Reason - A Technical Magazine
ISSN 2277–1654
Vol - X ● 2011

Research Article

ESCAPE SOA THROUGH CROSS DOMAIN ACCESS DWR APPROACH

Biswajit Pal

Associate- Cognizant Technology Solutions, Kolkata, India
email: pal.biswa@gmail.com

Abstract: There are a number of strategies that has the ability to call server side methods from the page itself directly. An emerging technology is introduced in this paper named DWR - Direct Web Remoting that can iterate through java collections from browser so that it is not dependent on XML/JSON based data exchange approach. The working strategy of DWR is also discussed to facilitate cross domain access. Its advantages and limitations are mentioned.

Keywords: *Webpage, DWR, cross domain access*

1. Introduction

Sometimes requirement comes in a way that there is need to access separate web domain from the same webpage running in browser. In the era of web 2.0 across the world, developers and designers work together to give desktop like feel to users for all web applications so that there will be no url redirection while interacting in a web portal. To achieve this, applications are become rich in AJAX calls.

But, in ajax based web applications, in which people are regularly working, they need to parse the data to and fro to render in page. There are several technologies existing in market to cater this like EXT-JS, JQUERY for easy handling JSON or XML string.

But, the parsing of XML/JSON string still exists that an application developer cannot avoid. If one looks into J2ee perspective, server side developer needs to prepare the JSON/XML response before sending the data to client and vice-versa. Here, the author will discuss another emerging technology which has the ability to call server side methods from the page itself directly and can iterate through java collections from browser so that he does not require to depend upon XML/JSON based data exchange approach. This is DWR - Direct Web Remoting.

Besides this, one of the best interesting features of DWR is that one can call separate server side function from the same page itself which are running in separate web domain, which is cross site or cross domain XMLHttpRequest. It gives one the flexibility to call different domain objects from the same browser, but user is not getting any feel of that. In this paper, detail of DWR is discussed.

2. Familiarizing with DWR

DWR is a Java library that enables Java on the server and JavaScript in a browser to interact and call each other. DWR generates JavaScript to allow web browser call into Java code almost as if it is running locally. It can marshal virtually any data including collections, POJOs, XML and binary data like images and PDF files. All that is required is a security policy that defines what is allowed. In nutshell, one can call java function directly from the page itself.

Now, the question will arise in mind why one uses DWR though there are stable products in market. As it is discussed in the beginning, through DWR one can iterate java collection in java script and not only that one can play with user defined java beans in addition. So, one does not require parsing XML to render data which is a tedious job. Now, the step to configure DWR is discussed first in a web application.

3. Configuration of DWR

- **To install the DWR. JAR file**
  Dwr.jar is to place into the WEB-INF/lib directory of the web application.
- **To install the Commons Logging JAR file**
  DWR depends on Commons Logging. The commons-logging.jar is to download and to place it into the WEB-INF/lib directory of web application.
- **To add the DWR servlet definition and mapping to web.xml**
  One is to add the following lines to web application's deployment descriptor (WEB-INF/web.xml). The <servlet> section needs to go with any existing <servlet> sections, and likewise with the <servlet-mapping> section.

```
<servlet>
   <servlet-name>dwr-invoker</servlet-name>
   <display-name>DWR Servlet</display-name>
   <description>Direct Web Remoter Servlet</description>
   <servlet-class>org.directwebremoting.servlet.DwrServlet</servlet-
class>
   <init-param>
       <param-name>debug</param-name>
       <param-value>true</param-value>
 </init-param>>
</servlet>

<servlet-mapping>
    <servlet-name>dwr-invoker</servlet-name>
    <url-pattern>/dwr/*</url-pattern>
</servlet-mapping>
```

- **Creating the DWR configuration file (dwr.xml)**
  To create a new file in web application's WEB-INF directory (alongside web.xml) named dwr.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dwr PUBLIC "-//GetAhead Limited//DTD Direct Web Remoting
2.0//EN" "http://getahead.org/dwr/dwr20.dtd">
<dwr>
<allow>
<create creator="new" javascript="AjaxService" scope="application">
<param name="class" value="com.example.dwr.AjaxService "/>
            </create>
      </allow>
</dwr>
```

Now the application is configured with DWR.

1. Cross Domain Calls

One of the distinguished features of DWR is that one can call/access java methods residing in different web domain from the same page. This feature enables one to interact with functionality implemented in distributed way within the scope of same page, and moreover, the user will not able to distinguish as one is calling through XMLHttpRequests. If one tries to visualize, this will look like the following as shown in Fig. 1.

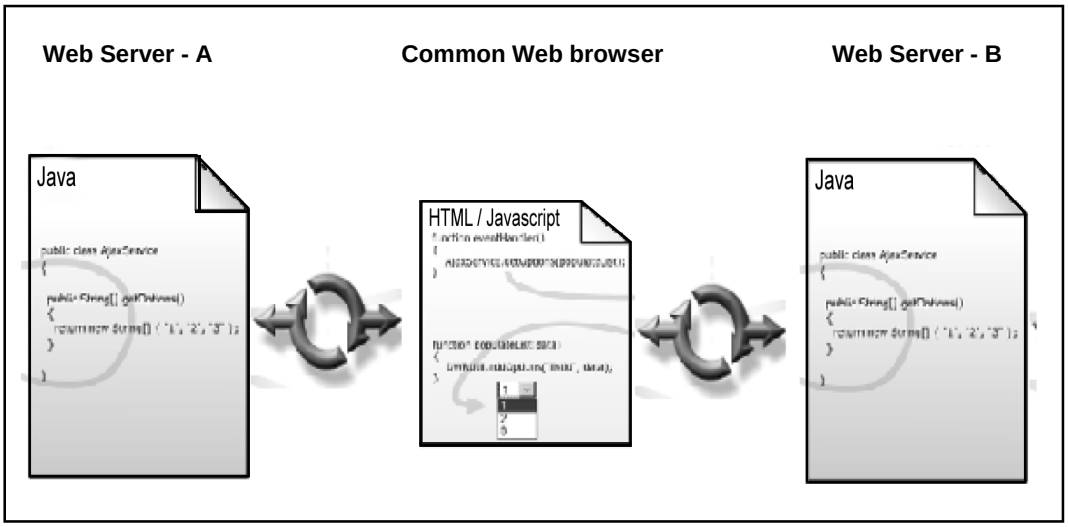

Figure 1 The scheme of cross domain access using DWR

4.1 Cross domain scripting – escaping complex SOA in simpler way

When one is interacting with different objects in a distributed environment, one generally thinks to do that using service oriented architecture or remote procedure calls. And to achieve that one needs to write lot of extra java codes for marshalling and unmarshalling of objects. One also needs to write complex codes for communicating with service provider domain as well as application domain which is benefiting from the service. So, one will need a fancy server to make it happen, one has to do a lot of unnecessary extra coding. Because server to server communication running on different web server is costlier in terms of complexity.

With Cross-Domain scripting, all one needs is Ajax and it is all set. No server side computation and no server side coding are required, no need to make calls out of the server- life becomes a lot simpler. But only bad news is that there are some security implications as because one is allowing one's script to talk to foreign server beyond one's web server scope. Unless one is very careful with how one does one's XHR call, one has to trust the foreign web sites.

4.2 Configuration to achieve cross domain access

To make the application cross domain enable, one needs to follow the following configuration.

4.2.1 Web.xml configuration

In web.xml file, one needs to add the following section to allow DWR application for X-Domain call.

```
<init-param>
  <param-name>allowScriptTagRemoting</param-name>
  <param-value>true</param-value>
</init-param>

<!-- Disables DWR's CSRF protection -->
<init-param>
  <param-name>crossDomainSessionSecurity</param-name>
  <param-value>false</param-value>
</init-param>

<!-- Enables GET requests which are necessary for X-domain calls -->
<init-param>
  <param-name>allowGetForSafariButMakeForgeryEasier</param-name>
  <param-value>true</param-value>
</init-param>
</servlet>
```

4.2.2 Configuration in Java script

1. To specify a path to DwrServlet in JavaScript, before engine.js is included .This is required because DWR makes an initial call to server when engine.js is loading.

```
<script>
    Var pathToDwrServlet ='http: //Remote_IP/remote-context name/dwr';
</script>
```

2. In the page from where one is calling a remote method (resides in a remote server) one needs to override path parameter in DWR interface .

```
<Interface Name >._path = '<remote_ip><remote-context name>/dwr';
```

As an example:

```
<script>
    DWRExample._path='http: //Remote_IP/remote-context name/dwr';
    DWRExample .demoMthod ();
</script>
```

4.2.3 Configuration in JSP

In the JSP, one needs to include the corresponding java-script name for the Remote java class. This java script will be dynamically generated by DWR. One only has to include the js file. Basically, this js represents the remote java class at page level, which enables DWR to call the remote function. The script is as follow:

```
<script type='text/javascript' src='http://<remote_ip><remote-context name>;/dwr/interface/Example.js'/>
```

Now, the configuration is done with and the application is ready for cross domain access.

5. Conclusion

In the previous paragraphs, the technique named direct web remoting (DWR) is introduced and its working strategy is discussed for cross domain access. Also its advantages and limitations are mentioned.

References

[1] http://directwebremoting.org/

[2] Proof of Concept on DWR Cross Domain Access.