XML-RPC

The Real Simple Access
Object Protocol

DI Siegfried GOESCHL
Motivation

- You need some client/server communication ?!
- You cannot spend a few megabucks for RPC calls ?!
- You don't want to spend a few weeks to get something running ?!
- You have a heterogenous environment in terms of operating systems and programming languages ?!
- You might have a few firewalls in your way ?!
What is XML-RPC ?!

- It's a specification and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet.

- It's remote procedure calling using HTTP as the transport and XML as the encoding.

- XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned.
The XML-RPC Story

- UserLand used internally a protocol called „RPC“
- Dave Winer started to work with Microsoft on something called „SOAP“ based on „RPC“
- The specification forked and „XML-RPC“ was published
- XML-RPC was used for UserLand’s Frontier (April, 1998)
- XML-RPC gained quite a bit of popularity
- Hannes Wallnöfer implemented an XML-RPC library in Java (Helma XML-RPC)
- Helma XML-RPC is part of the TURBINE framework
- Helma XML-RPC was migrated to APACHE-XML recently
XML-RPC Implementations

- C/C++
- ObjectiveC
- Delphi/Kylix
- Java
- Perl
- PHP
- Python
- TCL
- Ruby
- Rebol
- Scheme
- Lisp
- Lingo client
- AppleScript
- JavaScript
- ASP
- Cold Fusion
- WebObjects
- Zope
- Flash client
Our First XML-RPC Request

POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
<methodCall>
    <methodName>examples.getStateName</methodName>
    <params>
        <param>
            <value><i4>41</i4></value>
        </param>
    </params>
</methodCall>
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><string>SouthDakota</string></value>
    </param>
  </params>
</methodResponse>
<table>
<thead>
<tr>
<th>Datatype</th>
<th>Desc</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>A signed, 32-bit integer.</td>
</tr>
<tr>
<td>string</td>
<td>An ASCII string, which may contain NULL bytes</td>
</tr>
<tr>
<td>boolean</td>
<td>Either true or false.</td>
</tr>
<tr>
<td>double</td>
<td>A double-precision floating point number</td>
</tr>
<tr>
<td>dateTime.iso8601</td>
<td>e.g. 19980717T14:08:55</td>
</tr>
<tr>
<td>base64</td>
<td>Raw binary data of any length with base64 encoding</td>
</tr>
<tr>
<td>array</td>
<td>An one-dimensional array of values. Individual values may be of any type. &lt;arrays&gt;s can be recursive, any value may contain an &lt;array&gt; or any other type, including a &lt;struct&gt;</td>
</tr>
<tr>
<td>struct</td>
<td>A collection of key-value pairs. The keys are strings; the values may be of any type. Structs can be recursive and can contain a &lt;struct&gt; any other type including &lt;array&gt;</td>
</tr>
</tbody>
</table>
Here's an example of a four-element array:

```xml
<array>
  <data>
    <value><i4>12</i4></value>
    <value><string>Egypt</string></value>
    <value><boolean>0</boolean></value>
    <value><i4>-31</i4></value>
  </data>
</array>
```
Here's an example of a two-element `<struct>`:

```xml
<struct>
  <member>
    <name>lowerBound</name>
    <value><i4>18</i4></value>
  </member>
  <member>
    <name>upperBound</name>
    <value><i4>139</i4></value>
  </member>
</struct>
```
Let's look at some code

```java
package server;
import helma.xmlrpc.*;

public class XmlRpcServer {

    public static void main( String[] args ) {
        try {
            WebServer webserver = new WebServer(12345);
            webserver.addHandler("wisdomserver",
                new WisdomServer());
        }
        catch( Exception e ) {
            e.printStackTrace();
        }
    }
}
```

Create HTTP Server

Register Server Object
package server;

public class WisdomServer {

    public String getWisdom() {
        return "Resistance is futile";
    }

    public String getWisdomForUser( String name) {
        return "Dear " + name + ": " + getWisdom();
    }
}

Let's look at some code
public void testMe() {

    String serverAddr = "http://localhost:12345";
    XmlRpcClient xmlrpc = null;
    xmlrpc = new XmlRpcClient( serverAddr );

    String result = (String) xmlrpc.execute(
            "wisdomserver.getWisdom",
            new Vector()
    );
}

Let’s look at some code

Initialize Client

Call Server
# import XML-RPC library
package require XMLRPC

# simple XML-RPC call
XMLRPC::create wisdomserver.getWisdom \  
 -proxy http://localhost:12345 \  
 -params {}

# pass an additional argument
XMLRPC::create wisdomserver.getWisdomForUser \  
 -proxy http://localhost:12345 \  
 -params { name string }
Using TCL as XML-RPC client
How to implement a XML-RPC server

1) Choose a SAX Parser
   - MinML or XERCES are fine

2) How to run a server
   - Use built-in light-weight multithreaded webserver
   - Embed XML-RPC server into a servlet engine
   - Use TURBINE with a built-in XML-RPC service

3) How to invoke a server method
   - Use reflection to find the appropriate server method
   - Roll your own dispatcher
Embedding a XML-RPC into a servlet

```java
public void doGet(HttpServletRequest request, 
        HttpServletResponse response) throws IOException 
{
    try 
    { 
        byte[] result = xmlrpc.execute(
            request.getInputStream());

        response.setContentType( "text/xml" );
        response.setContentLength( result.length );
        OutputStream out = response.getOutputStream();
        out.write (result);
        out.flush ();
    }
    catch( Exception e ) { ... }
}
```

Delegate To XML-RPC

Create HTTP Response
## The XML-RPC/JAVA Mapping

<table>
<thead>
<tr>
<th>XML-RPC data type</th>
<th>Data Types generated by Parser</th>
<th>Types expected by the Invoker as input</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;i4&gt;</code> or <code>&lt;int&gt;</code></td>
<td>java.lang.Integer</td>
<td>int</td>
</tr>
<tr>
<td><code>&lt;boolean&gt;</code></td>
<td>java.lang.Boolean</td>
<td>boolean</td>
</tr>
<tr>
<td><code>&lt;string&gt;</code></td>
<td>java.lang.String</td>
<td>java.lang.String</td>
</tr>
<tr>
<td><code>&lt;double&gt;</code></td>
<td>java.lang.Double</td>
<td>double</td>
</tr>
<tr>
<td><code>&lt;dateTime.iso8601&gt;</code></td>
<td>java.util.Date</td>
<td>java.util.Date</td>
</tr>
<tr>
<td><code>&lt;struct&gt;</code></td>
<td>java.util.Hashtable</td>
<td>java.util.Hashtable</td>
</tr>
<tr>
<td><code>&lt;array&gt;</code></td>
<td>java.util.Vector</td>
<td>java.util.Vector</td>
</tr>
<tr>
<td><code>&lt;base64&gt;</code></td>
<td>byte[]</td>
<td>byte[]</td>
</tr>
</tbody>
</table>
Advanced Features of Apache XML-RPC

- Asynchronous client invocations
  - Creates a worker thread
- Stand-alone XML-RPC server allows access restriction based on TCP/IP address
- Connection Keep Alive for reusing sockets
- Authenticated HTTP connection
  - Username and password are BASE64 encoded
- Custom Serializers are currently not available
  - Other XML-RPC libraries such as Marquée allow the registration of custom serializers
XML-RPC versus SOAP

Pros

- Has an elaborate type system
- Is the foundation of XML-based RPC mechanism (webservices, BizTalk)
- Utilizes multiple transport layers such as HTTP, FTP or SMTP (at least in theory)

Contras

- Interoperability between SOAP implementation
- Suffers from feature creep

If XML-RPC is too simplistic then use SOAP!
The Invoice20one Enterprise Project
The Requirements

We needed a RPC mechanism which

- Is easy to integrate in proprietary environments (e.g. Navision with C libraries)
- Works in the presence of firewalls
- Can be used for stand-alone server or within a servlet engine such as TOMCAT
- Has no associated license costs
- Ability to use SSL and client-side authentication
- Easy to embed into a scripting language

XML-RPC was the answer ...
The Implementation

- XML-RPC client transmits XML documents (invoices according to the openTRANS spec with two java.util.Hashtables, one java.util.Vector and a dozen of other parameters)
- XML-RPC server can be run as stand-alone TURBINE application or within TOMCAT
  - No changes of client configuration required
- XML-RPC server is build using a simple framework
  - Pluggable XML-RPC commands
  - Tracing of input and output parameters
The Result

- XML-RPC java libraries including XML parser are very small (64 Kbyte)
- We can embed XML-RPC directly into ERP systems using Java or C/C++
- We are able to transmit a few thousands invoices/h
- Works well apart from Turbine stand-alone application (older version of XML-RPC) with more than 10 simultaneous client requests (bug in thread pool code while adding new worker threads)
XML-RPC is a simple but powerful RPC mechanism using XML over HTTP
XML-RPC allows lightweight client/server applications on heterogenous platforms
XML-RPC is gaining more popularity despite of SOAP through KDE, Turbine and Apache XML project
XML-RPC is easy to integrate with existing scripting languages such as Perl or TCL
XML-RPC works well for moderate client/server applications in terms of response time and scalability

*Does distributed computing have to be any harder than this?*
## Resources

<table>
<thead>
<tr>
<th>Website</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.xmlrpc.org">www.xmlrpc.org</a></td>
<td>The home of XML-RPC</td>
</tr>
<tr>
<td><a href="http://www.soapware.org">www.soapware.org</a></td>
<td>Site for informations about SOAP</td>
</tr>
<tr>
<td>xml.apache.org/xml-rpc</td>
<td>The Apache XML-RPC page</td>
</tr>
<tr>
<td><a href="http://www.jxml.com/coins/">www.jxml.com/coins/</a></td>
<td>The home of Coins which replaces the Java serialization with XML</td>
</tr>
<tr>
<td>xmlrpc-c.sourceforge.net</td>
<td>C/C++ extensions of XML-RPC</td>
</tr>
<tr>
<td>xmlrpc.sourceforge.net</td>
<td>The home of Marquée XML RPC another implementation of XML-RPC in Java</td>
</tr>
<tr>
<td>xmlrpc-c.sourceforge.net/xmlrpc-howto.html</td>
<td>The XML-RPC How-To by Eric Kidd with examples in C, C++, Java, Perl, PHP, Python, Ruby, Microsoft .NET</td>
</tr>
</tbody>
</table>