UNIVERSIDADE FUMEC
FACULDADE DE CIÊNCIAS EMPRESARIAIS
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ANA CLARA DE PINHO SCHREIBER
CASSIANO MEDEIROS SANTANA
JOÃO PAULO CARNEIRO ARAMUNI

WEB SERVICES COM JAX-WS

BELO HORIZONTE
2013

UNIVERSIDADE FUMEC
FACULDADE DE CIÊNCIAS EMPRESARIAIS
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ANA CLARA DE PINHO SCHREIBER
CASSIANO MEDEIROS SANTANA
JOÃO PAULO CARNEIRO ARAMUNI

WEB SERVICES COM JAX-WS

Atividade Auto Instrucional apresentada à Universidade Fumec, como requisito parcial para obtenção de pontos na disciplina de Desenvolvimento de Sistemas Distribuídos.
Prof. Flávio Velloso Laper.

BELO HORIZONTE
2013

SUMÁRIO

1. Definição e aplicações

1.1. Web Service

O termo Web Service descreve uma maneira padronizada de integrar aplicações web usando XML (Short Extensible Markup Language), SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) e UDDI (Universal Description Discovery and Integration). Cada um possuindo uma função especifica.

O que torna um web service distinto de uma aplicação web comum é o fato de o web service não prover o seu usuário com uma interface. Ao invés disso, web services compartilham lógicas de negócio, dados e processos através de um ambiente de programação por meio de uma rede.

Pela definição do W3C (World Wide Web Consortium) um web service é uma aplicação de software identificada por uma URI (Uniform Resource Identification) em que as interfaces e dependências podem ser definidas, descritas e identificadas como artefatos XML.

Em Java, os serviços Web são definidos por classes. Um Web Service pode oferecer várias operações, representada por um método de classe. Uma prática muito importante de ser seguida é sempre dividirmos a interface de um serviço da sua implementação. A interface de um serviço é chamada de SEI (Service Endpoint Interface) e a implementação é chamada de SIB (Service Implementation Bean).

Não há nada de muito diferente em relação a um código qualquer normal de interface Java. Porém nota-se que as anotações @WebService, @WebMethod e @SOAPBinding provavelmente são novidades para aqueles que ainda estão aprendendo sobre Web Services. @Webservice é uma anotação que avisa ao compilador Java que o arquivo atual corresponde à definição SEI de um serviço Web. @WebMethod indica que um determinado método corresponde a uma operação de serviço e assim pode ser invocado por um cliente. @SOAPBinding indica que o serviço utilizará a abordagem SOAP e não Rest que é a outra abordagem suportada

1.2. Web Service com Jax-WS

O JAX – WS (Java API for XML Web Services) é uma API de Java para a criação de web services. Ela faz parte da API Java EE da Oracle. Jax –WS provê muitas annotations para simplificar o desenvolvimento e distribuição de ambos clientes e endpoints de web services.

Uma continuação liberada de Java API para XML-based RPC 1.1 (JAX-RPC), JAX-WS simplifica a tarefa de desenvolver serviços web utilizando tecnologia Java. Encaminha alguns dos resultados em JAX-RCP 1.1 providenciando suporte para protocolos múltiplos assim como SOAP 1.1, SOAP 1.2, XML e suprindo uma facilidade de manutenção para protocolos adicionais junto com HTTP.

JAX-WS utiliza JAXB 2.0 para ligação de dados e suportes habituais para controlar serviços gerados em interfaces endpoint. Com seu apoio para anotações, JAX-WS simplifica o desenvolvimento de serviços web e diminui o tamanho do tempo de execução dos arquivos Jar.

1.3. Anotações do Jax-WS

O JAX-WS (Java API for XML-Based Web Services) baseia-se no uso de anotações para especificar os metadados associados às implementações de serviço da Web e para simplificar o desenvolvimento de serviços da Web. As anotações descrevem como uma implementação de serviço no lado do servidor é acessada como um serviço da Web ou como uma classe Java no lado do cliente acessa os serviços da Web.

O padrão de programação JAX-WS apresenta o suporte para anotação de classes Java com o metadados que é utilizado para definir um aplicativo de terminal em serviço como um serviço da Web e como um cliente pode acessar o serviço da Web. O JAX-WS suporta o uso de anotações com base na especificação Metadata Facility for the Java Programming Language (Java Specification Request (JSR) 175), na especificação Web Services Metadata for the Java Platform (JSR 181) e nas anotações definidas pela especificação JAX-WS 2.0 (JSR 224) que inclui as anotações JAXB. Utilizando as anotações a partir do padrão JSR 181, você pode simplesmente anotar a classe de implementação de serviço ou a interface de serviços e agora o aplicativo é ativado como um serviço da Web. O uso de

anotações na origem Java simplifica o desenvolvimento e a implementação de serviços da Web, definindo algumas das informações adicionais que geralmente são obtidas de arquivos do descritor de implementação, arquivos WSDL ou metadados de mapeamento de XML e WSDL para os artefatos de origem.

Utilize anotações para configurar ligações, cadeias de manipulador, nomes de portType, serviço e outros parâmetros WSDL. As anotações são utilizadas no mapeamento de Java para WSDL e esquema, e em tempo de execução para controlar como o tempo de execução de JAX-WS é processado e responde a chamadas de serviço da Web.

As anotações suportadas por JAX-WS são listadas na tabela a seguir. O destino para anotações é aplicável para estes objetos Java:

- tipos, como classe Java, enum ou interface
- métodos
- campos que representam variáveis de instância local em uma classe Java
- parâmetros em um método Java

2. Implantação

2.1. Serviço Standalone

São chamados stand Alone, ou stand-alone (literalmente "ficam em pé por si só") os programas completamente auto-suficientes: para seu funcionamento não necessitam de um software auxiliar, como um interpretador, sob o qual terão de ser executados.

Por exemplo, um programa Java é tipicamente compilado para bytecode e necessita de uma Java Virtual Machine para ser executado. Também este é o caso de um programa Perl, que vai depender de um interpretador.

Já um programa escrito em C ou C++, depois de compilado e tornado executável, poderia ser chamado standalone, uma vez que precisaria apenas de bibliotecas. As quais, inclusive poderiam ser anexadas fisicamente a ele, através de ligação estática.

2.2. Utilização de Endpoints

Service Endpoint Interface, ou Interface do Terminal de Serviço (SEI) é uma classe de interface Java que define os métodos a serem expostos como um serviço Web.

O JAX-WS (Java API for XML-Based Web Services) suporta dois tipos diferentes de implementações de terminal em serviço, a interface do terminal em serviço JavaBeans padrão e uma nova interface do Provedor para permitir que os serviços funcionem no nível de mensagem XML. Utilizando as anotações no terminal em serviço ou o cliente, é possível definir o terminal em serviço como um serviço da Web.

A tecnologia JAX-WS permite a implementação dos serviços da Web com base na interface do terminal em serviço JavaBeans padrão e uma nova interface do Provedor. Os terminais JavaBeans no JAX-WS são semelhantes às implementações de terminal na especificação JAX-RPC (Java API for XML-based RPC). Diferente do JAX-RPC, o requisito para um SEI (Service Endpoint Interface) é opcional para os serviços baseados em JavaBeans. Os serviços JAX-WS que não possuem um SEI associado são considerados portadores de um SEI implícito, enquanto os serviços

que possuem um SEI associado são considerados portadores de um SEI explícito. As interfaces do terminal em serviço requeridas pelo JAX-WS também são mais genéricas do que as interfaces de terminal em serviço requeridas pelo JAX-RPC. Com o JAX-WS, não é necessário que o SEI estenda a interface java.rmi.Remote conforme requerido pela especificação JAX-RPC.

O modelo de programação JAX-WS também alavanca o suporte para anotar as classes Java com metadados para definir um aplicativo do terminal em serviço como um serviço da Web e definir a maneira como um cliente pode acessar o serviço da Web. O JAX-WS suporta anotações baseadas na especificação Metadata Facility para Java Programming Language (JSR 175), na especificação Web Services Metadata para Java Platform (JSR 181) e anotações definidas pela especificação JAX-WS 2.0 (JSR 224), que inclui anotações JAXB (Java Architecture for XML Binding). Utilizando as anotações, a implementação do terminal em serviço pode descrever independentemente o serviço da Web sem precisar de um arquivo WSDL. As anotações podem fornecer todas as informações WSDL necessárias para configurar a sua implementação de terminal em serviço ou o cliente de serviços da Web. É possível especificar anotações na interface do terminal em serviço utilizada pelo cliente e pelo servidor ou na classe de implementação de serviço do lado do servidor.

Para desenvolver um serviço da Web JAX-WS, é necessário anotar a sua classe Java com a anotação javax.jws.WebService para terminais JavaBeans ou a anotação javax.jws.WebServiceProvider para um terminal do Provedor. Essas anotações definem a classe Java como um terminal em serviço da Web. Para um terminal JavaBeans, a interface do terminal em serviço ou a implementação do terminal em serviço é uma classe ou interface Java, respectivamente, que declara os métodos de negócios fornecidos por um serviço da Web específico. Os únicos métodos em um terminal JavaBeans que podem ser chamados por um cliente de serviços da Web são os métodos de negócios que são definidos na interface de terminal em serviço explícita ou implícita.

Todos os terminais JavaBeans precisam ter a anotação @WebService (javax.jws.WebService) incluída na classe bean. Se o bean de implementação de serviço também utilizar um SEI, esse endpoint deve ser referenciado pelo atributo endpointInterface nessa anotação. Se o bean de implementação de serviço não utilizar um SEI, o serviço será descrito pelo SEI implícito definido no bean.

O modelo de programação JAX-WS introduz a nova API do Provedor, javax.xml.ws.Provider, como uma alternativa para as interfaces de terminal em serviço. A interface do Provedor suporta uma abordagem mais orientada ao sistema de mensagens para serviços da Web. Com a interface do Provedor, é possível criar uma classe Java que implemente uma interface simples para produzir uma classe de implementação de serviço genérico. A interface do Provedor possui um método, o método invoke, que utiliza os genéricos para controlar os tipos de entrada e saída ao trabalhar com diversas cargas úteis de mensagem ou mensagens. Todos os terminais do Provedor devem ser anotados com a anotação @WebServiceProvider (javax.xml.ws.WebServiceProvider). Uma implementação de serviço não poderá especificar a anotação @WebService, se implementar a interface javax.xml.ws.Provider.

3. WSDL

Uma aplicação que acessa um serviço Web deve conhecer como e que tipo de informação deve trocar com ele. O Web Services Description Language (WSDL) é um documento, baseado em XML, que descreve este tipo de serviço formalmente, provendo um modo simplificado de representar suas informações. Este descreve formalmente os metadados de um Serviço Web. Através dele, clientes podem saber quais argumentos o serviço espera receber, quais são os dados retornados como resposta e qual o protocolo de comunicação ou transporte ele suporta [Newcomer 2002]. Além de utilizar estas informações para interagir com um serviço, o cliente pode ainda utilizá-las para buscar uma aplicação que melhor preencha seus requisitos.

Um documento WSDL é dividido logicamente em dois diferentes agrupamentos: as descrições concretas e abstratas. A descrição concreta é composta de elementos orientados à ligação física entre o cliente e o serviço. A descrição abstrata é composta de elementos orientados à descrição das capacidades do serviço (POTT; KOPACK, 2003). Os quatro elementos XML abstratos de um documento WSDL são:

wsdl:types: É usado para indicar que um tipo WSDL está sendo declarado. Normalmente, um tipo de dados definido pelo usuário onde é composto de tipos de dados primitivos ou complexos.

wsdl:messages: Elemento que descreve mensagens que deverão ser enviadas ou recebidas por esse serviço, contendo um conjunto de <wsdl:types>.

wsdl:operation: Análoga à uma chamada de método em uma linguagem de programação como Java, por exemplo. Contudo somente três tipos de mensagens são permitidas em uma operação: mensagens de entrada (Input), mensagens de saída (Output), e mensagens de falha (Fault).

wsdl:portType: Conjunto de todas as operações que um serviço Web pode realizar. Os três elementos XML concretos de um documento WSDL são (POTT; KOPACK, 2003):

wsdl:service: Elemento que contém referência a todas as portas que estão contidas no documento WSDL.

wsdl:port: Elemento que contém informações de IP e porta do serviço Web que está representado no WSDL.

wsdl:binding: Esse elemento tem dois propósitos: Primeiramente, interligar elementos concretos e abstratos no documento WSDL. O segundo propósito é prover um conjunto de informações como protocolo do serviço Web. Adicionalmente, o elemento raiz do documento WSDL é o <wsdl:definitions>. Nele, é especificado o namespace do documento, também chamado de targetnamespace.

4. SOAP

4.1. Definição

SOAP (Simple Object Access Protocol, em português Protocolo Simples de Acesso a Objetos) é um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída. Ele se baseia na Linguagem de Marcação Extensível (XML) para seu formato de mensagem, e normalmente baseia-se em outros protocolos da Camada de aplicação, mais notavelmente em Chamada de Procedimento Remoto (RPC) e Protocolo de Transferência de Hipertexto (HTTP), para negociação e transmissão de mensagens. SOAP pode formar a camada base de uma pilha de protocolos de web services, fornecendo um framework de mensagens básico sob o qual os serviços web podem ser construídos. Este protocolo baseado em XML consiste de três partes: um envelope, que define o que está na mensagem e como processá-la, um conjunto de regras codificadas para expressar instâncias do tipos de dados definidos na aplicação e uma convenção para representar chamadas de procedimentos e respostas.

Sua especificação define um framework que provê maneiras para se construir mensagens que podem trafegar através de diversos protocolos e que foi especificado de forma a ser independente de qualquer modelo de programação ou outra implementação específica. Por não se tratar de um protocolo de acesso a objetos, o acrônimo não é mais utilizado.

Geralmente servidores SOAP são implementados utilizando-se servidores HTTP, embora isto não seja uma restrição para funcionamento do protocolo. As mensagens SOAP são documentos XML que aderem a uma especificação fornecida pelo órgão W3C.

O primeiro esforço do desenvolvimento do SOAP foi implementar RPCs sobre XML.

4.2. RPC

Entre outras utilizações, SOAP foi desenhado para encapsular e transportar chamadas de RPC, e para isto utiliza-se dos recursos e flexibilidade do XML, sob HTTP. RPCs ou chamadas remotas de procedimento, são chamadas locais a métodos de objetos (ou serviços) remotos. Portanto, podemos acessar os serviços de um objeto localizado em um outro ponto da rede, através de uma chamada local a este objeto. Cada chamada ou requisição exige uma resposta. Processo de uma chamada RPC: Antes de serem enviadas pela rede, as chamadas de RPC (emitidas pela aplicação cliente) são encapsuladas (ou serializadas) segundo o padrão SOAP. O serviço remoto, ao receber a mensagem faz o processo contrário, desencapsulando-a e extraindo as chamadas de método. A aplicação servidora então processa esta chamada, e envia uma resposta ao cliente. O processo então se repete: a resposta é também serializada e enviada pela rede. Na máquina cliente, esta resposta é desencapsulada e é repassada para a aplicação cliente.

- Invocação de um método no Web Services
- Em geral chamadas síncronas
- Ligado à interface de programação
- Utilização de codificação SOAP para os dados
- Codificação SOAP (cada elemento tem uma indicação de seu tipo)

<nome xsi:type=”xsd:string”>JOAO</nome>
<idade xsi:type=”xsd:int”>30</idade>

Items obrigatórios para chamadas RPC

- A URI do destino alvo;
- O nome do método;
- Os parâmetros do método (requisição ou resposta);
- Uma assinatura do método – opcional;
- Um cabeçalho (header) - opcional.

Encoded:

Com estilo RPC Web Services, o método inteiramente especificado no arquivo WSDL e no corpo SOAP. Isto inclui os parâmetros enviados e os valores de retorno. Dessa maneira, você tem uma ligação bastante forte com esse schema. O Encoded indica que a mensagem SOAP irá conter as informações codificadas como xsi:type=":xsd int, xsi: type =" xsd:string ".

Literal:

Permanece estilo RPC, mas o WSDL terá sua codificação alterada, de modo que os dados são serializados de acordo com o esquema informado.

4.3. Document

- Envio de documentos XML para o Web Services
- Envio e ou recebimento de um documento XML
- Ligado a XML schemas que definem os documentos. Intermediários: Nós que podem processar uma mensagem SOAP no caminho do seu emissor para o seu receptor.

<nome>JOAO</nome>
<idade>30</idade>

Literal:

No estilo Document Web Services, o cliente envia documento XML padrão para o servidor. O aplicativo de servidor é responsável por mapear os objetos do servidor (parâmetros, chama o método, e assim por diante) e os valores nos documentos XML. O protocolo não impõe nenhuma restrição sobre a forma como o documento deve ser estruturado. O que significa que os dados são serializados de acordo com o esquema dado.

4.4. wsimport tool

A wsimport é uma ferramenta de linha de comando que processa um arquivo Web Services Description Language (WSDL) existente e gera os artefatos portáteis

necessários para o desenvolvimento de Java API para serviços Web baseados em XML (JAX-WS) aplicações Web Service.

A ferramenta de linha de comando wsimport apoia a abordagem top-down para o desenvolvimento de serviços Web JAX-WS. Quando você começa com um arquivo WSDL existente, usa a wsimport para gerar os artefatos portáteis JAX-WS necessários.

A ferramenta wsimport lê um arquivo WSDL existente e gera os seguintes artefatos portáteis:

- Service Endpoint Interface (SEI) - O SEI é a notação Java que representa o arquivo WSDL para o serviço Web. Essa interface é usada para implementar JavaBeans endpoints ou criar instâncias de *proxy client* dinâmicas.
- Extensão da classe javax.xml.ws.Service - Esta é uma classe gerada que estende a classe javax.xml.ws.Service . Essa classe é usada para configurar e criar tanto proxy dinâmico quanto as instâncias de despacho .
- *Required data beans*, incluindo qualquer arquitetura Java para XML Binding (JAXB) beans que são necessários para modelar os dados de Web Service.

Você pode empacotar os artefatos gerados em um arquivo Web (WAR) com o arquivo WSDL e documentos de esquema, juntamente com a implementação do endpoint para ser implantado (deployed).

5. Handlers

Um Handler permite que você envie e processe mensagens e objetos Runnable associados ao MessageQueue de uma thread. Cada instância Handler está associada a uma única thread e sua respectiva fila de mensagens. Quando você cria um novo Handler, ele é vinculado a thread / fila de mensagens do thread que o está criando - a partir desse ponto em diante, ele vai entregar mensagens e Runnables à fila de mensagens e executá-los como se eles saíssem da fila de mensagem.

Há dois usos principais para um Handler: (1) para agendar mensagens e Runnables para serem executados em algum momento no futuro, e (2) para enfileirar uma ação a ser executada em uma thread diferente da sua.

Mensagens de agendamento são realizadas com os métodos: post(Runnable), postAtTime(Runnable, long), postDelayed(Runnable, long), sendEmptyMessage(int), sendMessage(Message), sendMessageAtTime(Message, long) e sendMessageDelayed(Mensagem, long). As versões do post permitem que você enfileire objetos Runnable a serem chamados pela fila de mensagens quando são recebidos; as versões de sendMessage permitem enfileirar um objeto de mensagem contendo um conjunto de dados que serão processados pelo método handleMessage(Message) do Handler (que exige que você implemente uma subclasse de Handler).

Ao postar ou enviar para um Handler, você também permite que o item seja processado assim que a fila de mensagens está pronta para fazê-lo, ou especificar um atraso antes de ser processado ou o tempo absoluto para que ele seja processado. Os dois últimos permitem implementar timeouts, ticks e outros comportamentos baseados em sincronismo.

Quando um processo é criado para sua aplicação, a sua thread principal é dedicada à execução de uma fila de mensagens que se encarrega de gerenciar os objetos de aplicação de nível superior (atividades, receptores de broadcast, etc) e qualquer janela que eles criam. Você pode criar suas próprias threads, e comunicar de volta com a thread principal através de um Handler. Isso é feito chamando o mesmo método post ou sendMessage, mas a partir da sua nova thread. O Runnable ou mensagem será então agendado na fila de mensagens do Handler e processado quando apropriado.

6. Segurança

6.1. HTTPS

HTTPs é a combinação do protocolo HTTP com o SSL (Secure Sockets Layer). É a maneira mais comum atualmente de trafegar documentos via HTTP de maneira segura. Provê encriptação de dados, autenticação de servidor, integridade de mensagem e autenticação de cliente.

O protocolo HTTPS é utilizado, em regra, quando se deseja evitar que a informação transmitida entre o cliente e o servidor seja visualizada por terceiros, como, por exemplo, no caso de compras online. A existência na barra de tarefas de um cadeado (que pode ficar do lado esquerdo ou direito, dependendo do navegador utilizado) demonstra a certificação de página segura (SSL). A existência desse certificado indica o uso do protocolo HTTPS e que a comunicação entre o browser e o servidor se dará de forma segura.

6.2. Autenticação

A autenticação é um processo que busca verificar a identidade digital do usuário de um sistema, normalmente, no momento em que ele requisita um log in (acesso) em um programa ou computador. A autenticação normalmente depende de um ou mais "fatores de autenticação".

O termo "autorização" é muitas vezes confundido com o termo autenticação, mas apesar de serem relacionados, o significado de ambos é muito diferente. A autenticação é o processo que verifica a identidade de uma pessoa, por sua vez, a autorização verifica se esta pessoa possui permissão para executar determinadas operações. Por este motivo, a autenticação sempre precede a autorização.

Os fatores de autenticação para humanos são normalmente classificados em três casos:

- Aquilo que o usuário é (impressão digital, padrão retinal, sequência de DNA, padrão de voz, reconhecimento de assinatura, sinais elétricos unicamente identificáveis produzidos por um corpo vivo, ou qualquer outro meio biométrico).

- Aquilo que o usuário tem (cartão de identificação, security token, software token ou telefone celular)
- Aquilo que o usuário conhece (senha, frase de segurança, PIN)

Frequentemente é utilizada uma combinação de dois ou mais métodos. A Secretaria da Receita Federal, por exemplo, pode requisitar um certificado digital (o que se possui) além da senha (o que se sabe) para permitir o acesso a uma declaração de imposto de renda, neste caso o termo "autenticação de dois fatores" é utilizado.

7. Referências Bibliográficas

ALONSO, Gustavo; CASATI, Fabio; KUNO, Harumi; MACHIRAJU, Vijay; VERLAG, Springer. *Web Services – Concepts, Architectures and Applications.* Artigo disponível em: http://www.inf.ethz.ch/personal/alonso/Web-book/Chapter-5.pdf. Acesso em: 14/10/2013.

WEBOPEDIA. *Web Services.* Artigo disponível em: http://www.webopedia.com/TERM/W/Web_Services.html. Acesso em: 14/10/2013.

MKYONG. *JAX-WS Tutorial.* Artigo disponível em: http://www.mkyong.com/tutorials/jax-ws-tutorials/. Acesso em: 14/10/2013.

SPINOLA, Eduardo. *Desenvolvendo Web Services utilizando JAX-WS.* Artigo disponível em: http://www.devmedia.com.br/desenvolvendo-web-services-utilizando-jax-ws/2374. Acesso em: 14/10/2013.

MEDEIROS, Higor. *Desenvolvendo e Usando Web Services em Java.* Artigo disponível em: http://www.linhadecodigo.com.br/artigo/3654/desenvolvendo-e-usando-web-services-em-java.aspx. Acesso em: 17/10/2013.

IBM. Desenvolvendo uma implementação de Terminal em Serviço para Aplicativos JAX-WS com Anotações. Acesso em: 17/10/2013. Manual disponível em: http://pic.dhe.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=%2Fcom.ibm.websphere.wsfep.multiplatform.doc%2Finfo%2Fae%2Fae%2Ftwbs_devjaxwsendpt.html.

PASTORE, Pablo. *HTTP, HTTPs e TLS Uma breve introdução.* Artigo disponível em: http://www.gta.ufrj.br/grad/03_1/http/. Acesso em: 17/10/2013.

BACCARO, Marco. *SOAP e RPC.* Artigo disponível em: http://marcobaccaro.wordpress.com/2010/06/29/soap-e-rpc/. Acesso em: 17/10/2013.

JAX-WS. Disponível em: https://jax-ws.java.net/. Acesso em: 17/10/2013. http://marcobaccaro.wordpress.com/2010/06/29/soap-e-rpc/.

Handler. Disponível em: http://developer.android.com/reference/android/os/Handler.html. Acesso em: 25/10/2013.

IBM. Disponível em: http://pic.dhe.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=%2Fcom.ibm.websphere.wsfep.multiplatform.doc%2Finfo%2Fae%2Fae%2Frwbs_wsimport.html. Acesso em: 25/10/2013