Welcome to WSIF: Web Services Invocation Framework

1 Introduction

The Web Services Invocation Framework (WSIF) is a simple Java API for invoking Web services, no matter how or where the services are provided. Please refer to the release notes before you proceed with using WSIF.

WSIF enables developers to interact with abstract representations of Web services through their WSDL descriptions instead of working directly with the Simple Object Access Protocol (SOAP) APIs, which is the usual programming model. With WSIF, developers can work with the same programming model regardless of how the Web service is implemented and accessed.

WSIF allows stubless or completely dynamic invocation of a Web service, based upon examination of the meta-data about the service at runtime. It also allows updated implementations of a binding to be plugged into WSIF at runtime, and it allows the calling service to defer choosing a binding until runtime.

Finally, WSIF is closely based upon WSDL, so it can invoke any service that can be described in WSDL.

2 Why should I use WSIF?

What does all this enable? Imagine your complicated Enterprise software system consisting of various pieces of software, developed over a period of tens of years - EJBs, legacy apps accessed using Java's connector architecture, SOAP services hosted on external servers, old code accessed through messaging middleware. You need to write software applications that use all these pieces to do useful things; yet the differences in protocols, mobility of software, etc. comes in the way.

The software you use moves to a different server, so your code breaks. The SOAP libraries you use change - say for example you moved from using Apache SOAP to Apache Axis - so your code breaks since it uses a now deprecated SOAP API. Something that was previously accessible as an EJB is now available through messaging middleware via JMS - again, you need to fix the code that uses the software. Or lets suppose you have an EJB which is offered as a SOAP service to external clients. Using SOAP obviously results in a performance penalty as compared to accessing the EJB directly. Of course, SOAP is a great baseline protocol for platform andlanguage independence, but shouldn't java clients be able to take advantage of the fact that the software they are accessing is really an EJB? So your java customers pay a performance penalty since you have to use SOAP for to accomodate you non-java clients.

WSIF fixes these problems by letting you use WSDL as a *normalized description* of disparate software, and allows you to access this software in a manner that is independent of protocol or location. So whether it is SOAP, an EJB, JMS (or potentially .NET and other software frameworks), you have an API centered around WSDL which you use to access the functionality. This lets you write code that adapts to changes easily. The separation of the API from the actual protocol also means you have flexibility - you can switch protocols, location, etc. without having to even recompile your client code. So if your an externally available SOAP service becomes available as an EJB, you can switch to using RMI/IIOP by just changing the service description (the WSDL), without having to make any modification in applications that use the service. You can exploit WSDL's extensibility, its capability to offer multiple bindings for the same service, deciding on a binding at runtime, etc.

You will find more details about WSIF and its architecture in the overview.

3 How can I contribute to WSIF?

You can contribute to WSIF by participating in discussions on the wsif-user and wsif-dev mailing lists. If you find an issue or bug that interests you, feel free to download WSIF's source code and work on it. You will find instructions for accessing the source code here.

You can find a list of outstanding bugs in ASF JIRA.