Building Next Generation Web Applications using JSON-RPC-Java

Michael Clark <michael@metaparadigm.com>

http://oss.metaparadigm.com/jsonrpc/

Presentation Overview

- **Current state of Web interaction**
- **A more dynamic approach**
  - The *"Weblication"* concept and *"web remoting"*
  - The XMLHttpRequest object
  - The data [un]marshalling problem
- **JSON, JSON-RPC and JSON-RPC-Java**
  - Background on these technologies
  - Transparently calling remote Java methods from Javascript
- **Using JSON-RPC-Java to create highly dynamic web apps**
  - A Simple application disected
  - Advanced topics: Asynchronous operation, Type mapping
  - Demos

Current state of Web Interaction

- **The Web has allowed for a lot greater reach**
  - Created the 'inside-out' business model – customer empowerment
  - No special client software installation req'd other than a standards compliant web browser
- **Step backwards in terms of interactivity**
  - Desktop application are much more responsive
  - With the web we have to wait while an application reloads and renders 100K of HTML after a single button click
- **Signs of improvement**
  - Google Mail (Gmail), Google Suggests

The “Weblication”

- **Highly interactive Web Applications**
  - Can be paralled to a traditional desktop Application
  - Designed for high usuability as users tend to enter these applications and use for an extended duration
- **Advanced usage of JavaScript and DHTML**
  - Mimic traditional user interface rather than being page based
  - Large amount of front end application logic runs in the browser (rather than in the JSP, ASP, etc).
- **Use of remote calls to avoid page reloading**
  - Many are browser specific (using ActiveX)
- **'Gmail' is a good example**

A more dynamic approach

- **Technologies to avoid page reloading**
  - Perhaps we could call this *"web remoting"*
- **MSRS - Microsoft Remote Scripting**
  - Uses a Java applet to provide remote communications to the server
  - Only works on browsers with JVM **and** Liveconnect
    - Limited platforms and browsers.
- **JSRS – JavaScript Remote Scripting**
  - Uses hidden frames and form posting to send data to and from server
  - Only supports asynchronous operation
- **.NET (SOAP, etc)**
  - Microsoft only ActiveX (if you want to live in an IE only world)

The XMLHttpRequest object

- **Allow you to send HTTP requests from your JavaScript code**
- **Part of the Web Applications 1.0 Working Draft Specification**
  - Apple, Mozilla Foundation, Opera
  - http://www.whatwg.org/specs/web-apps/current-work/
- **Supported in a wide range of browsers**
  - Internet Explorer 5.0+, Mozilla 1.3+, Firefox 1.0, Safari 1.2, Opera 7.6+, Konqueror 3.3 (with patch), other Gecko browsers (Netscape, Camino, K-Meleon, Galeon)
  - No IE 5.x on Macintosh
    - Who cares, plenty of other browsers to choose from
  - http://oss.metaparadigm.com/jsonrpc/browser.jsp

The XMLHttpRequest object (cont.)

Using the XMLHttpRequest object

```
var http = new XMLHttpRequest();
http.open("POST", "/some-url/");
http
http
var
```

```
msxmlNames = [ "MSXML2.XMLHTTP.5.0", "MSXML2.XMLHTTP.4.0",
          "MSXML2.XMLHTTP.3.0", "MSXML2.XMLHTTP", "Microsoft.XMLHTTP" ];

function getHttp() {
    // Mozilla XMLHttpRequest
    try {
        return new XMLHttpRequest();
    } catch(e) {}

    // Microsoft MSXML ActiveX
    for (var i=0i < msxmlNames.length; i++)
    try {
        return new ActiveXObject(msxmlNames[i]);
    } catch (e) {}
}
```

Microsoft complications

The data [un]marshalling problem

- **Programmers can send and receive data easily but we need a common format to encode complex data structures.**
  - Seen many adhoc solutions: from simple comma seperated lists to more elboarate marshalling and unmarshalling schemes.
  - Need to avoid writing data formatting and parsing code each time we want to send a message, makes the programming challenge greater to adpoting this technology.
  - We could use XML-RPC?
  - XML is a bit heavy weight for the job. Would need to implement an XML parser in JavaScript to get this to work cross-browser.
  - Even with an XML-RPC implementation, still would need to use cumbersome API to call remote methods.

JSON – JavaScript Object Notation

- **JavaScript is the defacto web client scripting language**
  - Why not speak a language the browser already speaks when sending and receiving data to and from our JavaScript applications.
- **JavaScript Object Notation**
  - Lightweight data-interchange format with language bindings for C, C++, C#, Java, JavaScript, Perl, TCL and others.
  - Subset of ECMA-262 ECMAScript 3rd Edition (JavaScript 1.5 or MS Jscript 5.0)
  - No parsing required in JavaScript to unmarshall – just assign to a variable
  - http://json.org/

JSON – JavaScript Object Notation (cont.)

- **Built on these primitives:**
  - **Object, Array, String, Number, Boolean, null**
    - An object is an unordered set of name/value pairs. An object begins with **{** (left brace) and ends with **}** (right brace). Each name is followed by **:** (colon) and the name/value pairs are separated by **,** (comma).
    - An array is an ordered collection of values. An array begins with **[** (left bracket) and ends with **]** (right bracket). Values are separated by **,** (comma).
    - A value can be a string in double quotes, or a number, or true or false or null, or an object or an array. These structures can be nested.
    - A string is a collection of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string.

JSON – JavaScript Object Notation (cont.)

Simplified JSON description

object **::**
  { members **}**
  {}
members **::**
  string **:** value
  members **,** string **:** value
array **::**
  [elements **]**
  []
elements **::**
  value
  elements **,** value
value **::**
  string
  number
  object
  array
  true
  false
  null

Example JSON - An array of objects with nested arrays

```
[
 { “country”: “New Zealand”,
   “population”: 3993817,
   “animals”: [“sheep”, “kiwi”]
 },
 { “country”: “Singapore”,
   “population”: 4353893,
   “animals”: [“merlion”, “tiger”]
  }
]
```

JSON-RPC – Remote Procedure Call

- **An remote procedure call protocol similar to XML-RPC only uses JSON instead of XML as the data marshalling format**
  - http://xmlrpc.com/
  - http://json-rpc.org/
- **JSON-RPC vs. XML-RPC**
  - JSON is much lighter-weight than XML
  - JSON is the native format of JavaScript
  - No need for a JavaScript XML parser in the browser
  - Parsing of JSON is done natively by JavaScript engine using simple object assignment
- **SOAP**
  - Don't want to go there :)

JSON-RPC – Remote Procedure Call (cont.)

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 185

<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
   <param>
    <value>
     <i4>41</i4>
    </value>
   </param>
  </params>
</methodCall>
```

Exampe XML-RPC message

Equivalent JSON-RPC message

```
POST /JSON-RPC HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: oss.metaparadigm.com
Content-Type: text/plain
Content-length: 59

{
  “method”: “examples.getStateName”,
  “params”: [ 41 ]
}
```

JSON-RPC-Java

- **JSON-RPC-Java is a JSON-RPC implementation in Java**
  - Works with tomcat, JBoss and other web containers
- **Uses Java reflection**
  - Goal is to make accessing remote Java objects from JavaScript completely transparent
  - Export an object's methods with one line of code
  - Creates dynamic proxies in JavaScript for exported Java objects
  - Dynamically maps between Java and JavaScript objects
- **Leverages J2EE security model**
  - Session specific exporting of objects, can integrate with JAAS
- **Wide range of browser support + Unicode**
  - Internet Explorer, Mozilla, Firefox, Safari, Opera and Konqueror

JSON-RPC-Java (cont.)

- **Transparent marshalling and unmarshalling of:**
  - Primitive types (int, long, short, byte, boolean, char, float, double)
  - Numbers (Float, Integer, etc...), Strings, Char and Byte arrays
  - Java Beans (any Object with set and get methods)
  - Arrays of primitive types, Strings, Numbers, Collections and Java Beans
  - Exceptions (although type information isn't currently preserved
  - Concrete and abstract collection types:
    - List, ArrayList, LinkedList, and Vector
    - Map, HashMap, TreeMap, and LinkedHashMap
    - Set, HashSet, TreeSet, and LinkedHashSet
    - Dictionary, Hashtable
  - Any arbitrary nested combination of the above

JSON-RPC-Java (cont.)

- **Object Broker functionality**
  - References
    - Objects of classes registered as References will be returned as opaque reference objects to JavaScript (not by value by by reference).
    - When these opaque references are passed to succussive Java method calls they will then be reassociated back to the original Java object (great for security sensitive objects).
  - Callable References
    - Objects of classes registered as Callable References will return dynamic proxies to allow invocation on the particular object instance in the server-side Java.
    - Supports the 'factory' pattern such as that used by the EJB home
    - There are extensions to the JSON-RPC protocol in the provided JSON-RPC JavaScript client for dynamic proxy creation support.

A Simple JSON-RPC-Java example

Example JSP (Hello.jsp)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<jsp:useBean id="JSONRPCBridge" scope="session" class="com.metaparadigm.jsonrpc.JSONRPCBridge" />
<jsp:useBean id="hello" scope="session" class="com.metaparadigm.jsonrpc.test.Hello" />
<% JSONRPCBridge.registerObject("hello", hello); %>
<html>
  <head>
    <script type="text/javascript" src="jsonrpc.js"></script>
    <script type="text/javascript" src="hello.js"></script>
  </head>
  <body bgcolor="#ffffff" onLoad="onLoad()">
    <p>Who: <input type="text" id="who" size="30" value="Michael">
      <input type="button" value="Say Hello" onclick="clickHello()"></p>
  </body>
</html>
```

```
package com.metaparadigm.jsonrpc.test;

public class Hello
{
    public String sayHello(String who)
    {
            return "hello " + who;
    }
}
```

Example Java (Hello.java)

```
function onLoad()
{
    jsonrpc = new JSONRpcClient("/JSON-RPC");
}

function clickHello()
{
    var whoNode = document.getElementById("who");
    var result = jsonrpc.hello.sayHello(whoNode.value);
    alert("The server replied: " + result);
}
```

Example JavaScript (hello.js)

Advanced Topics (Asynchronous calls)

- **Asynchronous calls vs Synchronous calls**
  - Synchronous calls are more convenient to program with
  - Asynchronous calls keep the browser available while the call happens in the background **(recommended approach)**
  - JSON-RPC-Java makes asynchronous RPC calls easy:

```
function gotAnimalResult(result, exception) {
   if(exception) { alert(exception.message); }
   // do stuff here ...
}
jsonrpc.country.getAnimals(gotAnimalResult, "Singapore");
```

  - Give your application push capability
    - Make an async call and have the server block on the call until some event happens
  - Supports multiple concurrent operations

Advanced Topics (Type Mapping)

- **Java to JavaScript Type mapping**
  - To allow JSON-RPC-Java to transparently unmarshall complex nested objects and with that, the usage of Java's container classes, JSON-RPC needs a mechanism to preserve type information.
    - JavaScript's typeless nature
    - Java's single Object base class pattern for Container genericity
      - (We could solve this with Java 5.0 parameterized types)
  - JSON-RPC-Java does 'Class hinting' by adding a 'javaClass' attribute to objects sent to JavaScript to allow less ambiguous marshalling when the objects are returned back to the Java side.
  - Special mappings for container types: List, Map, Set (see docs)
  - Java Bean readable properties are mapped directly to JavaScript objects (without the get or set prefix), writable properties are set when the JavaScript 'bean' object is sent back to the Java side

JSON-RPC-Java Demos

- Demos
  - http://oss.metaparadigm.com/jsonrpc/demos.html
- Download
  - http://oss.metaparadigm.com/jsonrpc-dist/json-rpc-java-0.8.tar.gz
- Mailing List
  - http://oss.metaparadigm.com/mailman/listinfo/json-rpc-java
- Contribute

```
# export CVSROOT=:pserver:anoncvs@cvs.metaparadigm.com:/cvsroot
# cvs login
Logging in to :pserver:anoncvs@cvs.metaparadigm.com:2401/cvsroot
CVS password: <enter 'anoncvs'>
# cvs co json-rpc-java
# cd json-rpc-java
< hack, hack, hack, ...>
# cvs diff -u | mail -s “my patch” michael@metaparadigm.com
```