Administración de Autenticación IIS, Windows Vista, Microsoft Corporation

Todas las características, excepto el método pasado en la figura, son probables ser encontradas en el servidor web de cualquier otro vendedor. Adaptado específicamente a los ambientes de Windows no separados por los cortafuegos, la autentificación integrada de Windows puede utilizar NTLM o el Kerberos V5 y solamente trabaja con el Internet Explorer 2.0 en adelante. Cuando un usuario con un IE el navegador procura tener acceso a un recurso protegido, IIS negocia con el navegador, y uno de los dos mecanismos es utilizado.

Todos estos métodos autentican peticiones entrantes de HTTP. Sin embargo, si el servidor es configurado con la ayuda de SSL/TLS, entonces las conexiones del cliente se podrían autenticar con los certificados del cliente X.509 tras en las cuentas del sistema operativo. Incluso sin los certificados del cliente, el uso de SSL/TLS es siempre una buena idea en el

caso básico o una breve autentificación, porque protege el intercambio de los datos de la autentificación, de lo contrario sería vulnerable a ataques de escucha detrás de puertas y/o ataques de respuesta.

Es importante para entender que cualquiera, inclusive anónima, petición del HTTP es dirigido por algún tratante del HTTP funcionamiento con una identidad particular reconocida por el sistema operativo. Si una delegación simple, también conocida pues la personificación, no se habilita, entonces todos los tratantes utilizan la misma identidad que el mismo IIS utiliza. Una vez que el administrador permita la personificación, los tratantes comienzan a trabajar con la identidad del usuario autenticado. Pero las peticiones anónimas no tienen ninguna identidad, como se pudiera suponer, equivocadamente.

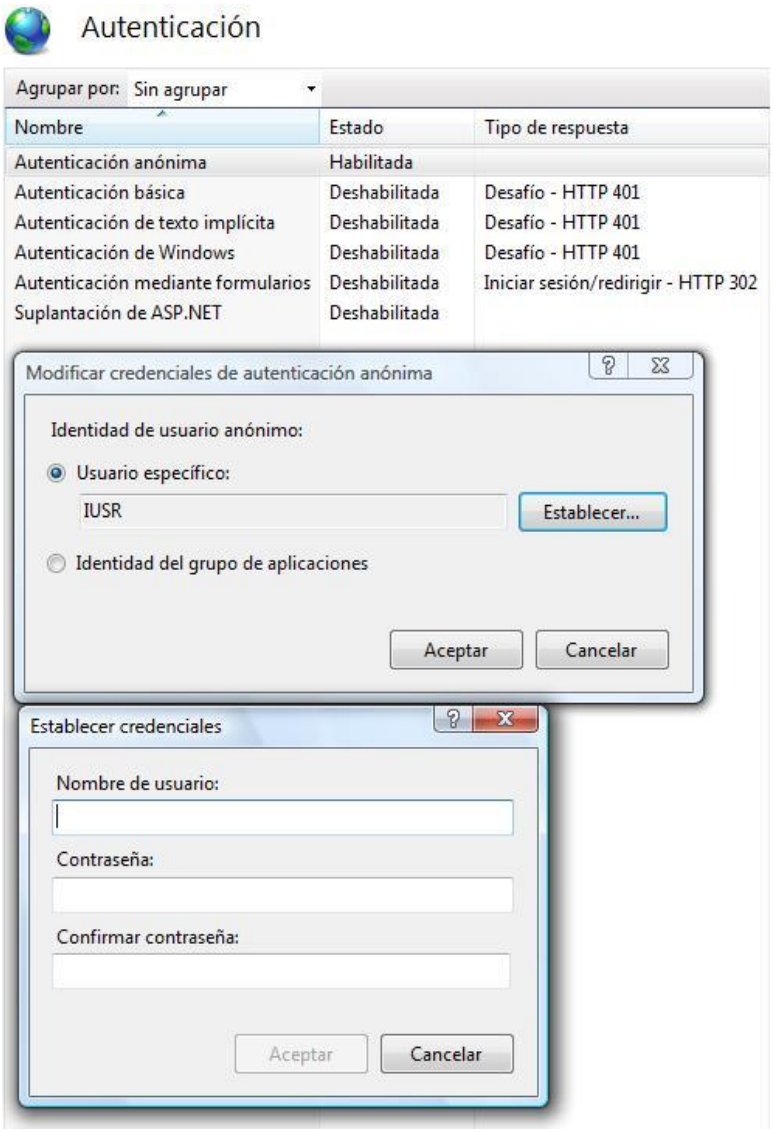


Ilustración 11: Selección de la Cuenta de Usuario Anónimo, Windows Visa, Microsoft Corporation

Mientras que el cuadro 11 ilustra, IIS proporciona las credenciales almacenadas a Windows usando una cuenta especial del usuario, IUSR, que se puede cambiar a cualquier otra cuenta. Incluso para las peticiones anónimas, los tratantes del HTTP personificaron una cuenta legítima del usuario, cuál se asigna privilegios particulares y concedió algunos

permisos. Cuando IIS controla la contraseña, Windows autentica el usuario y crea una conexión de la red que evite que el usuario tenga acceso recursos en otras máquinas. Cuando IIS no controla la contraseña, utiliza medios levemente diversos y, consecuentemente, una conexión local se crea en lugar de otro. La conexión local permite que el tratante correspondiente del HTTP personifique a un usuario anónimo mientras que los recursos de la red que tiene acceso, incluyendo llamadas a los componentes alejados de COM y de COM+. Con nativo el soporte del Kerberos multi-hop de la delegación, introducido en Windows 2000, cosas llegó a ser incluso más difícil. En el final del día, todo esto requiere tener aún más cuidado mientras que diseña usos distribuidos con Microsoft tecnologías.

Protegiendo los datos

La autenticidad, confidencialidad y la protección de la integridad de los mensajes en el tránsito entre los clientes del HTTP e IIS se pueden poner en ejecución usando cualquier SSL/TLS, configurando Servidor web, o IPSEC (IETF 2002, Microsoft 1999), si esta tecnología es parte de la infraestructura de la compañía. Puesto que el SSL es una tecnología madura que cada Web máster está al corriente de su actualidad. Hay que tener presente que ambos acercamientos dan, lo, puede o no puede ser suficiente, dependiendo de la aplicación de los servicios web.

Control de Acceso

Todos estos esfuerzos por IIS – para usar cuentas de sistema para las peticiones anónimas, clientes del mapa autenticados con los certificados X.509 en cuentas de usuario, y la personificación de la ayuda por el HTTP tratante - es principalmente por los mecanismos nativos del control de acceso del sistema operativo para proteger recursos del IIS. Cuando un tratante del HTTP tiene acceso a un HTML o a cualquiera el otro archivo para procesar la petición correspondiente del HTTP, permisos del sistema de ficheros de Windows bajo la forma de Access Control List discrecional (DACL) es utilizado por el sistema operativo para hacer cumplir políticas del control de acceso. Tener presente que este mecanismo está solamente disponible en sistemas de ficheros de NTFS. Si un archivo que se alcanzará reside en una partición del FAT, no se hace ningún chequeo del acceso.

Dos otros mecanismos de control de permisos – permisos web y las restricciones basadas en IP se podían utilizar además de controles de DACL, sí los hay. Los permisos Web te permiten para fijar políticas de grano grueso de la autorización por el sitio Web, directorio virtual, directorio, o archivo. Puesto que la identidad del usuario no es considerado en los permisos Web, la misma política se aplica a todas las peticiones, que podrían ser útiles cuando necesitas fijar un campo común bajo filigrana para un rama de los recursos de IIS. Los permisos Web permiten modificar de las opciones siguientes que son importantes para los servicios web.

- Lectura – Los datos pueden ser leídos o descargados.
- Escritura – Los datos pueden ser escritos o los archives pueden ser subidos usando protocolos HTTP PUT.

- Exploración de Directorios – Los clientes pueden ver la lista de directorios y archivos.
- Permisos de Ejecución:
    - Ninguno —Archivos ASP (.ASP) y archives ejecutables (.BAT, .EXE, y .DLL) no se ejecutarán.
    - Solo Scripts— Archivos ASP, pero los archives ejecutables no se ejecutaran.
    - Scripts y Ejecutables—ambos tipos de archives se ejecutaran.

En IIS 7.0 cambia la manera de cómo configurar los niveles de confianza y solo se limita a: Total, Alto, Medio, Bajo, Mínimo. Como se muestra en **Figura 12**, Luego toca especificar el tipo de aplicaciones a las cuales se les permite ejecutarse y de la manera en que se ejecutan, como se ilustra en la siguientes graficas.

Niveles de confianza de .NET (Página)

Utilice la página de características **Niveles de confianza de .NET** para establecer el elemento de confianza en el archivo Web.config. El elemento de confianza permite configurar el nivel de seguridad de acceso a código (CAS) que se otorga a una aplicación.

Lista de elementos de la interfaz de usuario

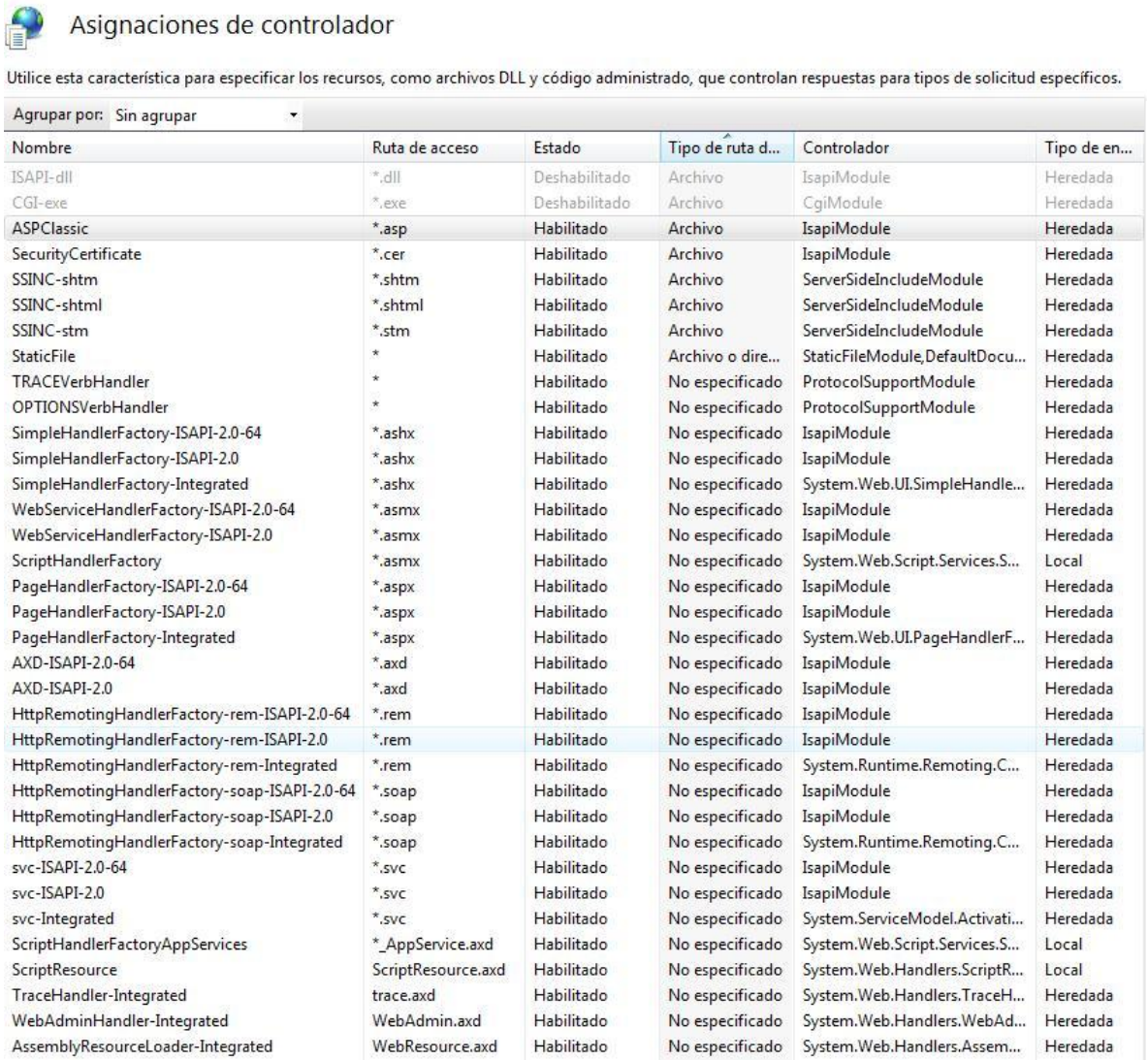
Nombre de elemento	Descripción

Nivel de confianza	Muestra el nombre del nivel de confianza. Entre las opciones disponibles se incluyen:
	• **Total (interno)**: Especifica permisos no restringidos. Concede a la aplicación ASP.NET permisos de acceso a cualquier recurso sujeto a la seguridad del sistema operativo. Se admiten todas las operaciones privilegiadas.
	• **Alto (web_hightrust_config)**: Especifica un nivel alto de seguridad de acceso a código, lo que significa que la aplicación no puede realizar de forma predeterminada ninguna de las acciones siguientes:
	  • Llamar a código no administrado.
	  • Llamar a componentes a los que se presta servicio.
	  • Escribir en el registro de eventos.
	  • Obtener acceso a colas de servicio de Message Queue Server (MSMQ).
	  • Obtener acceso a orígenes de datos de ODBC, OLEDB u Oracle.
	• **Medio (web_mediumtrust_config)**: Especifica un nivel intermedio de seguridad de acceso a código, lo que significa que, además de las restricciones del nivel de confianza Alto, la aplicación ASP.NET no puede realizar de forma predeterminada ninguna de las acciones siguientes:
	  • Obtener acceso a archivos que están fuera del directorio de la aplicación.
	  • Obtener acceso al Registro.
	  • Realizar llamadas a servicios web o de red.
	• **Bajo (web_lowtrust_config)**: Especifica un nivel bajo de seguridad de acceso a código, lo que significa que, además de las restricciones del nivel de confianza Medio, la aplicación no puede realizar de forma predeterminada ninguna de las acciones siguientes:
	  • Escribir en el sistema de archivos.
	  • Llamar al método Assert.
	• **Mínimo (web_minimaltrust_config)**: Especifica un nivel mínimo de seguridad de acceso a código, lo que significa que la aplicación sólo tiene permisos de ejecución.

Información relacionada

Para obtener información de procedimiento sobre cómo se configuran los niveles de confianza de .NET, vea Configuring .NET Trust Levels in IIS 7.0 (en inglés) en el documento en línea IIS 7.0 Operations Guide disponible en Microsoft TechCenter.

Ilustración 12: Niveles de Confianza - IIS 7.0 Help, Windows Vista, Microsoft Corporation



Asignaciones de controlador

Utilice esta característica para especificar los recursos, como archivos DLL y código administrado, que controlan respuestas para tipos de solicitud específicos.

Agrupar por: Sin agrupar

Nombre	Ruta de acceso	Estado	Tipo de ruta d...	Controlador	Tipo de en...
ISAPI-dll	*.dll	Deshabilitado	Archivo	IsapiModule	Heredada
CGI-exe	*.exe	Deshabilitado	Archivo	CgiModule	Heredada
ASPClassic	*.asp	Habilitado	Archivo	IsapiModule	Heredada
SecurityCertificate	*.cer	Habilitado	Archivo	IsapiModule	Heredada
SSINC-shtm	*.shtm	Habilitado	Archivo	ServerSideIncludeModule	Heredada
SSINC-shtml	*.shtml	Habilitado	Archivo	ServerSideIncludeModule	Heredada
SSINC-stm	*.stm	Habilitado	Archivo	ServerSideIncludeModule	Heredada
StaticFile	*	Habilitado	Archivo o dire...	StaticFileModule,DefaultDocu...	Heredada
TRACEVerbHandler	*	Habilitado	No especificado	ProtocolSupportModule	Heredada
OPTIONSVerbHandler	*	Habilitado	No especificado	ProtocolSupportModule	Heredada
SimpleHandlerFactory-ISAPI-2.0-64	*.ashx	Habilitado	No especificado	IsapiModule	Heredada
SimpleHandlerFactory-ISAPI-2.0	*.ashx	Habilitado	No especificado	IsapiModule	Heredada
SimpleHandlerFactory-Integrated	*.ashx	Habilitado	No especificado	System.Web.UI.SimpleHandle...	Heredada
WebServiceHandlerFactory-ISAPI-2.0-64	*.asmx	Habilitado	No especificado	IsapiModule	Heredada
WebServiceHandlerFactory-ISAPI-2.0	*.asmx	Habilitado	No especificado	IsapiModule	Heredada
ScriptHandlerFactory	*.asmx	Habilitado	No especificado	System.Web.Script.Services.S...	Local
PageHandlerFactory-ISAPI-2.0-64	*.aspx	Habilitado	No especificado	IsapiModule	Heredada
PageHandlerFactory-ISAPI-2.0	*.aspx	Habilitado	No especificado	IsapiModule	Heredada
PageHandlerFactory-Integrated	*.aspx	Habilitado	No especificado	System.Web.UI.PageHandlerF...	Heredada
AXD-ISAPI-2.0-64	*.axd	Habilitado	No especificado	IsapiModule	Heredada
AXD-ISAPI-2.0	*.axd	Habilitado	No especificado	IsapiModule	Heredada
HttpRemotingHandlerFactory-rem-ISAPI-2.0-64	*.rem	Habilitado	No especificado	IsapiModule	Heredada
HttpRemotingHandlerFactory-rem-ISAPI-2.0	*.rem	Habilitado	No especificado	IsapiModule	Heredada
HttpRemotingHandlerFactory-rem-Integrated	*.rem	Habilitado	No especificado	System.Runtime.Remoting.C...	Heredada
HttpRemotingHandlerFactory-soap-ISAPI-2.0-64	*.soap	Habilitado	No especificado	IsapiModule	Heredada
HttpRemotingHandlerFactory-soap-ISAPI-2.0	*.soap	Habilitado	No especificado	IsapiModule	Heredada
HttpRemotingHandlerFactory-soap-Integrated	*.soap	Habilitado	No especificado	System.Runtime.Remoting.C...	Heredada
svc-ISAPI-2.0-64	*.svc	Habilitado	No especificado	IsapiModule	Heredada
svc-ISAPI-2.0	*.svc	Habilitado	No especificado	IsapiModule	Heredada
svc-Integrated	*.svc	Habilitado	No especificado	System.ServiceModel.Activati...	Heredada
ScriptHandlerFactoryAppServices	*_AppService.axd	Habilitado	No especificado	System.Web.Script.Services.S...	Local
ScriptResource	ScriptResource.axd	Habilitado	No especificado	System.Web.Handlers.ScriptR...	Local
TraceHandler-Integrated	trace.axd	Habilitado	No especificado	System.Web.Handlers.TraceH...	Heredada
WebAdminHandler-Integrated	WebAdmin.axd	Habilitado	No especificado	System.Web.Handlers.WebAd...	Heredada
AssemblyResourceLoader-Integrated	WebResource.axd	Habilitado	No especificado	System.Web.Handlers.Assem...	Heredada

Ilustración 13: Asignación de Controladores IIS 7.0, Windows Vista, Microsoft Corporation

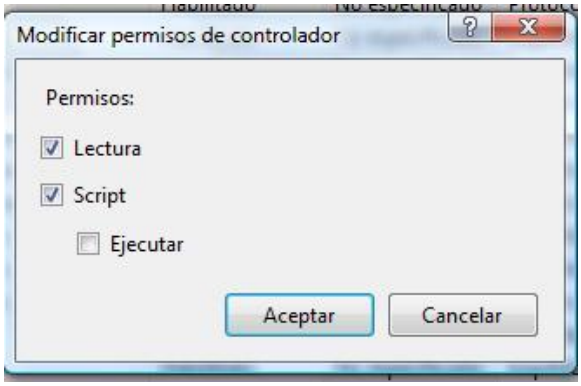


Ilustración 14: Modificar Permisos Controladores IIS 7.0, Windows Vista, Microsoft Corporation

Aún más de grano grueso es el mecanismo de restricción por IP y del Domain Name proporcionado por IIS en instalaciones del servidor de Windows. Usándolo, puedes conceder el acceso a todos los anfitriones con excepción de alguno especificado, o reversible, negar el acceso a todos sino los anfitriones particulares. Como con permisos Web, este mecanismo se puede configurar para controlar el acceso a un sitio web entero o abajo a los archivos específicos.

Registro

La revisión de la seguridad no es una opción para IIS, aunque sus instalaciones de registración pueden servir como substitución parcial. Si está configurado para hacer así pues, IIS registra la información en formato de texto sobre peticiones del HTTP en %winnt% \ system32 \ LogFiles \ W3SVC<n>, donde está el <n> número del caso del sitio Web. Puedes seleccionar, por caso del sitio Web, qué información se registra sobre procesado solicita. Hay alrededor 20 detalles específicos y un número de características de proceso de la contabilidad que se podrían registrar en cada petición. En Windows Vista

se maneja el concepto de reglas de seguimiento por errores y esto lo que haces es que cuando hay un error en el tipo de petición esta se guarda en el archivo de registro.

CAPITULO VI

Servicios web desde el Punto de Vista de J2EE y Como se Protegen

Una de las promesas de los servicios Web es la capacidad de hacer tu servidor de aplicaciones existente disponibles para tus empleados, clientes, y socios sin importar si son local o remoto, establecer una conexión ocasional, o tener una relación a largo plazo con tu compañía. Se parecería que la plataforma de Java 2 edición de la empresarial (J2EE), incluyendo los Enterprise Java Beans (EJB), cabrían en conformidad con este nuevo paradigma distribuido.

El sistema de software que provee los servicios relacionados con EJB al desarrollador de la aplicación es el servidor de aplicación. Servidores de aplicaciones que proporcionan un ambiente conveniente para la construcción de aplicaciones de negocio distribuido, están extensamente disponibles a número de vendedores, incluyendo IBM, BEA, Oracle, Sun e Iona. La mayor parte de estos vendedores han aumentado sus servidores de aplicaciones para usar servicios Web. Porque los servidores de aplicación se apuntan en el despliegue de la empresa, no es ninguna sorpresa que la seguridad esté generalmente tratado en estas arquitecturas. Sin una buena solución de la seguridad que protege los datos corporativos sobre un servidor de aplicación, la mayoría de los negocios no estarían dispuestos a hacer sus datos accesibles a los clientes Web a través de Internet.

Aun cuando Java es un sistema plataforma-neutral, tiene algunas áreas que no tratan todavía completamente los requisitos de los servicios Web. Por ejemplo, hasta hace poco

tiempo no había especificación en cómo manejar mensajes SOAP, la cual es uno de los protocolos básicos de la mensajería de los Servicios Web de como se definen comúnmente. El trabajo en curso de la comunidad de proceso de Java (JCP) está definiendo un número de estos pedazos que faltan. Estas extensiones a Java toman la forma de peticiones de la especificación de Java (JSRs).

El modelo de Java cabe bien en la mayoría de los casos cuando se utiliza como servicio Web. Una de las nuevas ediciones para los servicios Web basados en Java se refiere a la naturaleza del acceso a un contenedor de Java, ya sea un servlet, o un contenedor de EJB. Tradicionalmente, una petición a un contenedor de Java utilizó el protocolo del HTTP con la petición en el encabezado del HTTP, cookies, y/o el POST del mensaje HTTP. En el caso de los servicios Web, el formato del mensaje es SOAP.

El contenedor de Java se debe preparar para manejar seguridad del SOAP. Sin embargo, hay que tener en cuenta los servidores tradicionales de Java no pueden incluso manejar los mensajes llanos de SOAP, aún menos confiar en la seguridad de SOAP, sin una cierta actualización de ayuda. Por ejemplo, los contenedores basados en EJB solamente en EJB 2.0 no pueden manejar seguridad del SOAP y el SOAP como tal y no pueden así participar como servicio Web seguro sin ayuda externa. Sin embargo, los contenedores de Java que han puesto el JSR en ejecución antedicho pueden manejar seguridad del SOAP. Si vas a utilizar un EJB el servidor de aplicaciones para desplegar un servicio Web, sea seguro que compruebas si la versión que utilizarás se haya aumentado para ser consecuente con el JSR pertinente.

Comparado a .NET, Java es una tecnología madura que ha ido funcionando para varios años. Por otra parte, Java no ha utilizado servicios Web hasta hace poco tiempo. Además, como puedes ver el JSR nuevo que acabamos de discutir, más trabajo es necesario permitir que Java este totalmente acoplado con el mundo de los servicios Web.

La transición del modelo de la seguridad de Java a los servicios Web es evolutiva, y estructuras en mecanismos existentes de la seguridad. El acercamiento evolutivo celebra no sólo para el modelo de Java pero también para .NET. Java y .NET se han concentrado primero en seguridad del perímetro, según lo apoyado por los servidores Web tales como IIS en el caso de .NET, y Apache Tomcat en el caso de Java.

Java WSDP 1.5

Para solucionar los problemas de compatibilidad que se estaban presentando, Sun Microsystems presento un api encargada de manejar los diferentes estándares que presentaban en lo referido a los servicios web.

El Java Servicios web Developer Pack 1.5 (Java WSDP 1.5) es una caja de herramientas integrada que permite que los desarrolladores de Java a desarrollar, probar, y desplegar servicios de web rápida y fácilmente. La Java WSDP 1.5 es una descarga junta que contiene las tecnologías dominantes, las herramientas, y APIs que simplifiquen el desarrollo de los servicios de web usando la plataforma de Java 2. Java WSDP 1.5 proporciona puestas en práctica de los estándares de los servicios de web incluyendo el idioma descriptivo de los servicios del Web (WSDL), el protocolo simple del acceso del

objeto (SOAP), y la descripción, el descubrimiento y la integración universales (UDDI). Además, proporciona las puestas en práctica para el desarrollo de aplicaciones web tal como páginas de Java Server (páginas de JSP) y la biblioteca estándar de la etiqueta de JSP (JSTL).

Java WSDP 1.5 también está continuando con los estándares dominantes proporcionando la ayuda para los perfiles de la organización de la interoperabilidad de los servicios Web (WS-I) para promover interoperabilidad entre servicios web, así como la especificación de la seguridad de los servicios del Web del OASIS. Vale el observar que a pesar de que la Java WSDP acentúa servicios de web, el paquete es una caja de herramientas integrada que permite que los desarrolladores de Java desarrollen y desplieguen no sólo servicios de web, pero también web y aplicaciones basadas en XML. Tales aplicaciones utilizan tecnologías comunes de XML y de los servicios web, y eso es porque todas las puestas en práctica relacionadas se lían juntas.

La seguridad se puede proporcionar en dos niveles:

- **Seguridad a nivel de Transporte**: Esto se puede hacer usando JAX-RPC para la autentificación básica o la autentificación HTTP/SSL excesivo de la certificación del cliente.
- **Seguridad a nivel de mensaje**: La información de la seguridad se contiene dentro del mensaje SOAP. Es decir la información de la seguridad viaja junto con el mensaje. Esto tiene la ventaja de permitir que las piezas del mensaje sean transportadas sin los nodos intermedios que ven o que modifican el mensaje. Por ejemplo, una porción de un mensaje se puede firmar por un remitente y cifrar para un receptor particular.

La porción cifrada se puede descifrar solamente por el receptor previsto. Java WSDP 1.5 proporciona un marco para los reveladores de uso de JAX-RPC que permite firmar/verifican mensajes del SOAP, los cifran/descifran piezas de mensajes, y realizan la autentificación basada nombre de usuario y contraseña.

Este marco (XWS-Seguridad) se basa en la especificación de la base de la seguridad de los servicios Web del OASIS (WSS)[9], y la puesta en práctica se ha probado para la interoperabilidad con las puestas en práctica de otros vendedores con la participación en acontecimientos interoperable del OASIS. El marco actual utiliza la puesta en práctica de XML-DSig [10]de Apache, que se basa en sintaxis de la firma de W3C XML y el proceso, y XML-Enc[11] de Apache que se base en sintaxis y el proceso del cifrado de W3C XML.

Seguridad desde J2EE

Las aplicaciones de Java EE, web, y servicios web se componen de los componentes que se pueden desplegar en diversos contenedores. Estos componentes se utilizan para construir aplicaciones empresariales de multicapa. La seguridad para los componentes es proporcionada por sus contenedores. Un envase proporciona dos clases de seguridad: seguridad declarativa y programática.

[9] Especificación de Seguridad OASIS: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
[10] Sintaxis de Firmas y Procesamiento de la W3C: http://www.w3.org/TR/xmldsig-core/
[11] Sintaxis de Encriptación XML y Procesamiento: http://www.w3.org/TR/xmlenc-core/

- **La seguridad declarativa**: Expresa la seguridad de los requisitos de los componente de la aplicación usando descriptores del despliegue. Los descriptores de despliegue son externo a la aplicación, e incluir la información específica cómo los roles de la seguridad y los requisitos de acceso trazado en ambiente: específicos roles, usuarios, y políticas de la seguridad.
- **La seguridad programática**: Incrustada en la aplicación y se utiliza para tomar decisiones de la seguridad. La seguridad programática es útil cuando la seguridad declarativa solamente no es suficiente para expresar el modelo de la seguridad de una aplicación.
- **Las anotaciones**: También llamadas los metadatos, se utilizan para especificar la información sobre seguridad dentro de un archivo de la clase. Cuando se despliega la aplicación, esta información se puede utilizar o eliminar por el despliegue del uso descriptor.

Mecanismos de Seguridad

Las características de una aplicación deben ser consideradas al decidir capa y tipo de seguridad que se preverá. Las secciones siguientes discuten las características de los mecanismos comunes que se pueden utilizar para asegurar las aplicaciones de Java EE. Cada uno de estos mecanismos se puede utilizar individualmente o con otros para proporcionar las capas de la protección basadas en las necesidades específicas de tu puesta en práctica.

El SE de Java proporciona soporte para una variedad de características y de mecanismos de la seguridad, incluyendo:

- **Autentificación de Java y servicio de la autorización** (JAAS): JAAS es un sistema de APIs que permite a servicios autenticar y hacer cumplir controles de acceso sobre usuarios. JAAS proporciona un marco *conectable* y extensible para autentificación de usuarios programático y autorización. JAAS es una base J2SE 1.5 API y es una tecnología subyacente para los mecanismos de la seguridad de Java EE.
- **Servicios de seguridad genéricos de Java** (Java GSS-API): Java GSS es una api basada en tokens. El API intercambiaba con seguridad mensajes entre aplicaciones que se comunican.
- **Extensión de la criptografía de Java** (JCE): JCE proporciona un marco y puestas en práctica para el cifrado, acuerdo dominante de la generación y de la llave, y mensaje Algoritmos de código de autentificación (MAC). Ayuda para el cifrado incluye bloque simétrico, bloque asimétrico, y cifras de flujo. Cifras del bloque funcionan en grupos de bytes mientras que las cifras de flujo funcionan en un byte al tiempo. El software también apoya flujos seguros y objetos sellados.
- **Extensión de sockets segura de Java** (JSSE): JSSE proporciona un marco y puesta en práctica para una versión de Java de los protocolos del SSL y de TLS y incluye la funcionalidad para el cifrado de datos, autentificación del servidor, integridad de mensaje, y autentificación opcional del cliente para permitir comunicaciones seguras del Internet.
- **Autentificación simple y capa de seguridad** (SASL): SASL es un estándar de Internet (RFC 2222) que especifica un protocolo para la autentificación y el establecimiento opcional de una capa de seguridad entre el cliente y los servidores de aplicaciones. SASL define cómo los datos de la autentificación deben ser intercambiados pero no sí especifica el contenido de los datos el mismo.

Java SE también proporciona un sistema de las herramientas para manejar keystores, certificados, y política archivos; generando y verificando firmas del jars; y obteniendo, enumerarando, manejo de tickets de Kerberos.

Mecanismos de Seguridad de J2EE

Los servicios de seguridad de Java EE son proporcionados por el contenedor del componente y se pueden poner en ejecución usando técnicas declarativas o programáticas. Servicios de seguridad de Java EE proporcionar un mecanismo robusto y fácilmente configurable para autenticar usuarios y autorizar acceso a las funciones de la aplicación y a los datos asociados en muchas de las diversas capas. Los servicios de seguridad de Java EE están a parte de los mecanismos de la seguridad del sistema operativo.

Asegurando los Contenedores

En Java EE, los contenedores componentes son responsables de proporcionar seguridad de la aplicación. Un contenedor proporciona dos tipos de seguridad: declarativo y programático.

La seguridad declarativa expresa requisitos de la seguridad de un componente de la aplicación usando descriptores del despliegue. Un descriptor del despliegue es un documento de XML con una extensión de .xml que describe los ajustes del despliegue de

una aplicación, de un módulo, o de un componente. Porque la información del descriptor del despliegue es declarativa, puede ser cambiada sin la necesidad de modificar el código de fuente. En el tiempo de pasada, el servidor de Java EE lee el descriptor del despliegue y actúa sobre la aplicación, el módulo, o el componente por consiguiente.

Las anotaciones permiten un estilo declarativo de la programación, y así que abarcar ambos los conceptos declarativos y programáticos de la seguridad. Los usuarios pueden especificar la información sobre seguridad dentro de un archivo de la clase usando anotaciones. Cuando la aplicación es desplegada, esta información es utilizada por el servidor de aplicación. No toda la seguridad la información se puede especificar usando anotaciones, sin embargo. Una cierta información debe especificarte en los descriptores del despliegue de la aplicación. Esto conduce a un estilo de programación declarativo, donde el programador dice lo que debe ser hecho y las herramientas emiten el código para hacerlo. También elimina la necesidad de mantener los archivos laterales que se deben mantener actualizados con los cambios en archivos de fuente. En lugar la información se puede mantener en el archivo de fuente.

La seguridad programática se incrusta en la aplicación y se utiliza para tomar decisiones de la seguridad. La seguridad programática es útil cuando es la seguridad declarativa solamente no es suficiente expresar el modelo de la seguridad de una aplicación. El API para la seguridad programática consiste en dos métodos del interfaz de EJBContext y dos métodos del interfaz de HttpServletRequest del servlet. Estos métodos permitir que los componentes tomen las decisiones de la lógica del negocio basadas en el papel de la seguridad del usuario que hace la petición o del usuario remoto.

Asegurando el Servidor de Aplicaciones

Se puede configurar el Sun Java System Application Server para los siguientes propósitos:

- Agregando, suprimiendo, o modificando usuarios autorizados.
- Configura a “listeners” seguros de HTTP y de IIOP.
- Configura los conectores seguros de JMX.
- Agregar, suprimir, o modificar instancias personalizadas.
- Define un interfaz para proveedores de autorización conectable que usan contratos de autorización de Java para los contenedores (JACC). El contrato de autorización de Java para los contenedores (JACC) define contratos de la seguridad entre el servidor de aplicaciones y los módulos de la política de la autorización. Estos contratos especifican cómo los proveedores de la autorización están instalados, configurados, y utilizados en decisiones del acceso.
- Usando los módulos conectables de auditoría.
- Cambiando la política de servicios para una aplicación.

Las características siguientes son específicas al servidor de aplicaciones:

- Seguridad del mensaje
- Un solo ingreso a través de todo el servidor de aplicaciones con un solo dominio seguro.
- Ingreso de sesión programable

Trabajando con Roles, Grupos de Usuarios, Roles y Dominios

Necesitas a menudo proteger los recursos para asegurarnos de que solamente los usuarios autorizados tienen acceso. La autorización proporciona el acceso controlado a los recursos protegidos. La autorización se basa en la identificación y la autentificación.

La identificación es un proceso que permite el reconocimiento de una entidad por un sistema, y la autentificación es un proceso que verifica la identidad del usuario, del dispositivo, o de la otra entidad en un sistema informático, generalmente como requisito previo a permitir el acceso a los recursos en un sistema.

Antes de empezar a explicar cómo autorizar usuarios desde *Sun Java System Application Server Admin Console* es prudente explicar ciertos conceptos de cómo se manejan los usuarios de J2EE.

Dominio *(Realm):* Para una Aplicación Web, un "dominio" es una base de datos completa de los usuarios y de los grupos que identifican a usuarios válidos de un Aplicación Web (o de un sistema de aplicaciones web) y es controlado por la misma política de autentificación.

El servicio de autentificación del servidor de Java EE puede gobernar a usuarios en múltiples dominios o *reinos*. En este lanzamiento del servidor de aplicación, el archivo, el realm-admin, y los reinos de certificados vienen pre configurado para el servidor de

aplicaciones. En el archivo del dominio, el servidor almacena las credenciales del usuario localmente en un archivo nombrado *keyfile*. Puedes utilizar la consola del Administrador para manejar a usuarios en el archivo del dominio. Al usar el archivo del dominio, el servicio de autentificación del servidor verifica la identidad del usuario comprobando el archivo del dominio.

Este dominio se utiliza para la autentificación de todos clientes a excepción de los clientes del navegador web que utilizan el protocolo y certificados de HTTPS. En el dominio certificado, el servidor almacena las credenciales del usuario en una base de datos de certificados. Al usar el dominio de certificado, el servidor utiliza certificados con Protocolo de HTTPS para autenticar a clientes de la web. Para verificar la identidad de un usuario en el dominio de certificado, el servicio de la autentificación verifica un certificado X.509.

El campo conocido común del certificado X.509 se utiliza como el nombre principal. El administrador de dominio es también un FileRealm y almacena las credenciales del usuario del administrador localmente en un archivo nombrado admin-keyfile. Puedes utilizar la consola del Admin para manejar a usuarios en este reino que manejas de la misma forma a usuarios en el reino del archivo.

Usuarios: Un usuario es una identidad individual (o de una aplicación) que se ha definido adentro el servidor de aplicaciones. En una aplicación web, un usuario puede tener un sistema de papeles o roles asociados con esa identidad, que les da derecho a tener acceso todos los recursos protegidos con esos roles. Los usuarios pueden ser asociados a un grupo. Un usuario de Java EE es similar a un usuario del sistema operativo. Típicamente,

ambos tipos de los usuarios representan a gente. Sin embargo, estos dos tipos de usuarios no son iguales.

El servicio de autentificación del servidor de Java EE no tiene ningún conocimiento del nombre del usuario y contraseña que proporcionas cuando abres una sesión al sistema operativo. El servicio de la autentificación del servidor de Java EE no está conectado con el mecanismo de la seguridad del sistema operativo. Los dos servicios de seguridad manejan a usuarios que pertenecen a diferentes dominios.

Grupos: Un grupo es un sistema de usuarios autenticados, clasificado por los rasgos comunes, definidos en el servidor de aplicaciones. Un usuario de Java EE del archivo de dominio puede pertenecer a un grupo del servidor de aplicaciones. (El usuario de A en el dominio del certificado no puede.) Un grupo del servidor de aplicaciones es una categoría de los usuarios clasificados por rasgos comunes, tales como título del trabajo o perfil del cliente. Por ejemplo, la mayoría de los clientes de una aplicación de comercio electrónico pudieron pertenecer al grupo del CLIENTE, pero los grandes compradores pertenecerían al grupo PREFERIDO.

Categorizar a usuarios en grupos hace más fácil controlar el acceso de una gran cantidad de usuarios. Un grupo del servidor de aplicaciones tiene un diverso alcance de un rol. Un grupo del servidor de aplicaciones se señala para el servidor de aplicaciones enteros, mientras que un rol se asocia solamente a una aplicación específica en el servidor de aplicaciones.

Rol: Un rol es un nombre abstracto para el permiso a accesar a un sistema particular de recursos en una aplicación. Un rol se puede comparar a una llave que pueda abrir una cerradura. Mucha gente pudo tener una copia de la llave. La cerradura no cuida quiénes eres, sólo que tú tienes la llave correcta.

Como Asegurar una Aplicación usando Roles de Seguridad de Java

En aplicaciones, se definen los roles usando anotaciones o en descriptores del despliegue de las aplicaciones tales como web.xml, ejb-jar.xml, y uso. XML

Puedes declarar los nombres de los roles de seguridad usados en aplicaciones web usando la anotación de los @DeclareRoles (preferida) o los elementos security-role-ref del descriptor del despliegue. Declarar nombres del rol de seguridad de esta manera te permite ligar los nombres del rol de seguridad usados en el código a los roles de seguridad definidos para una aplicación montada. En ausencia de este paso de enlace, cualquier nombre del rol de usado en el código será asumido que corresponde a un rol de seguridad del mismo nombre en la aplicación montada.

Una referencia de rol de seguridad (security-role-ref), incluyendo el nombre definido por la referencia, tiene como alcance al componente que contiene la anotación de la clase: los @DeclareRoles o elemento descriptor del despliegue contiene el despliegue security-role-ref elemento del descriptor.

Se puede también utilizar los elementos security-role-ref para esas referencias que fueron declaradas en anotaciones y desea haber enlazado a un rol de seguridad cuyo nombre difiera del valor de referencia. Si una referencia del rol de seguridad no se enlaza a un rol de seguridad en esta manera, el contenedor debe trazar el nombre de la referencia al rol de seguridad del mismo nombre.

Especificando Roles de Seguridad Usando Anotaciones

Las anotaciones son la mejor manera de definir roles de seguridad en una clase o un método. La anotación de los @DeclareRoles se utiliza para definir los roles de seguridad que abarcan el modelo de la seguridad de la aplicación. Esta anotación se especifica en una clase, y sería utilizada típicamente para definir los roles que se podrían probar (por ejemplo, llamando el isUserInRole) dentro de los métodos de la clase anotada.

Lo que sigue es un ejemplo de cómo esta anotación sería utilizada. En este ejemplo, el empleado es el único rol de seguridad especificado, pero el valor de este parámetro puede incluir una lista de los roles de la seguridad especificados por la aplicación.

```
DeclareRoles("RolePrueba")
public class ClasePrueba {
//...
}
```

Ilustración 15: Código de Ejemplo de una clase usando una anotación, Lenguaje Java

Especificar los @DeclareRoles (“RolePrueba”) es equivalente a definir el siguiente en el web.xml:

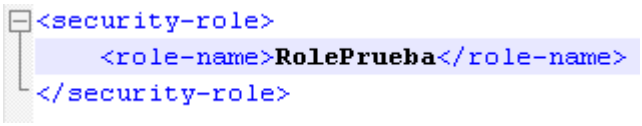


Ilustración 16: Código del Archivo XML donde se especifica el Rol

Esta anotación no se utiliza para enlazar roles de la aplicación a otros roles. Cuando tal enlace es necesario, es logrado definiendo una referencia apropiada del security-role-ref en el descriptor asociado de despliegue.

Cuando una llamada se hace al isUserInRole de la clase anotada, la identidad del "llamador" asociada a la invocación de la clase se prueba para la calidad de miembro en el rol con el mismo nombre que la discusión al isUserInRole. Si una referencia del security-role-ref se ha definido para el nombre del rol de la discusión, prueban al llamador para la calidad de miembro en el rol trazada al nombre del rol.

Especificando Roles de Seguridad

El fragmento de código XML siguiente de un descriptor del despliegue se toma como ejemplo para mostrar todos los elementos necesitados para especificar roles de la seguridad usando descriptores del despliegue:

Ilustración 17: Fragmento de Código XML Especificando los Permiso de Un Rol en Específico

En este ejemplo, el elemento del rol de seguridad enumera todos los roles de seguridad usados adentro de la aplicación: role1 y empleado. Esto permite al desplegado trazar todo el roles definidos en la aplicación a los usuarios y a los grupos definidos en la misma

El elemento del auth-constraint especifica los roles (role1, empleado) que pueden tener acceso a los métodos del HTTP (PUT, DELETE, GET, POST) situados en el directorio especificado por el elemento del patrón de la URL. Podrías también haber utilizado la anotación de los @DeclareRoles en el código de fuente para lograr esta tarea.

Se utiliza el elemento security-role-reference cuando una aplicación utiliza el HttpServletRequest. método del isUserInRole (String Role). El valor del nombre del rol debe ser la secuencia usada como el parámetro al HttpServletRequest. Método del isUserInRole (String Role). El role-link debe contener nombre de uno de los roles de la seguridad definidos en los elementos del security-role. El contenedor utiliza el trazado de security-role-reference al security-role cuando se determina del valor de retorno de la llamada.

Estableciendo Conexiones Seguras

La Tecnología SSL (Secure Socket Layer) es la seguridad que se pone en ejecución en la capa de transporte. El SSL permite que los navegadores y los servidores de web se comuniquen sobre una conexión segura. En esta conexión segura, los datos se están enviando se cifran antes de ser enviado y entonces se descifran al ser recibido y antes de procesar. El navegador y el servidor cifran todo el tráfico antes de enviar cualquier dato. Un conector del SSL HTTPS está ya habilitado en Sun Java System Application Server.

Especificar un requisito que protege los recursos a ser recibido sobre a la conexión protegida de la capa de transporte (SSL), especifican una restricción de los datos del usuario en el descriptor del despliegue de la aplicación. Lo que sigue es un ejemplo de un descriptor del despliegue de una aplicación de web.xml que especifica que el SSL está utilizado:

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>view dept data</web-resource-name>
        <url-pattern>/hr/employee/*</url-pattern>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
    </web-resource-collection>
    <auth-constraint>
        <role-name>DEPT_ADMIN</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

Ilustración 18: Especificación de Cómo se debe usar las restricciones de las conexiones SSL, The Java EE 5 Tutorial

Una restricción de los datos del usuario (<user-data-constraint> en el descriptor del despliegue) requiere que todos los patrones obligados del URL y los métodos del HTTP especificados en la restricción de la seguridad sean recibida sobre la conexión protegida de la capa de transporte tal como HTTPS (HTTP sobre el SSL). Una restricción de los datos del usuario especifica una garantía del transporte (<transport-guarantee> en el descriptor del despliegue). Las opciones para la garantía del transporte no incluyen CONFIDENTIAL, INTEGRAL, o NONE. Si especificas CONFIDENTIAL o INTEGRAL como restricciones de la seguridad, ese tipo de restricciones de la seguridad se aplica a toda la peticiones que hacen juego con los patrones del URL en el recurso de web de la colección y no apenas a la caja de diálogo de inicio de sesión.

La fuerza de la protección requerida es definida por el valor de la garantía del transporte.

- ✓ CONFIDENTIAL: cuando la aplicación requiere que los datos estén transmitidos para evitar que otras entidades observen el contenido de la transmisión.

- ✓ NTEGRAL: cuando la aplicación requiere que los datos estén enviados entre el cliente y el servidor de una manera tal que no pueda ser cambiado en tránsito.
- ✓ NONE: indica que el contenedor debe aceptar las peticiones restringidas en cualquier conexión, incluyendo desprotegida.

Las restricciones de los datos del usuario es práctico utilizar con la autentificación básica. Cuando el método de la autentificación de la conexión se fija al BASIC o FORM, las contraseñas no se protegen, significando que las contraseñas enviadas entre un cliente y un servidor en una sesión desprotegida se pueden ver e interceptar por los terceros. Usar una restricción de los datos del usuario con el mecanismo de la autentificación del usuario puede aliviar esta preocupación.

CAPITULO VI

.Net Remoting

.NET Remoting es una tecnología para la comunicación entre diferentes "*application domains*"[12]. Usando .NET Remoting para la comunicación entre application domains, puede suceder dentro del mismo proceso, entre procesos o en un mismo sistema, o entre procesos de diferentes sistemas.

Diferentes tecnologías pueden ser usadas para la comunicación con una aplicación cliente y una aplicación servidor. Se puede programar la aplicación usando sockets, o se puede usar las clases que provee el Framework del namespace System.Net que hacen mucho más sencillo trabajar con los diferentes protocolos, direcciones IP, y puertos. Usando esta tecnología siempre. Usando esta tecnología siempre tienes que enviar datos a través de la red. Los datos que envías pueden ser de tu propio protocolo personalizado donde el paquete sea interpretado por el servidor, de tal manera que el servidor conozca los métodos que deben ser invocados, no tendría que manejar solamente los datos que son enviados, sino también crear los hilos.

[12] Según Microsoft en sus diapositivas de "**Programa Microsoft Desarrollador Cinco Estrellas**" dice: "Un Application Domain (o Frontera de Aplicación) es la mínima unidad de aislamiento de aplicaciones dentro del CLR, y todos los assemblies que conforman una aplicación siempre son cargados dentro de uno. El aislamiento entre aplicaciones garantiza que:

- Una aplicación pueda ser detenida independientemente del resto
- Una aplicación no pueda acceder directamente a código en ejecución o recursos de otra aplicación (el CLR prohíbe invocaciones directas entre objetos cargados en distintos Application Domains)

Una falla en una aplicación no afecte al resto de las aplicaciones en ejecución."

Usando los Servicios Web de XML, se puede enviar mensajes a través de la red, con estos también se tiene independencia de la plataforma, y perdida de acoplamiento entre los clientes y el servidor, lo cual significa que es mucho más sencillo lidiar con los problema de diferentes versiones del programa.

Con .Net Remoting se tiene un alto nivel de acoplamiento entre el cliente y el servidor, ya que se comparte objetos del mismo tipo. Este trae la funcionalidad de los objetos CLR para los métodos que son invocados a través de diferentes application domains.

Los Assemblies Remotos pueden ser configurados para trabajar localmente en el application domain o como parte de una aplicación remota. Si el assembly es parte de una aplicación remota, el cliente recibe un proxy para hablar en vez del objeto verdadero. El proxy es un representante del objeto remoto en el proceso del cliente, usado por la aplicación del cliente para llamar métodos. Cuando el cliente llama un método en el proxy, este envía un mensaje en el canal que se pasa al objeto remoto.

Las Aplicaciones de .NET trabajan dentro de un dominio de aplicación (application domain). Un dominio de aplicación se puede considerar como subprocesos dentro de un proceso. Tradicionalmente, los procesos fueron utilizados como límite del aislamiento. Una aplicación que funciona en un proceso no puede tener acceso y destruir memoria en otro proceso. La comunicación del a través de procesos es necesaria para que las aplicaciones se comuniquen una con otra. Con .NET, el dominio de aplicación es el nuevo límite de seguridad dentro de un proceso, porque el código de MSIL es tipo-seguro y comprobable. Diversas aplicaciones pueden funcionar en el interior el mismo proceso

pero dentro de diversos dominios de aplicación. Los objetos dentro del mismo dominio de aplicación pueden obrar recíprocamente directamente; un proxy es necesario para tener acceso a objetos en un diverso dominio de aplicación.

La lista siguiente proporciona una descripción de los elementos dominantes de esta arquitectura:

Un objeto remoto es un objeto que está funcionando en el servidor. El cliente no llama métodos en este objeto directamente, sino que usa a su vez un proxy. Con .NET es fácil distinguir objetos remotos de objetos locales: cada clase que se deriva de *MarshalByRefObject* nunca sale de su dominio de aplicación. El cliente puede llamar métodos del objeto remoto vía proxy.

Un canal se utiliza para la comunicación entre el cliente y el servidor. Hay piezas del cliente y del servidor del canal. El Framework 2.0 de .NET ofrece los tipos de canal que se comunican vía el TCP, el HTTP, o el IPC. Puedes también crear un canal personalizado que se comunique usando un diverso protocolo.

Los mensajes se envían en el canal; se crean para la comunicación entre el cliente y el servidor. Estos mensajes contienen información sobre el objeto remoto, el nombre del método llamado, y todos los argumentos.

El definidor de formato define cómo los mensajes se transfieren en el canal. .NET 2.0 tiene SOAP y definidores de formato binarios. El definidor de formato SOAP se puede utilizar

para comunicarse con los servicios web que no se basan en el framework de .NET. Los definidores de formato binarios son mucho más rápidos y se pueden utilizar eficientemente en un ambiente del Intranet. Por supuesto, también tienes la opción para crear un definidor de formato personalizado.

Proveedor definido de formato para asociar un definidor de formato a un canal. Creando un canal, puedes especificar el Proveedor del definidor de Formato que se utilizará, y éste alternadamente define el definidor de formato que se utiliza para transferir los datos en el canal.

El cliente llama métodos en un proxy en vez del objeto remoto. Existen dos tipos de proxys: el proxy transparente y proxy real. Para el cliente, el proxy transparente parece el objeto remoto. En el proxy transparente, el cliente puede llamar los métodos puestos en ejecución por los objetos remotos. Alternadamente, el proxy transparente llama el método de Invoke () en el proxy real. El método de Invoke () utiliza el tubería de mensajes para pasar el mensaje al canal.

Una tubería de Mensaje, es un objeto interceptor. Los interceptores se utilizan tanto en el cliente como en el servidor. Una tubería se asocia al canal. El proxy real utiliza la tubería de mensaje para pasar el mensaje en el canal, así que la tubería puede hacer una cierta interceptación antes de que el mensaje entre el canal. Dependiendo de donde se está utilizando, se conoce como tubería de enviar, una tubería de contexto de servidor, una tubería de contexto de objeto, y así sucesivamente.

El cliente puede utilizar un activador para crear un objeto remoto en el servidor o para conseguir un poder de un objeto servidor-activado.

RemotingConfiguration es una clase para uso general para configurar los servidores y a clientes remotos. Esta clase se puede utilizar para leer archivos de la configuración o para configurar objetos remotos dinámicamente.

ChannelServices es una clase para uso general para colocar los canales y después para enviar mensajes a ellos.

CAPITULO VII

.Net Remoting vs RMI

Él Método de Invocación Remota de Java (RMI) permite invocar un método en un objeto que existe en otra dirección espacio, el cual puede ser la misma máquina o una diferente. Las aplicaciones RMI generalmente están compuestas por dos programas separados: uno servidor y uno cliente. En una aplicación típica de servidor los objetos remotos, hacen referencias a ellos accesiblemente, y espera por los clientes para invocar métodos en esos objetos remotos. Una aplicación cliente obtiene referencia de uno o más objetos remotos en el servidor y después invoca métodos sobre él.

Para un cliente localizar un objeto de servidor por primera vez, RMI depende una central de nombres o un directorio de servicios, el cual mantiene información disponible sobre los objetos de servidor. Un objeto de servidor hace sus métodos disponibles para la invocación remota uniéndolo a un nombre en repositorio de nombres. Un cliente adquiere una referencia del objeto a un servidor RMI haciendo una búsqueda (lookup) por una referencia de un objeto e invoca los métodos en el objeto de servidor tal como si el objeto del servidor RMI residiera en la dirección espacio del cliente.

RMI confía en los tipos similares de objetos que son automáticamente generados por el Compilador RMU desde la implementación del servidor: *stubs* y *skeletons*. Un stub es un objeto del lado del cliente que actúa como un proxy al servidor remoto e implementa la interfaz del objeto remoto. Es responsable por formar y destruir datos en el lado del

cliente. Un skeleton es un objeto del lado del servidor que envía llamadas a la actual implementación objeto remoto y es responsable de formar y destruir datos en el lado del servidor.

.Net remoting provee medios comprensivos de construir aplicaciones distribuidas. Se pueden construir aplicaciones que usan objetos en otros Application Domains en el mismo computador o cualquier otro que pueda ser alcanzable por su red. Es fácil configurar .Net Remoting para encontrar los requerimientos de las aplicaciones. Por ejemplo, puedes escoger diferentes protocolos de comunicación (HTTP, TCP, ICP), diferentes formatos de serialización (SOAP, Binario), diferentes tipos de invocación (síncrono y asíncrono), diferentes formas de activación de un objeto servidor (activación del cliente, activación del servidor), ciclos de vida de los objetos, etc.

Para usar .Net remoting para crear una aplicación en la cual los dos componentes se comunican directamente a través de los límites de un Application Domains, se necesita:

- Un objeto que pueda ser remoto, cuyos métodos pueden ser invocados desde diferentes Application Domains.
- Un host de la aplicación para escuchar las peticiones para el objeto.
- Una aplicación cliente para hacer peticiones para el objeto

Similitudes

Llamada de procedimientos Remotos (Remote Procedure Calls - RPC) es un mecanismo tradicional que permite a las aplicaciones llamar procedimientos que existen en otros computadores. RPC hace uso de métodos proxys el cual tiene la misma firma que el

método remoto, pero también el código para transferir datos entre el cliente y el servidor. Parámetros son juntados y enviados al servidor donde son separados y pasados al método pedido, el valor de retorno es tratado de la misma forma.

En un cierto sentido, RMI y .Net Remoting tienen apenas un acercamiento orientado al objeto construido encima de mecanismo ya existente del RPC. Mientras que el RPC permite que llames procedimientos sobre la red, RMI y .Net Remoting permite que llames los métodos de un objeto sobre la red. Los objetos se pasan hacia adelante y hacia atrás entre el cliente y el servidor como parámetros y valores de la vuelta.

Ni RMI ni .Net Remoting requiere una lengua secundaria para definir las interfaces remotas, tales como CORBA IDL. Puesto que Java RMI se junta firmemente con la lengua de Java y .Net Remoting se junta con la lengua de C#, ambas ayudas escoge la clase, herencia múltiple del interfaz.

RMI soporta Garbage Collector Distribuido (DGC). El servidor sigue a clientes que tienen su stub y guarda una cuenta de tales clientes. La cuenta disminuye cuando el cliente abandona explícitamente referencia. Si la cuenta de la referencia alcanza cero, el objeto es basura que es recogida. Net Remoting soporta Garbage Collector Distribuido usando cuentas de la referencia también. Además, el curso de la vida de objetos puede ser controlado asociando un arriendo a cada servicio alejado. Se suprime el servicio cuando expira.

Diferencias

Java RMI pasa por un directorio llamado RmiRegistry para almacenar las referencias de los objetos remotos. Este servicio, siendo sí mismo distribuyó el objeto, se debe lanzar manualmente o dinámicamente antes de cualquier comunicación entre un cliente y un servidor. .NET adopta un acercamiento diferente. En vez de usar los servidores de nombres, un cliente se comunica con el servidor en un puerto especificado primero. Por lo tanto, aplicaciones distribuidas desarrolladas en .Net Remoting no depende de ningún servidor de nombres para localizar los servicios remotos.

Una diferencia fundamental opone ambos framework es RMI requiere la creación de interfaces remota mientras que .Net Remoting no. Net Remoting no puede funcionar con un constructor que no sea por defecto al conectar con los objetos bien conocidos. Java RMI no tiene esta limitación.

Aunque varios artículos mencionan que es mucho más fácil desarrollar usando el código de librería de .Net Remoting que en Java RMI. Es propenso a cometer el error de escribir el archivo de la configuración y utilizar la línea de comando del csc interruptores. Sin embargo .Net Remoting proporciona una solución más completa y más flexible a desarrollo de aplicaciones distribuida. El framework se puede configurar depende de las necesidades de la aplicación. Por ejemplo, una aplicación puede utilizar el HTTP o el TCP como el protocolo de comunicación, y formateado del SOAP o formateado binario como la serialización del objeto. La consecuencia de esta flexibilidad es por supuesto una curva de aprendizaje más larga para tener una comprensión comprensiva del marco entero.

Java RMI incluye una puesta en práctica genérica y reutilizable de la fábrica, llamada framework de la activación, que maneja los detalles de servidores que lanzan en el telecontrol máquinas agradable y transparente. En vez de usar UnicastRemoteObject, el telecontrol la puesta en práctica extiende la clase de Activatable. En .Net Remoting, como ya hemos mencionado, los objetos del servidor pueden ser servidor activado o cliente activado. Para el servidor los objetos activados, pueden ser Singleton o SingleCall.

.Net Remoting se puede utilizar solamente en la plataforma de Windows. Aunque RMI/JRMP confía en Java, puede ser utilizado en cualquier sistema operativo que tenga una puesta en práctica virtual de la máquina de Java, extendiéndose de los chasis al UNIX a las ventanas. Aunque ambos las tecnologías se basan en el mecanismo orientado al objeto del RPC, su alto acoplamiento firmemente con sus idiomas indican que no lo hacen interoperable. RMI-IIOP proporciona una manera conveniente para cualquier lengua que "hable CORBA" para hablar con Java. El protocolo de CORBA IIOP es parte de la especificación desde JDK 1.3. Una de las ventajas de RMI-IIOP que CORBA excesivo es los reveladores no necesita aprender la lengua de la definición de interfaz de CORBA (IDL).

Conclusión

Un servicio web es definido como un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones, estos son aplicaciones modulares auto-contenidas, auto-descriptivas que pueden ser publicadas, localizadas e invocadas a través del ambiente web.

Estos están diseñados para ser multiplataforma y multilenguaje, así mismo tienen la habilidad de solucionar una gran cantidad de problemas a nivel empresarial que reducen costos de programación, operacionales y de respuesta.

Las tecnologías empleadas para el desarrollo de un servicio web en IIS son las siguientes:

- .Net Framework.
- HTTP.
- XML.
- SOAP.
- UDDI.
- WSDL.

La seguridad en un servicio web es primordial por ellos existen un número de servicios encargados de la seguridad que normal mente se utilizan para cumplir con estos requerimientos. Entre ellos están:

1. Criptografía
2. autenticación
3. Protocolos de uso Criptográfico
4. Sistemas de Autenticación
5. Autorización
6. Seguridad en tecnologías de capa intermedia

Bibliografía

Apple Computer, Inc. WebObjects Web Services Programming Guide [Libro]. - Cupertino, CA : Apple Computer, Inc., 2002, 2005.

Cabrera Luis Felipe and Kurt Chris Web Services Architecture and Its Specifications: Essentials for Understanding WS [Book]. - [s.l.] : Microsoft Press, 2005.

Christensen Erik (Microsoft) [et al.] Lenguaje de descripción de servicios Web (WSDL) 1.0 [Online] // Artículos técnicos en Español. - Ariba, International Business Machines Corporation, Microsoft, 2000. - 02 04, 2007. - http://www.microsoft.com/spanish/msdn/articulos/archivo/090201/voices/wsdl.asp#_3.__.

Hartman Bret [et al.] Mastering Web Services Securty [Book]. - Indianapolis, Indiana : Wiley Publishing, Inc., 2003.

Hoang Lam Thuan L. Thai .NET Framework, 3a Edición [Book]. - [s.l.] : O'Reilly, August 2003.

Hondo M. Nagaratnam N., Nadalin A. IBM System Journal [Book]. - 2002. - Vol. XLI.

http://www.programacionweb.net/
http://www.programacionweb.net/utilidades/headers.php?url=www.unitecnologica.edu.co [Online]. - 2003 - 2007.

Jendrock Eric [et al.] The Java™ EE 5 Tutorial [Book]. - Santa Clara, California : Addison-Wesley, 2006. - Vol. III.

Lenguaje de descripción de servicios Web (WSDL) 1.0 [Book].

MSDN Introducción a Microsoft .NET, Programa Microsoft Desarrollador Cinco Estrellas [Journal]. - 2006.

Rusty Harold Elliotte XML 1.1 Bible [Book]. - Indianapolis, Indiana : Wiley Publishing, Inc., 2004. - Vol. 3 ED..

Wikipedia Wikipedia - La enciclopedia Libre [Online]. - Enero 27, 2007. - Enero 30, 2007. - http://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol.