Web Services Invocation Framework: Overview

1 Overview

WSIF stands for the Web Services Invocation Framework. It supports a simple Java API for invoking Web services, no matter how or where the services are provided. The framework allows maximum flexibility for the invocation of any WSDL-described service.

In the WSDL specification, Web service binding descriptions are *extensions* to the specification. So the SOAP binding, for example, is one way to expose the abstract functionality (*and there could be others*). Since WSIF mirrors WSDL very closely, it also views SOAP as just one of several ways you might wish to expose your software's functionality. WSDL thus becomes a normalized description of software, and WSIF is the natural client programming model.

The WSIF API allows clients to invoke services focusing on the abstract service description - the portion of WSDL that covers the port types, operations and message exchanges without referring to real protocols. The *abstract invocations* work because they are backed up by protocol-specific pieces of code called *providers*. A provider is what conducts the actual message exchanges according to the specifics of a particular protocol - for example, the SOAP provider that is packaged with WSIF uses a specific SOAP engine like Axis to do the real work.

The decoupling of the abstract invocation from the real provider that does the work results in a flexible programming model that allows dynamic invocation, late binding, clients being unaware of large scale changes to services - such as service migration, change of protocols, etc. WSIF also allows new providers to be registered dynamically, so you could enhance your client's capability without ever having to recompile its code or redeploy it.

Using WSIF, WSDL can become the centerpiece of an integration framework for accessing software running on diverse platforms and using widely varying protocols. The only precondition is that you need to describe your software using WSDL, and include in its description a binding that your client's WSIF framework has a provider for. WSIF defines and comes packaged with providers for local java, EJB, JMS, and JCA protocols. That means you can define an EJB or a JMS-accessible service directly as a WSDL binding and access it transparently using WSIF, using the same API you would for a SOAP service or even a local

java class.

TODO: Put a picture showing WSIF client with pluggable providers to access service using different protocols.

2 WSIF Structure

In WSDL a binding defines how to map between the abstract PortType and a real service format and protocol. For example, the SOAP binding defines the encoding style, the SOAPAction header, the namespace of the body (the targetURI), and so forth.

WSDL allows there to be multiple implementations for a Web Service, and multiple Ports that share the same PortType. In other words, WSDL allows the same interface to have bindings to for example, SOAP and IIOP.

WSIF provides an API to allow the same client code to access any available binding. As the client code can then be written to the PortType it can be a deployment or configuration setting (or a code choice) which port and binding it uses.

WSIF uses 'providers' to support these multiple WSDL bindings. A provider is a piece of code that supports a WSDL extension and allows invocation of the service through that particular implementation. WSIF providers use the J2SE JAR service provider specification making them discoverable at runtime.

Clients can then utilize any new implementations and can delegate the choice of port to the infrastructure and runtime, which allows the implementation to be chosen on the basis of quality of service characteristics or business policy.

3 WSDL bindings for EJBs, JMs, JCA...

WSIF defines additional binding extensions so that EJBs, local java classes, software accessible over message queues using the JMS API and software that can be invoked using the Java Connector architecture can also be described in WSDL. WSIF is packaged with providers that allow transparent invocation of such software given the corresponding WSDL description. Here are the documents that describe these bindings:

- Local java binding extensions for WSDL
- EJB binding extensions for WSDL
- JMS binding extensions for WSDL
- JCA binding extensions for WSDL