Table of Contents

Java and SOAP By Robert Englander ISBN: 0-596-00175-4 Publisher: O'Reilly
Print Publication Date: 2002/05/01
Prepared for Douglas Looms, Safari ID: dlooms@erols.com
User number: 328147 Copyright 2007, Safari Books Online, LLC.
This PDF is exclusively for your use in accordance with the Safari Terms of Service. No part of it may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher. Redistribution or other use that violates the fair use priviledge under U.S. copyright laws (see 17 USC107) or that otherwise violates the Safari Terms of Service is strictly prohibited.

Chapter 4. RPC-Style Services

Although SOAP is not limited to a particular style of distributed computing, it lends itself to a remote procedure call (RPC) model. This is only what you would expect; according to the SOAP specification, one of SOAP's design goals is to encapsulate and exchange RPC calls. This approach maps very nicely to Java programming because method calls on Java objects can be easily translated to RPC calls. We'll start this chapter by looking at the structure of SOAP RPC request and response messages. From there we'll implement some RPC services in Java and deploy those services in both Apache SOAP and in GLUE. And, of course, we'll write some Java code to make use of those services.

Copyright Safari Books Online #328147

4.1. SOAP RPC Elements

Creating a SOAP RPC request uses the SOAP structure and encoding described in Chapter 2 and Chapter 3. No new XML or data encoding styles are needed for RPC. Let's take a look at what's required to represent an RPC method call in SOAP:

- The target object
- Method name
- Method parameters
- SOAP header data

4.2. A Simple Service

Let's design a simple service called `CallCounterService`. This service keeps track of the number of method calls it receives. It's not exactly useful by itself, but it gives us a chance to get a service up and running to see how services are deployed and how the service activation models work. We'll place the classes for this example in the package `javasoap.book.ch4`. Make sure to create the appropriate directory for this package on your system and make it available on the classpath used by your server and client systems.

Here's the source code for the `MethodCounter` class, which implements the `CallCounterService`. Note that the Java class name does not have to be the same as the service name.

Java and SOAP By Robert Englander ISBN: 0-596-00175-4 Publisher: O'Reilly
Print Publication Date: 2002/05/01
Prepared for Douglas Looms, Safari ID: dlooms@erols.com
User number: 328147 Copyright 2007, Safari Books Online, LLC.
This PDF is exclusively for your use in accordance with the Safari Terms of Service. No part of it may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher. Redistribution or other use that violates the fair use priviledge under U.S. copyright laws (see 17 USC107) or that otherwise violates the Safari Terms of Service is strictly prohibited.

4.3. Deploying the Service

Service deployment is not part of the SOAP specification, so each implementation has its own deployment procedure. We'll look at service deployment using two SOAP implementations: Apache SOAP and GLUE.

4.3.1. Deploying with Apache SOAP

In Apache SOAP, you must create a *deployment descriptor*, which is an XML file that contains information about the service and the Java class that implements the service. Let's take a look at a deployment descriptor for `CallCounterService`, implemented by the Java class `javasoap.book.ch4.MethodCounter`:

4.4. Writing Service Clients

In the next few sections, we'll write some Java code that invokes methods on the services we've exposed in both Apache SOAP and GLUE. For now, we won't mix and match technologies — we'll use Apache SOAP APIs to call an Apache SOAP server, and GLUE APIs to call a GLUE server.

So what about interoperability? One of the most important aspects of SOAP as a wire protocol is that your choice of implementation should not prohibit you from communicating successfully with other SOAP implementations. However, there are still some problems with SOAP interoperability. Some technologies are better at it right now than others, and in some cases you have to jump through a few hoops to make different SOAP technologies communicate properly with each other. So let's put off this subject until Chapter 9, where we'll cover these issues in detail.

4.5. Deploying with Request-Level Scope

Up until now, all the examples we've created have been deployed using application-level scope (service activation). In this mode, a single instance of the implementing service class handles all of the method invocations. Let's change the deployment to request-level scope. With request-level scope, a new instance of the service object is created for each method invocation, and that object is destroyed when the invocation is complete.

First we'll change the example running under Apache SOAP by modifying its deployment descriptor. The only change required is to set the `scope` attribute of the `provider` element to `Request`; the rest of the deployment descriptor remains exactly the same:

4.6. Deploying with Session-Level Scope

Apache SOAP provides for session management by passing cookies via the HTTP headers. Earlier in this chapter, we saw an HTTP response from the server that included HTTP header entries called `Set-Cookie` and `Set-Cookie2`. The client application uses these

Java and SOAP By Robert Englander ISBN: 0-596-00175-4 Publisher: O'Reilly
Print Publication Date: 2002/05/01
Prepared for Douglas Looms, Safari ID: dlooms@erols.com
User number: 328147 Copyright 2007, Safari Books Online, LLC.
This PDF is exclusively for your use in accordance with the Safari Terms of Service. No part of it may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher. Redistribution or other use that violates the fair use priviledge under U.S. copyright laws (see 17 USC107) or that otherwise violates the Safari Terms of Service is strictly prohibited.

cookies if it needs to make subsequent method invocations against the same session as the original request. The Apache SOAP API uses a simple technique for handling this. `org.apache.soap.transport.http.SOAPHTTPConnection` includes a method called `setMaintainSession( )` that takes a single `boolean` parameter. This parameter, when set to `true`, tells the connection object to maintain the current session. The connection object implements this by keeping track of the cookies and sending them back to the server when the next method invocation takes place.

Go ahead and edit the deployment descriptor, setting the `scope` attribute to `Session`. Now run the client program again. You'll get the following output:

4.7. Passing Parameters

So far, the examples in this chapter have used methods that don't contain any parameters. We'll now spend a little time looking at how parameters are passed. Let's create a service called `urn:StockPriceService`, implemented by the `javasoap.book.ch4.StockPrice` class. The `getPrice( )` method takes parameters for the stock symbol as well as the currency. The method always returns the float value of 75.33 — not a very interesting stock, but at least it won't go down. Seriously though, we're just interested in how parameters are passed. The details of accessing a database or data feed to find a stock price are, for the moment, left to you.

```
package javasoap.book.ch4;
public class StockPrice {
   public float getPrice(String stock, String currency) {
      float result;
      // determine the price for stock and return it
      // in the specified currency
      result = (float)75.33;
      return result;
   }
}
```


Java and SOAP By Robert Englander ISBN: 0-596-00175-4 Publisher: O'Reilly
Print Publication Date: 2002/05/01
Prepared for Douglas Looms, Safari ID: dlooms@erols.com
User number: 328147 Copyright 2007, Safari Books Online, LLC.
This PDF is exclusively for your use in accordance with the Safari Terms of Service. No part of it may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher. Redistribution or other use that violates the fair use priviledge under U.S. copyright laws (see 17 USC107) or that otherwise violates the Safari Terms of Service is strictly prohibited.