Bare JAX-WS

Paul Glezen, IBM

Abstract

This document is incomplete; it is made available for review purposes only.

This document is a member of the Bare Series of WAS topics distributed in both stand-alone and in collection form.

Table of Contents

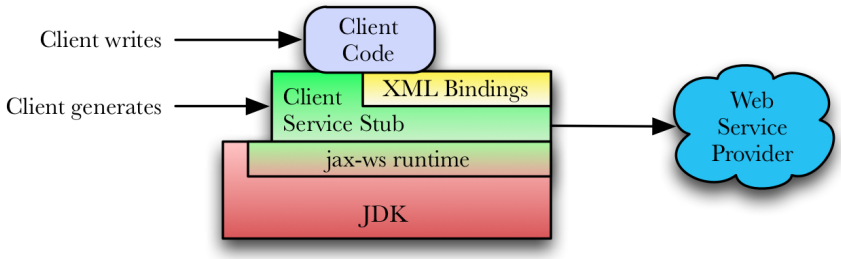
1. Bare JAX-WS Web Services 1
    1.1. JAX-WS Clients 1
2. References 2

1. Bare JAX-WS Web Services

1.1. JAX-WS Clients

Java **A**PI for **X**ML - **W**eb **S**ervices (JAX-WS) is a specification that addresses Java web services development. For web service clients, this amounts to generating Java client service stubs and XML bindings for use in invoking the web service. Figure 1 illustrates how these components fit together. The green and yellow components are generated from the WSDL. The green is the service stub. Its objects make the remote call to the service provider. The yellow represents the XML-type-to-Java-type bindings (or simply XML bindings). The XML bindings are actually addressed by a separate specification called **J**ava **A**PI for **X**ML **B**indings (JAXB) that is referenced by JAX-WS. This is part of what makes JAX-WS so much more powerful than its predecessor, **J**ava **A**PI for **X**ML - **R**emote **P**rocedure **C**all (JAX-RPC).

Figure 1. JAX-WS Overview



Another recent improvement is the inclusion of the JAX-WS runtime in the Java 6 SE (standard edition). One no longer needs to reference special "thin client" libraries to make web service clients run. The generated bindings run directly against a Java 6 or later runtime. And not only are the runtime classes available in the JRE, the *wsimport* utility, responsible for generating the bindings from the WSDL, is part of the JDK on any platform. No special IDEs or tools are needed.

The XML bindings are Java classes that map to the XML schema types defined in the WSDL. (One says that a Java type is bound to the XML schema type.) These types play the role of parameters for the service invocation. The invocation functions themselves are methods on the service stub objects. The bindings objects are passed as parameters to the service objects.

The generated service and binding objects tie into the JAX-WS runtime. This may be part of the JDK as in the diagram above. Or it may be implemented by a vendor such as Apache CXF or IBM WebSphere Application Server. In any case, it is responsible for

- marshaling the data structures into a serialized XML stream, and
- implementing the network protocol to transport the XML stream to the server.

Finally, the client code is the consumer of the service. It issues the request to the service stub and does whatever it requires with the result.

Web service clients may be *managed* or *unmanaged*. Managed clients are typically associated with an application server. The client is managed in the sense that aspects of its configuration are controllable through the administrative capabilities of the application server. References to the service stub objects are usually retrieved from JNDI. Unmanaged clients, also known as *thin clients*, do not rely on any underlying application server structure for configuration. Their service client proxy objects are directly instantiated. Their configuration is usually done by setting properties on the service stub instances. There is nothing wrong with running a thin client inside an application server. It simply won't benefit from enterprise manageability features.

1.1.1. Thin Clients

A thin client is one that does not expect the presense of any application server infrastructure. That's not to say a thin client can't run within an application server container. It simply doesn't depend on the container for resources or initialization.

Generating a thin client is easy and requires nothing more than a valid WSDL and JDK 6. The command for generating the JAX-WS bindings is **wsimport**.

2. References

[1] WAS 8.0 Info Center, IBM. Online: http://pic.dhe.ibm.com/infocenter/wasinfo/v8r0/

[2] WS-SecurityPolicy 1.2 Specification, December, 2006. OASIS. Online: http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512