



Spring Forward 2006
Sponsored by Chariot Solutions
and springdeveloper.com

Ajax, DWR and Spring

Bram Smeets
Interface21



Topics

- What is Ajax?
- The DWR approach to Ajax
- Advanced DWR
- Using Spring & DWR
- DWR roadmap
- Conclusion



Ajax

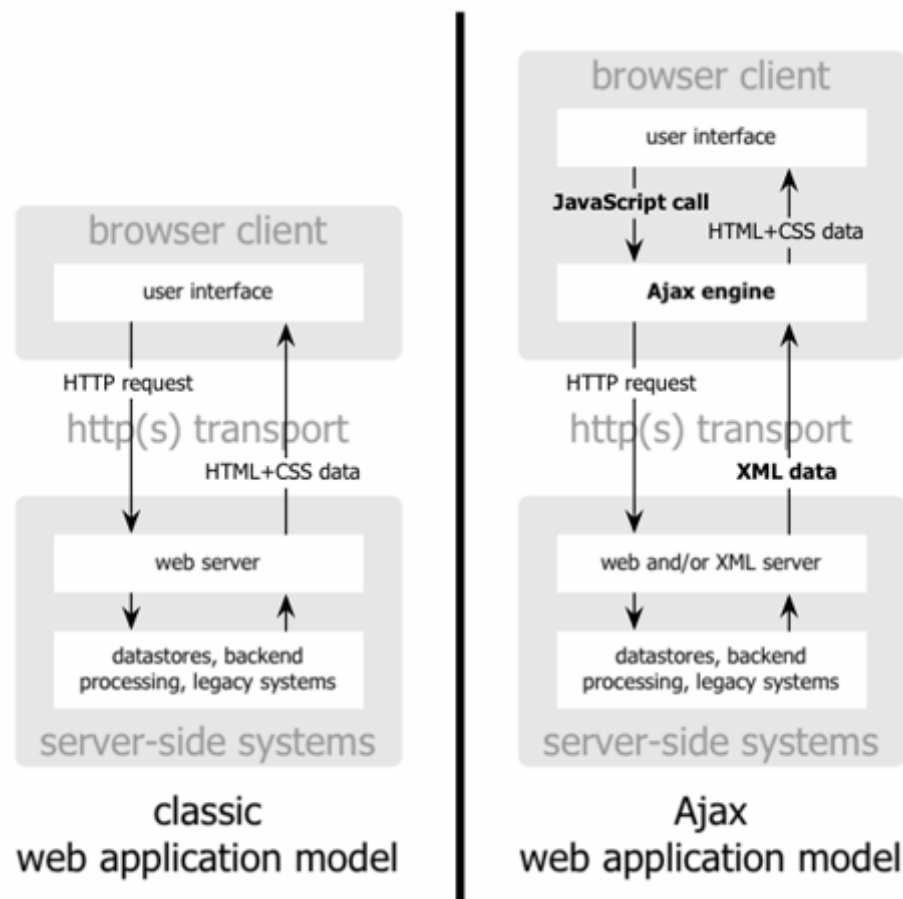
- Not one technology, several technologies
 - XHTML and CSS
 - Document Object Model (DOM)
 - XML (and XSLT)
 - XMLHttpRequest (XHR)
 - JavaScript

Asynchronous JavaScript And Xml

Jesse James Garrett, 2005

Ajax = DHTML + XHR

Ajax – Architecture (1)

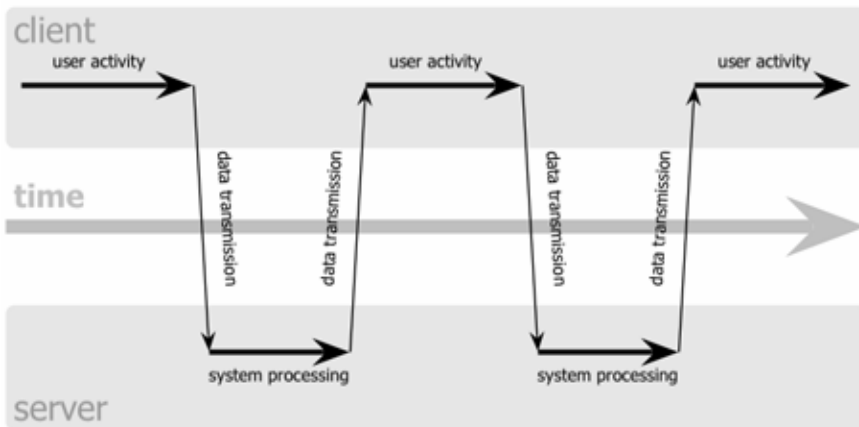


Jesse James Garrett, 2005

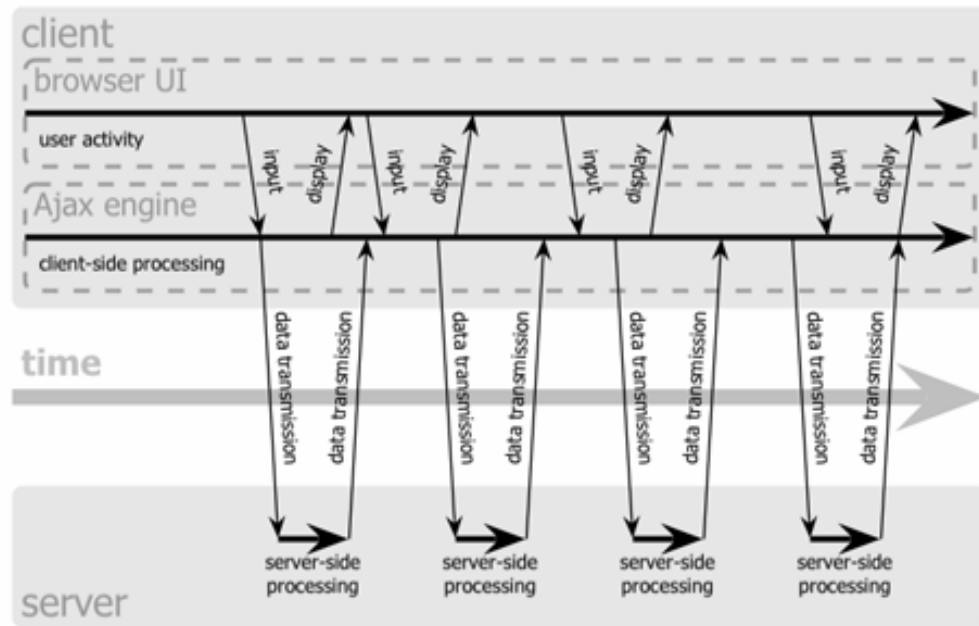


Ajax – Architecture (2)

classic web application model (synchronous)



Ajax web application model (asynchronous)



Jesse James Garrett, 2005



Ajax - Issues

- Browser incompatibilities
 - ActiveX in IE
 - Native in Mozilla/Firefox
- Back button and browser history
- Working with XML in JavaScript



Ajax – Framework overview

- JSON-RPC
- GWT
- DWR
- Many more...

- Why DWR?
 - Mostly used framework (according to Google and Burton Group)
 - Integrates best with Spring!


DWR – Overview

Web Browser

HTML / Javascript

```
function eventHandler()
{
    AjaxService.getOptions(populateList);
}

function populateList(data)
{
    DWRUtil.addOptions("listid", data);
}
```



The diagram shows a document icon representing the client-side code in the web browser. It contains two JavaScript functions. The first function, `eventHandler()`, calls `AjaxService.getOptions(populateList)`. The second function, `populateList(data)`, calls `DWRUtil.addOptions("listid", data)`. Below the code, a small UI element is shown: a dropdown menu with the value '1' selected, and a list box below it containing the values '1', '2', and '3'. A grey arrow points from the `eventHandler()` function to the dropdown menu, and another grey arrow points from the `populateList(data)` function to the list box.

Web Server

Java

```
public class AjaxService
{
    public String[] getOptions()
    {
        return new String[] { "1", "2", "3" };
    }
}
```

The diagram shows a document icon representing the server-side code on the web server. It contains a Java class named `AjaxService` with a public method `getOptions()` that returns a new array of strings containing "1", "2", and "3". A grey arrow points from the `getOptions()` method back to the `eventHandler()` function in the web browser, and another grey arrow points from the `addOptions()` method in the web browser back to the `getOptions()` method.

DWR





DWR – Features (1)

- RPC-style Ajax
- Java to JavaScript marshalling (and vice versa)
 - Uses JavaScript objects
- Facilitate browser incompatibilities
 - Supports most browsers
 - Even allows fallback to iframes
 - Allows for cross-side scripting



DWR – Features (2)

- Provides integration with most common Java frameworks
 - Struts, Webwork, JSF, Hibernate, Rife, Spring, and etcetera...
- Provides good security mechanism
- Comes with handy utility methods
- Reverse Ajax



DWR – Configuration (1)

- **Declare `DwrServlet` in `web.xml`**

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



DWR – Configuration (2)

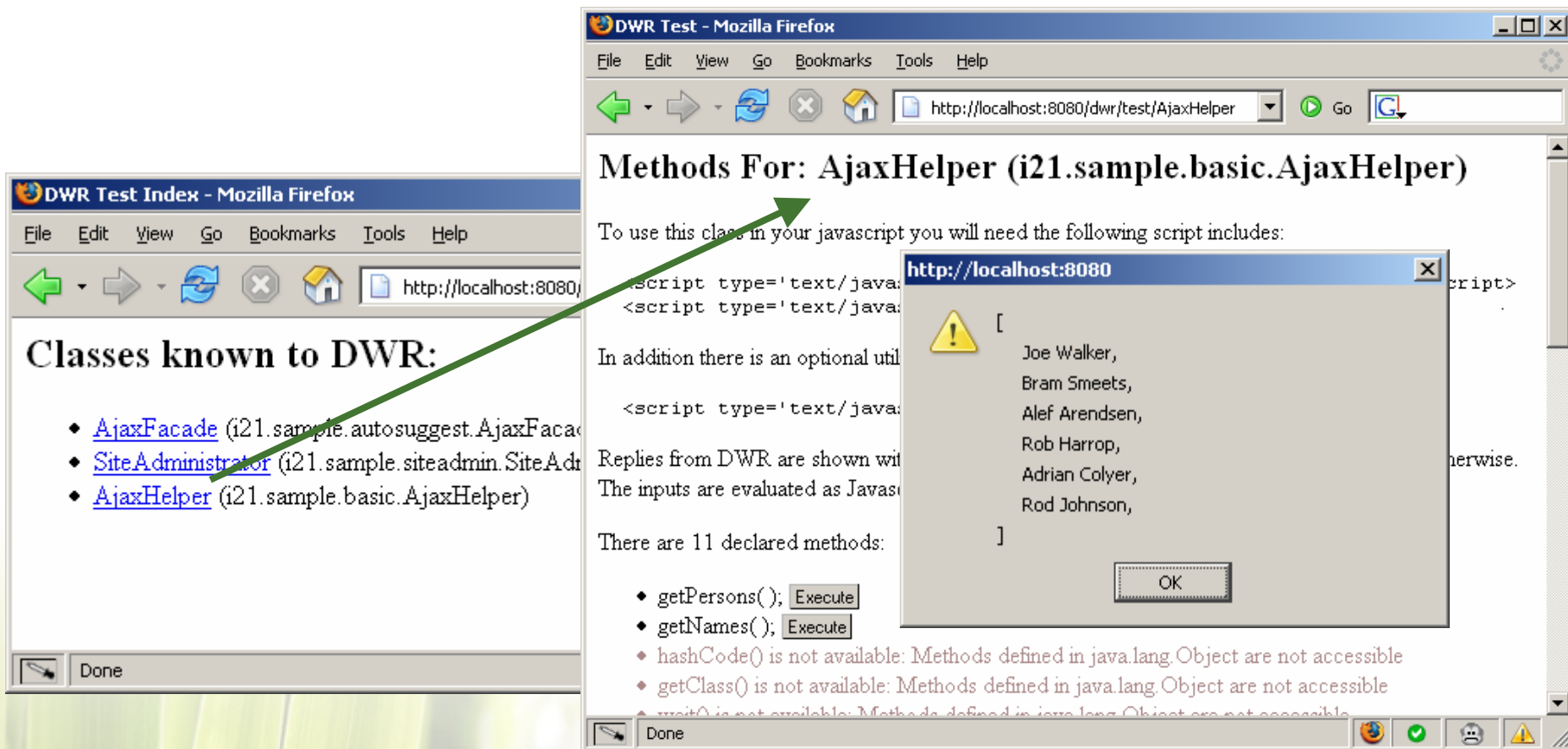
- Add `dwr.xml` in `/WEB-INF`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dwr PUBLIC
    "-//GetAhead Limited//DTD Direct Web Remoting 2.0//EN"
    "http://www.getahead.ltd.uk/dwr/dwr20.dtd">

<dwr>
  <allow>
    <create creator="new" javascript="AjaxFacade">
      <param name="class" value="i21.sample.basic.AjaxFacade"/>
    </create>
  </allow>
</dwr>
```

DWR – Debug pages

- Set the `init-param` of the `DWRServlet` to `true`



DWR Test Index - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://localhost:8080/

Classes known to DWR:

- [AjaxFacade](#) (i21.sample.autosuggest.AjaxFacade)
- [SiteAdministrator](#) (i21.sample.siteadmin.SiteAdministrator)
- [AjaxHelper](#) (i21.sample.basic.AjaxHelper)

DWR Test - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://localhost:8080/dwr/test/AjaxHelper

Methods For: AjaxHelper (i21.sample.basic.AjaxHelper)

To use this class in your javascript you will need the following script includes:

```
<script type='text/javascript' src='...'></script>
<script type='text/javascript' src='...'></script>
```

In addition there is an optional utility class:

```
<script type='text/javascript' src='...'></script>
```

Replies from DWR are shown with the following format:
The inputs are evaluated as Javascript expressions.

There are 11 declared methods:

- `getPersons()`;
- `getNames()`;
- `hashCode()` is not available: Methods defined in java.lang.Object are not accessible
- `getClass()` is not available: Methods defined in java.lang.Object are not accessible
- `wait()` is not available: Methods defined in java.lang.Object are not accessible

http://localhost:8080

[

- Joe Walker,
- Bram Smeets,
- Alef Arendsen,
- Rob Harrop,
- Adrian Colyer,
- Rod Johnson,

]



DWR – A sample web page

- In your web page add:

```
<script type="text/javascript" src="/dwr/interface/AjaxHelper.js"></script>  
<script type="text/javascript" src="/dwr/engine.js"></script>
```

- Call methods on the exposed object

```
<script type="text/javascript">  
  function getData() {  
    AjaxHelper.getData(handleData);  
  }  
  
  function handleData(data) {  
    alert('data: ' + data);  
  }  
</script>
```



Demo

DWR



DWR – Security

- No objects/classes are exposed without explicitly configuring it!
- All methods on exposed objects are exposed by default
 - Use include/exclude rules per method
- Only objects are marshaled for which a converter is registered
- Role restrictions
 - Use security constraints on DWR paths
 - Specify an auth constraint per creator (JAAS)
 - Use Acegi interceptors
- Disable debug mode in live environment!!



DWR – Utility functions

- Include the utility script

```
<script type="text/javascript" src="/dwr/util.js"></script>
```

- Among others, it offers:

- `$(‘id’)`
 - Works on most browsers
- `DWRUtil.toDescriptiveString(value)`
 - Generic `toString()` method
- `DWRUtil.cloneNode("template")`
- `DWRUtil.setValue(element, val)`
- `DWRUtil.getValue(element)`
- `DWRUtil.addOptions(list)`
- `DWRUtil.removeAllOptions(element)`
- `DWRUtil.addRows(element, data, ...)`
- `DWRUtil.removeAllRows(element)`



DWR – Advanced features (1)

- Signatures
 - Explicitly specify method signatures

```
<signatures>
  <![CDATA[
    import java.util.List;
    import i21.sample.basic.Person;
    PersonManager.batchUpdate(List<Person> list);
  ]]>
</signatures>
```



DWR – Advanced features (2)

- Accessing servlet parameters
 - Use `WebContext`

```
WebContext ctx = WebContextFactory.get();
ctx.getHttpServletRequest();
ctx.getHttpServletResponse();
ctx.getServletContext();
...
```

- Or add them to the method signature on your exposed beans
 - Auto filled in by DWR

```
Person getPerson(HttpServletRequest request, long id);
```



DWR – Advanced features (3)

- Ajax filters
 - Extra latency filter
- GMail style loading messages
 - `DWRUtil.useLoadingMessage();`
 - `DWRUtil.useLoadingMessage('Waiting...');`
 - Customize even further:
 - `$('disabledZone').style.color = white;`



DWR – Advanced features (4)

- Other options

- Globally: `DWREngine.setX()`
- Per call: `{ timeout: 500, callback: myCallback }`

- `timeout`
- `errorHandler/warningHandler`
- `preHook/postHook`
- `method` (`DWREngine.XMLHttpRequest`, `DWREngine.IFrame` or `DWREngine.ScriptTag`)
- `verb` (`GET` or `POST`)



DWR – Advanced features (5)

- **Batching**
 - `beginBatch()` ;
 - Make some DWR requests
 - `endBatch()` ;
- **Annotation support**
 - `@Create`
 - `@Convert`
 - `@RemoteMethod`
 - `@RemoteProperty`



Reverse Ajax (1)

- Call Javascript from Java code
- Many use cases
 - Chat application
 - Email application
 - Progress bar
- Work in progress!



Reverse Ajax (2)

- DWR provides the means to proactively send Javascript to the browser

– JavaScript: Turn Reverse Ajax on

```
DWREngine.setReverseAjax(true);
```

– Java: Send scripts to the browser

```
scriptSession.addScript("alert('Hi')");
```




Reverse Ajax (3)

- Use server-side API to call JavaScript functions
 - DWRUtil library

```
import org.directwebremoting.proxy.dwrutil.DwrUtil;  
DwrUtil util= new DwrUtil(...);  
util.addOptions("selectId", array);
```

- Script.aculo.us Effects library

```
import org.directwebremoting.proxy.scriptaculous.Effect;  
Effect effect= new Effect(...);  
effect.fade("selectId");
```



DWR & Spring

- DWR = AJAX with Spring made easy!
- Expose Spring-managed beans by means of JavaScript



DWR & Spring - Configuration (1)

- Option 1

- Define a `ContextLoaderListener` in `web.xml` to load the beans to expose
- Add a create statement to `dwr.xml`

```
<create creator="spring" javascript="AjaxFacade">  
  <param name="beanName" value="ajaxFacade"/>  
</create>
```

- Option 2

- Use the `ServletWrappingController` to wrap the `DWRServlet`

- Option 3

- Have DWR load the application context(s)

```
<create creator="spring" javascript="AjaxFacade">  
  <param name="beanName" value="ajaxFacade"/>  
  <param name="location-1" value="data-layer.xml"/>  
  <param name="location-2" value="service-layer.xml"/>  
</create>
```



DWR & Spring - Configuration (2)

- Use new DWR namespace

```
<bean id="ajaxFacade" class="i21.sample.autosuggest.AjaxFacade">  
  <dwr:remote javascript="AjaxFacade" />  
  <property name="autoSuggestService" ref="autoSuggestService" />  
</bean>
```

```
<dwr:configuration>  
  <dwr:create type="new" javascript="AjaxHelper"  
    class="i21.sample.basic.AjaxHelper" />
```

```
  <dwr:convert class="i21.sample.basic.Person" type="bean" />  
</dwr:configuration>
```

```
<dwr:controller id="dwrController" debug="true" />
```



Demo

Spring & DWR



DWR Roadmap

- Back button support
- Tighter integration with Spring
 - Extra form controllers e.g. for validation
 - Use Spring scoping
 - Enhancement of the Spring namespace support
- What feature(s) would you like?



Conclusion

- Ajax enables to build better web applications
- DWR provides easy Ajax
- DWR integrates nicely with Spring
- Still more to come...



Questions?

www.directwebremoting.org
bram.smeets@interface21.com



INTERFACE21 AND THE NO FLUFF JUST STUFF JAVA SYMPOSIUM SERIES BRING YOU

THE **SPRING EXPERIENCE**

December 7-10, 2006 in Hollywood, Florida at the Westin Diplomat



Spring Conference: The Spring Experience 2006 December 7th – 10th, Hollywood Florida by Interface21 and NoFluffJustStuff Java Symposiums

- ◆ World-class technical conference for the Spring community
- ◆ 3 full days, 5 concurrent tracks, 60 sessions
 - Core Spring 2.0
 - Core Enterprise 2.0
 - Core Web 2.0
 - Domain Driven Design
 - Just Plain Cool
- ◆ Enjoy five-star beach resort and amenities
- ◆ Converse with core Spring team and industry experts
 - Rod Johnson, Adrian Colyer, Ramnivas Laddad, Juergen Hoeller, etc.
- ◆ Registration at <http://www.thespringexperience.com>