Reading JSON Data

Another way to access external data is to pass it in as a dynamic parameter. Since all parameters are strings, we need some way to define collections and attributes in a string so that they can be extracted again. JSON (JavaScript Object Notation) is an ideal way to do this.

Building the JSON string

You can either use a JSON utility from http://json.org or easily construct the string yourself. As an example, assume you need to transfer a list of names and passwords to print as a report. Here is the source data:

```
<table border="1">
<tr><th>user</th><th>pass</th></tr>
<tr><td>admin</td><td>an</td></tr>
<tr><td>test</td><td>tt</td></tr>
<tr><td>user</td><td>ur</td></tr>
<tr><td>fred</td><td>fd</td></tr>
<tr><td>bill</td><td>bl</td></tr>
</table>
```

As a JSON string it looks like this:

```
[{"user":"admin","pass":"an"},{"user":"test","pass":"tt"},{"user":"user","pass":"ur"},
{"user":"fred","pass":"fd"},{"user":"bill","pass":"bl"}]
```

As you can see, a JSON Writer is hardly necessary in this case. However, if you want to use one, the code would look something like this:

```
import java.io.StringWriter;
import java.util.ArrayList;
import java.util.List;
import org.json.JSONObject;
import org.json.JSONWriter;

public class Test_JSON
{
 public static void main(String[] args) throws Exception
 {
  // build a list to simulate obtaining the data from elsewhere
  List<Pair> list = new ArrayList<Pair>();
  list.add(new Pair("admin","an"));
  list.add(new Pair("test","tt"));
  list.add(new Pair("user","ur"));
```

```
 list.add(new Pair("fred","fd"));
 list.add(new Pair("bill","bl"));

 // turn the list into a JSON string
 StringWriter sw = new StringWriter();
 JSONWriter jw = new JSONWriter(sw);
 jw.array();
 String[] names = new String[]{"user","pass"};
 for (Pair p : list) jw.value(new JSONObject(p,names));
 jw.endArray();

 // this is what it looks like
 System.out.println(sw.toString());
}

public static class Pair
{
 public Pair(String u, String p)
 {
  user = u;
  pass = p;
 }
 public String user;
 public String pass;
}
}
```

Accepting the JSON string

This is the easy part. Because our JSON string is already valid JavaScript syntax, we can just substitute it directly into the Object DataSource. Here is the code to enter into the Object DataSource Wizard:

```
function pushTo(/*PushContext*/ cxt, /*DataListener*/ dl)
{
 var array = ${ARRAY};
 dl.startData(this);
 for (var i=0;i<array.length;i++)
 {
  var rec = this.newRecordInstance();
  var data = rec.getData();
  data[0] = array[i].user;
  data[1] = array[i].pass;
  dl.processRecord(rec);
 }
```

```
  dl.endData(this);
}
```

The dynamic parameter ${ARRAY} is the substitution point. If you save and generate the data from this source, you will be prompted to enter ARRAY. Copy and paste the JSON string above and you will see it turned back into records.

Adding a Report

You can now create a fresh report and choose the newly created datasource. Make sure the **Propagate datasource parameters to report** option on the *Report Wizard - Choose Datasource* page is ticked. The report will now also prompt for ARRAY when rendered. If you are using the Runtime or Server tools, you can now pass your JSON string as a dynamic parameter to the report.

Here is a sample JSON Datasource.