WsmArchitecture

Overview

The Beehive WSM implementation is split into two parts – build-time and runtime. This split exists in order to enable support for multiple web service execution environments including Axis, Geronimo, and so on.

Note, the architecture descirbed here has not yet been fully realized in the WSM implementation.

WSM Build-time

The build-time part of WSM consists of an annotation processor and an object model that represents a web service. In general, the processing of a WSM web service can be thought of as a pipeline that executes in the following order:

- JSR-181 annotation processing. This stage basically implements the checks in the JSR-181 specification including semantically validating the annotations, checking an implementation bean against a service interface, and validating the contract between an implementation bean or service interface a specified WSDL.
- WSM JavaBean creation. This step converts the metadata representation of the web service into a JavaBean model that can be used for artifact generation.
- Artifact generation. For some web service deployment environments, source artifacts (Java, XML, etc) may need to be produced at build time; this is done by writing a runtime-specific code generator that can be passed to the annotation processor using a "-A" option.

WSM Runtime

Currently, WSM only supports the Axis 1.x web service runtime. We welcome contributions that extend this support to additional web service containers, though it might be worth waiting until after a WSM beta so that the APIs and extension points have stabilized.

Axis 1.x Support (current, 12/29/2005)

The WSM implementation supports the Axis 1.x runtime by plugging an Axis-specific artifact generator into the "Artifact generation" build-time step. Currently, this implementation simply serializes the WSM JavaBean model into a binary Java object along side the generated .class file for the web service implementation. Then, this serialized file is de-serialized at runtime and used to custom-wire an Axis SOAPService. This implementation is difficult to test and debug because the description of the web service is binary.

Axis 1.x Support (proposed)

It has been proposed that the Axis 1.x support for WSM be revamped to hvae the build-time generate a .wsdd file which can be used to deploy and further customize web services executed on Axis. This change would shift the WSM implementation away from the serialized JavaBean and onto a more transparent model. Initially, this would remove drop-in support for web services and would require a manual deployment step of a web service's associated .wsdd file. Support for hot redeployment of a web service would be re-implemented later. The initial benefit of this model would be that WSM could leverage more of Axis's infrastructure for deploying services in this fashion, thus significantly simplifying the WSM server-side code for Axis 1.x. Feedback welcome on this proposal...feel free to add thoughts here.