Dealing with SOAP Web Services

Objective

In this chapter, you will examine how you can interact with SOAP web services in Titanium.

Contents

In some enterprise settings, "Simple" Object Access Protocol (SOAP) is the format for XML data returned by a web service. SOAP web services are very much possible in Titanium, though they are the least simple option.

Avoid SOAP if you can

Although you can use SOAP web services (this may be your only option, especially if you are dealing with a 3rd party or legacy interface), it is recommended to avoid using SOAP web services in a Titanium application. SOAP retains the disadvantages of XML:

- The overhead of XML over the wire
- The need to translate from an XML format to a JavaScript object format

And compounds them because SOAP is even more verbose (much more XML being transported over the wire), and the results are even more difficult to parse. Some programming languages provide high-level tools, WSDL parsers, and other mechanisms to work around the complexities of a SOAP format, but JavaScript has historically never had any of those types of tools. This remains the case today, and as such, there are very few high-level libraries and tools to support SOAP in JavaScript.

The low-tech approach

The approach taken by a number of Titanium projects we have worked on is to stay very low-tech and POST manually-created SOAP envelopes (XML strings) to a web service endpoint. If you understand how HTTP and SOAP work together, you can manually construct a SOAP envelope to send to your server, with the appropriate contents:

```
var client = Ti.Network.createHTTPClient();
 client.onload = function() {
 var doc = this.responseXML.documentElement;
 //manually parse the SOAP XML document
};

var soapRequest = "<?xml version=\"1.0\" encoding=\"UTF-8\"?> \n" +
"<SOAP-ENV:Envelope xmlns:SOAP-ENV=\"http://schemas.xmlsoap.org/soap/envelope/\" \n" +
"xmlns:SOAP-ENC=\"http://schemas.xmlsoap.org/soap/encoding/\" \n" +
"xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" \n" +
"xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\" \n" +
"xmlns:xs=\"http://www.w3.org/2001/XMLSchema\" \n" +
"xmlns:wsse=\"http://schemas.xmlsoap.org/ws/2002/12/secext\"> \n" +
"<SOAP-ENV:Body id=\"_0\"> \n" +
"<GetUserDetailsReq> \n" +
"<Request> \n" +
"<SessionToken xsi:type=\"ns:IVRSessionToken\">XXXX</SessionToken> \n" +
"</Request> \n" +
"</GMGetUserDetailsReq> \n" +
"</SOAP-ENV:Body> \n" +
"</SOAP-ENV:Envelope>";

client.open('POST', 'https://someserver.com/someendpoint.asmx');
client.send({xml: soapRequest});
```

Bear in mind the above SOAP envelope is completely made up and derived from another service. In order to use your own SOAP web services in this fashion, you will need to understand what the contents of a SOAP request to your server actually looks like as an HTTP request. Here, other third party tools can help, particularly ones that let you inspect the raw HTTP requests and responses for your web service. On the Mac, you might consider using SOAP Client. The Eclipse Web Tools project also has a bit of SOAP oriented tooling.

Summary

In this chapter, you learned that you can interact with SOAP web services in Titanium, but that SOAP involves extra overhead and larger data payloads compared to JSON or XML.