Java Web Service Files

Table of contents

1 What is a Metadata Annotation?....2
2 WSDL Mapping Annotations....2
2.1 @WebService....2
2.2 @WebMethod....3
2.3 @WebParam....3
2.4 Separating the Public and Implementation Names of Methods and Parameters....4
2.5 @WebResult....5
3 Binding Annotations....5
3.1 @SOAPBinding....5
4 Security Annotations....5
4.1 @SecurityIdentity....5
4.2 @SecurityRoles....6
5 Handler Annotations....6
5.1 @SOAPMessageHandlers....6

This topic explains the basic techniques for exposing a Java class as a web service using the Beehive web service metadata annotations.

1. What is a Metadata Annotation?

A metadata annotation is a property setting on some piece of Java code. That's where the "meta" in "metadata annotation" comes from: metadata annotations overlay the Java code they modify.

Metadata annotations can decorate either (1) Java classes or (2) Java methods. Syntactically speaking, a class-decorating annotation appears as follows:

```
@WebService public class MyWebService {
                                    ... }
```

A method-decorating annotation appears as follows:

```
@WebService public class MyWebService {
                                    @WebMethod public String sayHelloWorld() {
                                    return "Hello world!"; } }
```

2. WSDL Mapping Annotations

These annotations let you control how your Java source implementation is exposed through the web service's WSDL file. More generally: these annotations let you control which parts of the WSDL are associated with which Java implementation code.

2.1. @WebService

To specify that a Java class should be exposed as a web service, decorate the class signature with the `@WebService` annotation. The imported class `javax.jws.WebService` is the backing class for the annotation.

```
import javax.jws.WebService; @WebService
                                    public class MyWebService { ... }
```

Attributes on the annotation determine properties on the web services WSDL. For example, the namespace advertised by the WSDL (and used for the SOAP XML documents generated by the web service) is specified by the `targetNamespace` attribute.

```
import javax.jws.WebService; @WebService(
                                    targetNamespace =
                                    http://www.openuri.org/my/web/service/wsdl
) public
                                    class MyWebService { ... }
```

The WSDL will contain the namespace referenced as follows:

```
 <wsdl:definitions targetNamespace="targetNamespace =
                                    http://www.openuri.org/my/web/service/wsdl">
```

```
                                    <wsdl:types> <schema
elementFormDefault="qualified"
                                    targetNamespace="targetNamespace =
http://www.openuri.org/my/web/service/wsdl">
                                    <element name="sayHelloWorld"> ...
```

SOAP documents generated by the this web service will reference the namespace as so:

```
[todo]
```

2.2. @WebMethod

The `@WebMethod` annotation decorates a method: it means that the method should be exposed as a web service method, i.e., a method that is invokable through SOAP messages.

```
import javax.jws.WebService;
import javax.jws.WebMethod;

@WebService(
    targetNamespace="targetNamespace =
http://www.openuri.org/my/web/service/wsdl"
)
public class MyWebService
{
    @WebMethod
    public String sayHelloWorld()
    {
        return "Hello world!";
    }
}
```

[todo]

2.3. @WebParam

The `@WebParam` annotation marks a parameter of a web-exposed method (a method exposed by the `@WebMethod` annotation).

```
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

@WebService(
    targetNamespace="targetNamespace =
http://www.openuri.org/my/web/service/wsdl"
)
public class MyWebService
{
    @WebMethod
    public String sayHelloWorld( @WebParam String greetee )
    {
        return "Hello " + greetee + "!";
    }
}
```

2.4. Separating the Public and Implementation Names of Methods and Parameters

Changing the name displayed in the WSDL for methods and parameters...

These features make it possible to change the underlying Java source implementation for a pre-existing WSDL, or to change the WSDL with only minor changes to the underlying Java source implementation.

To display a different method name in the WSDL, use the `@WebParam( operationName="[opName]" )` attribute.

```
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

@WebService(
    targetNamespace="targetNamespace =
http://www.openuri.org/my/web/service/wsdl"
)
public class MyWebService
{
    @WebMethod(
            operationName="getGreeting"
        )
    public String sayHelloWorld( @WebParam String greetee )
    {
        return "Hello " + greetee + "!";
    }
}
```

The WSDL will display the operation `getGreeting` to the client.

```
<wsdl:portType name="MyWebService">
  <wsdl:operation name="getGreeting">
    <wsdl:input message="impl:getGreetingRequest"
name="getGreetingRequest"/>
    <wsdl:output message="impl:getGreetingResponse"
name="getGreetingResponse"/>
  </wsdl:operation>
</wsdl:portType>
```

To display a different name for the method parameter, use the `@WebParam( name="[name]" )` attribute.

```
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

@WebService(
    targetNamespace="targetNamespace =
```

```
http://www.openuri.org/my/web/service/wsdl"
)
public class MyWebService
{
    @WebMethod(
            operationName="getGreeting"
        )
    public String sayHelloWorld( @WebParam( name="greetWho" ) String
greetee )
    {
        return "Hello " + greetee + "!";
    }
}
```

The relevant part of the WSDL is shown below.

```
<element name="getGreeting">
  <complexType>
    <sequence>
      <element name="greetWho" type="xsd:string"/>
    </sequence>
  </complexType>
</element>
```

Changing the method invoked...[todo: doable?]

2.5. @WebResult

[todo]

3. Binding Annotations

These annotations let you control the network protocols and message formats supported by your web service.

3.1. @SOAPBinding

[todo]

4. Security Annotations

These annotations let you request authentication and authorization services from the web service's deployment container.

4.1. @SecurityIdentity

[todo]

4.2. @SecurityRoles

[todo]

5. Handler Annotations

These annotations let you intercept incoming and outgoing SOAP messages for additional processing before and after the execution of web service's business processes.

5.1. @SOAPMessageHandlers

[todo]