Table of Contents

Java and SOAP By Robert Englander ISBN: 0-596-00175-4 Publisher: O'Reilly
Print Publication Date: 2002/05/01

Chapter 7. Faults and Exceptions

The difference between good software and bad software is often the way in which errors and problems are handled. It's much easier to deal with processes when everything is working properly than it is to deal with failures. Structured and object-oriented programming provide lots of techniques for handling errors. Methods often return a special value, like `null` or `-1`, to indicate an error of some kind. In Java, methods frequently throw an exception indicating that something unusual has occurred. Good software is written to expect certain types of errors, and is prepared to take appropriate action. In SOAP terminology, these unusual, or exceptional, circumstances are called *faults*. Faults occur whenever a service method is not able to process input parameters and return results properly. There are endless reasons why this may occur. Common problems that result in faults are bad method parameter values, back-end problems, and improperly formatted SOAP request messages. I'm sure you can think of plenty of others, and have probably had to write code to deal with them. In this chapter we'll look at the mechanisms for generating and handling faults in SOAP.

7.1. Throwing Server-Side Exceptions in Apache SOAP

The most common way for a service to generate a fault is to throw an exception. Apache SOAP has a mechanism for handling exceptions thrown from the Java class methods that implement service methods. The information in the exception is used to generate corresponding SOAP faults to be sent back to the caller.

The contents of a SOAP fault were covered back in Chapter 2. In that section we discovered that four fault codes could be used in a fault. One of those possibilities is `SOAP-ENV:Server`, which indicates that a problem occurred while processing the message that was not related to the contents of the request. This fault code could be used if, for example, the back-end server needed to properly process the message was not available. Let's create a simple service that generates the simplest of faults so that we can see the basic mechanism used in Apache SOAP. Here's the `javasoap.book.ch7.SampleFaultService` class that implements the service `urn:SampleFaultService`:

7.2. Creating a Fault Listener in Apache SOAP

To populate a fault with a `detail` element, we need to create a new class for fault listening by implementing the `org.apache.soap.server.SOAPFaultListener` interface. This interface defines a method called `fault( )`, which is called by the Apache SOAP framework when a fault needs to be generated. The only parameter passed to this method is an instance of `org.apache.soap.server.SOAPFaultEvent`,[1] which carries an instance of `org.apache.soap.Fault` and the `SOAPException` that was thrown by the service method. The fault object contains methods for getting and setting the fault code, fault string, fault actor, and details. This is the object we want to manipulate to provide detail information.

[1] If you're familiar with the JavaBeans patterns for events you should recognize this naming convention. The event that has taken place is a `SOAPFault`. The listener interface is named by appending `Listener` to the event name, and the object carrying the event information is named by appending `Event`.

But before we create the fault listener, we need a mechanism for providing it with the detail data. We don't have a direct path from our service methods to the fault listener, but we can take advantage of the exception mechanism. Since we've seen that the `SOAPException` that gets thrown by the service method is available to the fault listener, we can inherit from `SOAPException` and create a new class that meets our needs. But before we do that, we need a class to carry our detail information. Let's use a `java.util.Hashtable`. That way we can set up any number of attribute/value pairs to represent the fault detail information. With all this in mind, let's create a new exception type with a property called `Detail` that's an instance of `Hashtable`.

7.3. Throwing and Catching Exceptions in GLUE

The concepts covered for faults and exceptions in Apache SOAP are similar to those used in GLUE, although some of the techniques are different. Actually, GLUE has a few ways of dealing with exceptions, so if you're interested in all the possibilities, I encourage you to read the GLUE user's guide; we're going to cover only one technique here.

Let's skip the basics and get right into an example that returns faults with a detail section. GLUE includes its own SOAP exception class called `electric.net.soap.SOAPException`. It's not the same as the Apache `SOAPException`, so don't confuse the two. GLUE's `SOAPException` allows you to set the fault code, fault string, fault actor, and details, and has several constructors using a variety of combinations of these fault element values. Therefore, you don't have to write a fault handler to get the detail information into the fault; just throw an instance of `SOAPException` with the appropriate values. Let's create a service called `urn:AnotherFaultService` and implement it with the `AnotherFaultService` class: