



Pimp My Webapp (with Google Web Toolkit)

Hermod Opstvedt
Chief Architect
DnB NOR ITUD
Common components





Pimp My Webapp

- What is Google Web Toolkit (GWT)?
 - Pronounced GWiT.
 - An effort to enable Java developers to write Internet applications without needing to know JavaScript (in depth).
 - Java Runtime Emulation (not everything is supported).



Pimp My Webapp

- Why use GWT or any RIA?
 - Better response on GUI because it executes client side. Only data is transmitted on the wire.
 - Easier to implement “fancy” stuff such as fade in/out.



Pimp My Webapp

- Advantages of GWT
 - No need to learn JavaScript to create a RIA application
 - An understanding of JavaScript is advantageous.
 - All coding is done in Java
 - A familiar language
 - Apache license (2.0).



Pimp My Webapp

- All debugging is done stepping through in Java
 - Makes it easier to follow code and find bugs.
- Static type checking.
- JavaScript errors caught at compile time.



Pimp My Webapp

- You can roll your own scripts and use them.
 - JavaScript Native Interface (JSNI)
- Supports I18N
 - Dynamic
 - Static
- Hosted mode browser for debugging.



Pimp My Webapp

- Java to JavaScript
 - Because you are working in Java, it needs to be compiled into JavaScript before it can be executed client side (in a browser).
 - Compilation is done using a smart compiler that generates scripts for various browsers.
 - Not all browsers behave correctly.



Pimp My Webapp

- Download & Install

- Two versions available

- Version 1.4.x supports Java 1.4
- Version 1.5.x supports Java 5 (Supports generics, *etc.*)

- <http://code.google.com/webtoolkit/download.html>



Pimp My Webapp

➤ IDE support.

- Eclipse opensource plugins
 - ✓ Cypal studio – <http://code.google.com/p/cypal-studio/>
 - ✓ Eclipse guru – <http://eclipseguru.org/>
- Commercial Eclipse plugins
 - ✓ GWT Designer, Instantiations – <http://www.instantiations.com/gwt designer/>



Pimp My Webapp

- Hello World

- Step 1 is to set up an initial GWT project

- applicationCreator

- ✓ Creates a base project structure

- Or

- Projectcreator

- ✓ Creates both a project and a base working application



Pimp My Webapp

➤ Or

- IDE plugin

➤ Or

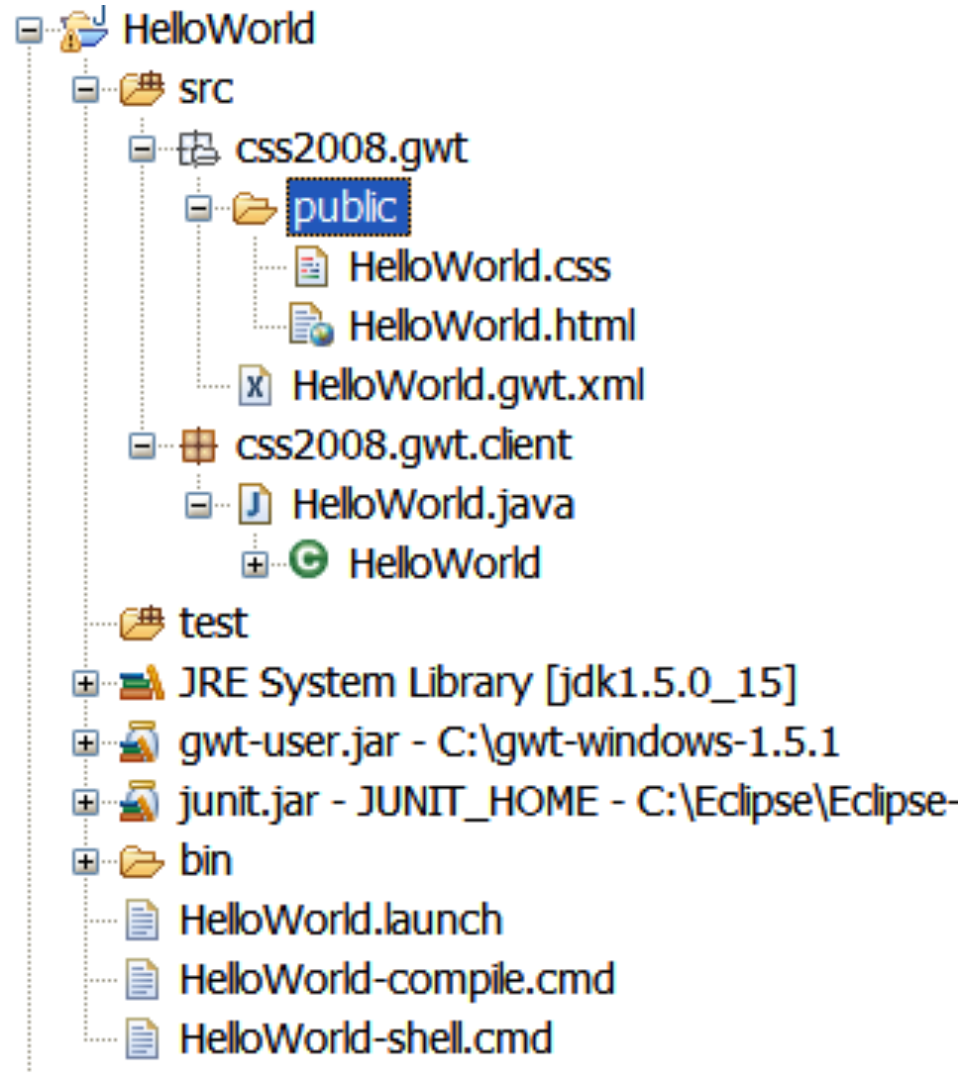
- Manually set up the files



Pimp My Webapp

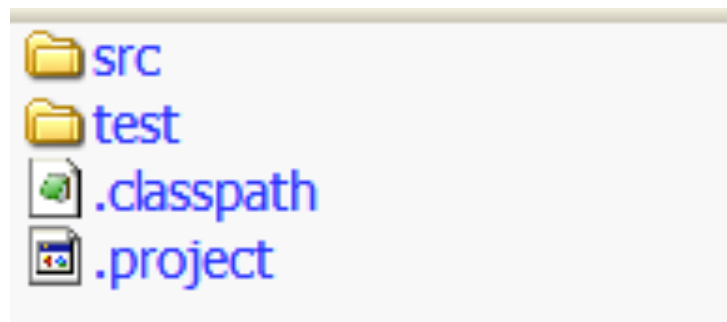
- applicationCreator [-eclipse projectName] [-out dir] [-overwrite] [-ignore] className
 - ✓ -eclipse Creates a debug launch config for the named eclipse project
 - ✓ -out The directory to write output files into (defaults to current)
 - ✓ -overwrite Overwrite any existing files
 - ✓ -ignore Ignore any existing files; do not overwrite
 - ✓ className The fully-qualified name of the application class to create

Pimp My Webapp



Pimp My Webapp

- `projectcreator [-ant projectName] [-eclipse projectName] [-out dir] [-overwrite] [-ignore]`
 - ✓ `-ant` Generate an Ant buildfile to compile source (.ant.xml will be appended)
 - ✓ `-eclipse` Generate an eclipse project
 - ✓ `-out` The directory to write output files into (defaults to current)
 - ✓ `-overwrite` Overwrite any existing files
 - ✓ `-ignore` Ignore any existing files; do not overwrite





Pimp My Webapp

- The HelloWorld GWT files:
 - HelloWorld.launch
 - An Eclipse launch file to run the application in the hosted mode browser
 - HelloWorld-compile.cmd
 - A command script that runs the GWT compiler, converting it into JavaScript and images.
 - HelloWorld-shell.cmd
 - A command script to run the application in the hosted mode browser



Pimp My Webapp

➤ HelloWorld.css

- Cascading Style Sheet for the application

➤ HelloWorld.html

- The web page that is loaded in the browser

➤ HelloWorld.gwt.xml

- The GWT project file, which controls how GWT handles the project.
 - ✓ Third party addins
 - ✓ Stylesheets
 - ✓ RPC Servlets
 - ✓ *etc.*



Pimp My Webapp

➤ HelloWorld.java

- The Java file that contains the base application, which is converted to JavaScript



Pimp My Webapp

➤ HelloWorld.launch

```
<?xml version="1.0" encoding="UTF-8"?>
<launchConfiguration type="org.eclipse.jdt.launching.localJavaApplication">
<booleanAttribute key="org.eclipse.jdt.launching.DEFAULT_CLASSPATH" value="false"/>
<stringAttribute key="org.eclipse.jdt.launching.MAIN_TYPE" value="com.google.gwt.dev.GWTShell"/>
<listAttribute key="org.eclipse.jdt.launching.CLASSPATH">
<listEntry value="&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;&#13;&#10;&lt;runtimeClasspathEntry
containerPath=&quot;org.eclipse.jdt.launching.JRE_CONTAINER&quot; javaProject=&quot;HelloWorld&quot; path=&quot;1&quot;
type=&quot;4&quot;/&gt;&#13;&#10;"/>
<listEntry value="&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;&#13;&#10;&lt;runtimeClasspathEntry
internalArchive=&quot;/HelloWorld/src&quot; path=&quot;3&quot; type=&quot;2&quot;/&gt;&#13;&#10;"/>
<listEntry value="&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;&#13;&#10;&lt;runtimeClasspathEntry
id=&quot;org.eclipse.jdt.launching.classpathentry.defaultClasspath&quot;&gt;&#13;&#10;&lt;memento
project=&quot;HelloWorld&quot;/&gt;&#13;&#10;&lt;/runtimeClasspathEntry&gt;&#13;&#10;"/>
<listEntry value="&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;&#13;&#10;&lt;runtimeClasspathEntry
externalArchive=&quot;C:/gwt-windows-1.5.1/gwt-dev-windows.jar&quot; path=&quot;3&quot; type=&quot;2&quot;/&gt;&#13;&#10;"/>
</listAttribute>
<stringAttribute key="org.eclipse.jdt.launching.VM_ARGUMENTS" value=" -Xmx256M"/>
<stringAttribute key="org.eclipse.jdt.launching.PROGRAM_ARGUMENTS" value="-out www css2008.gwt.HelloWorld/HelloWorld.html"/>
<stringAttribute key="org.eclipse.jdt.launching.PROJECT_ATTR" value="HelloWorld"/>
<booleanAttribute key="org.eclipse.debug.core.appendEnvironmentVariables" value="true"/>
</launchConfiguration>
```





Pimp My Webapp

➤ HelloWorld-compile.cmd

```
@java -Xmx256M -cp "%~dp0\src;%~dp0\bin;C:/gwt-windows-1.5.1/gwt-user.jar;C:/gwt-windows-1.5.1/gwt-dev-windows.jar" com.google.gwt.dev.GWTCompiler -out "%~dp0\www" %* css2008.gwt.HelloWorld
```



Pimp My Webapp

➤ HelloWorld-shell.cmd

```
@java -Xmx256M -cp "%~dp0\src;%~dp0\bin;C:/gwt-windows-1.5.1/gwt-user.jar;C:/gwt-windows-1.5.1/gwt-dev-windows.jar" com.google.gwt.dev.GWTShell -out "%~dp0\www" %*  
css2008.gwt.HelloWorld/HelloWorld.html
```



Pimp My Webapp

➤ HelloWorld.css

```
/** Add css rules here for your application. */  
button {  
    display: block;  
    font-size: 16pt  
}  
  
.widePanel {  
    width: 100%  
}  
  
img {  
    margin-top: 20px;  
}
```



Pimp My Webapp

➤ HelloWorld.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<!-- The HTML 4.01 Transitional DOCTYPE declaration-->
<!-- above set at the top of the file will set -->
<!-- the browser's rendering engine into -->
<!-- "Quirks Mode". Replacing this declaration -->
<!-- with a "Standards Mode" doctype is supported, -->
<!-- but may lead to some differences in layout. -->

<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <!-- -->
    <!-- Any title is fine -->
    <!-- -->
    <title>HelloWorld</title>
```



Pimp My Webapp

```
<!-- -->

<!-- This script loads your compiled module. -->
<!-- If you add any GWT meta tags, they must -->
<!-- be added before this line. -->
<!-- -->
<script type="text/javascript" language="javascript" src="css2008.gwt.HelloWorld.nocache.js"></script>
</head>

<!-- -->
<!-- The body can have arbitrary html, or -->
<!-- you can leave the body empty if you want -->
<!-- to create a completely dynamic UI. -->
<!-- -->
<body>

    <!-- OPTIONAL: include this if you want history support -->
    <iframe src="javascript:''" id="__gwt_historyFrame" tabIndex='-1'
style="position:absolute;width:0;height:0;border:0"></iframe>

</body>
</html>
```



Pimp My Webapp

➤ HelloWorld.java

```
package css2008.samples.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.ClickListener;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.Widget;

/**
 * Entry point classes define onModuleLoad().
 */
public class HelloWorld implements EntryPoint {
```



Pimp My Webapp

```
/**
 * This is the entry point method.
 */
public void onModuleLoad() {
    final Button button = new Button("Click me");
    final Label label = new Label();
    button.addClickListener(new ClickListener() {
        public void onClick(Widget sender) {
            if (label.getText().equals(""))
                label.setText("Hello World!");
            else
                label.setText("");
        }
    });

    // Assume that the host HTML has elements defined whose
    // IDs are "slot1", "slot2". In a real app, you probably would not
    want
    // to hard-code IDs. Instead, you could, for example, search for all
    // elements with a particular CSS class and replace them with widgets.
    //
    RootPanel.get("slot1").add(button);
    RootPanel.get("slot2").add(label);
}
}
```



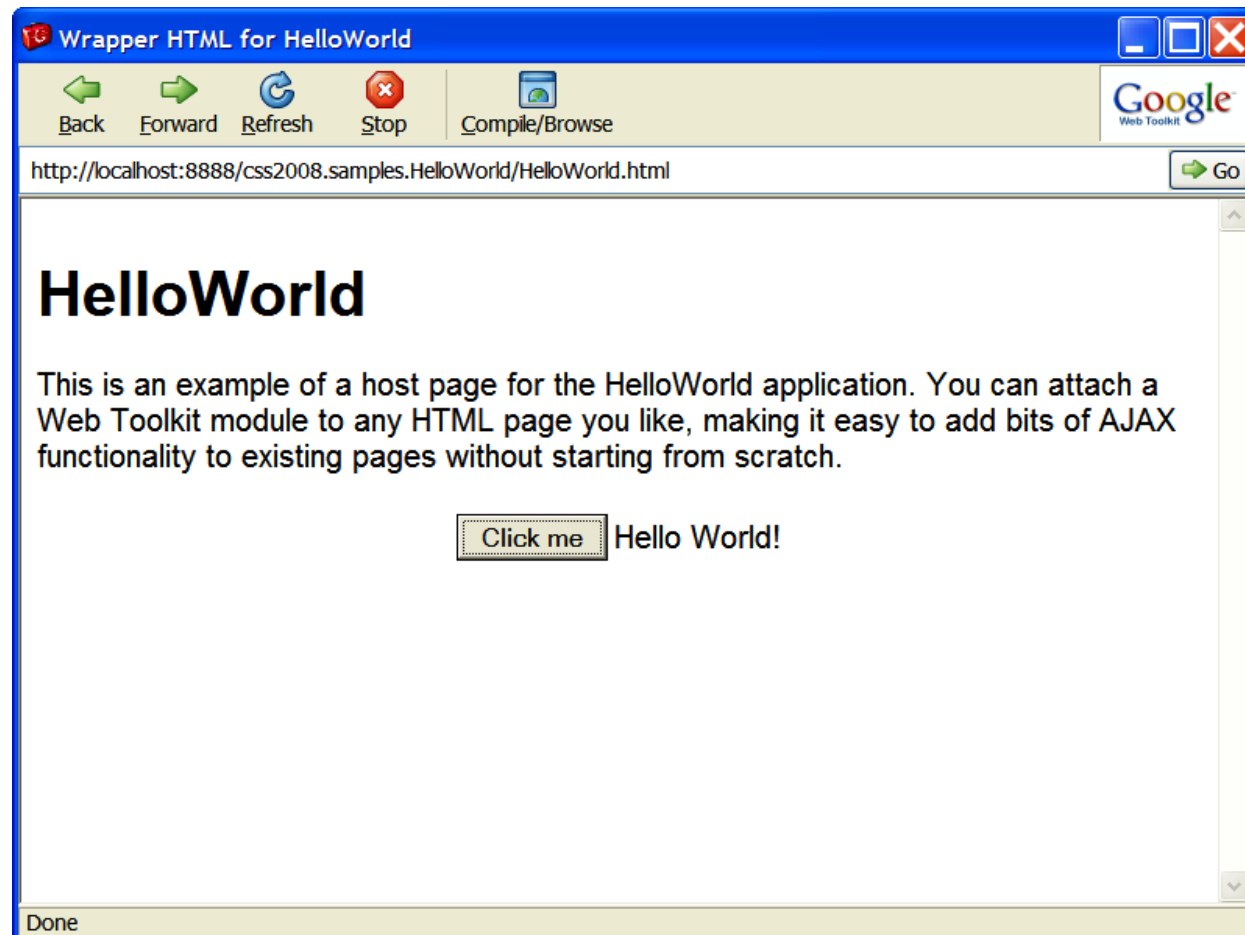
Pimp My Webapp

- Deploying

- Deploy your app to any http server, taking care to bring along all the files that were generated under the www directory.

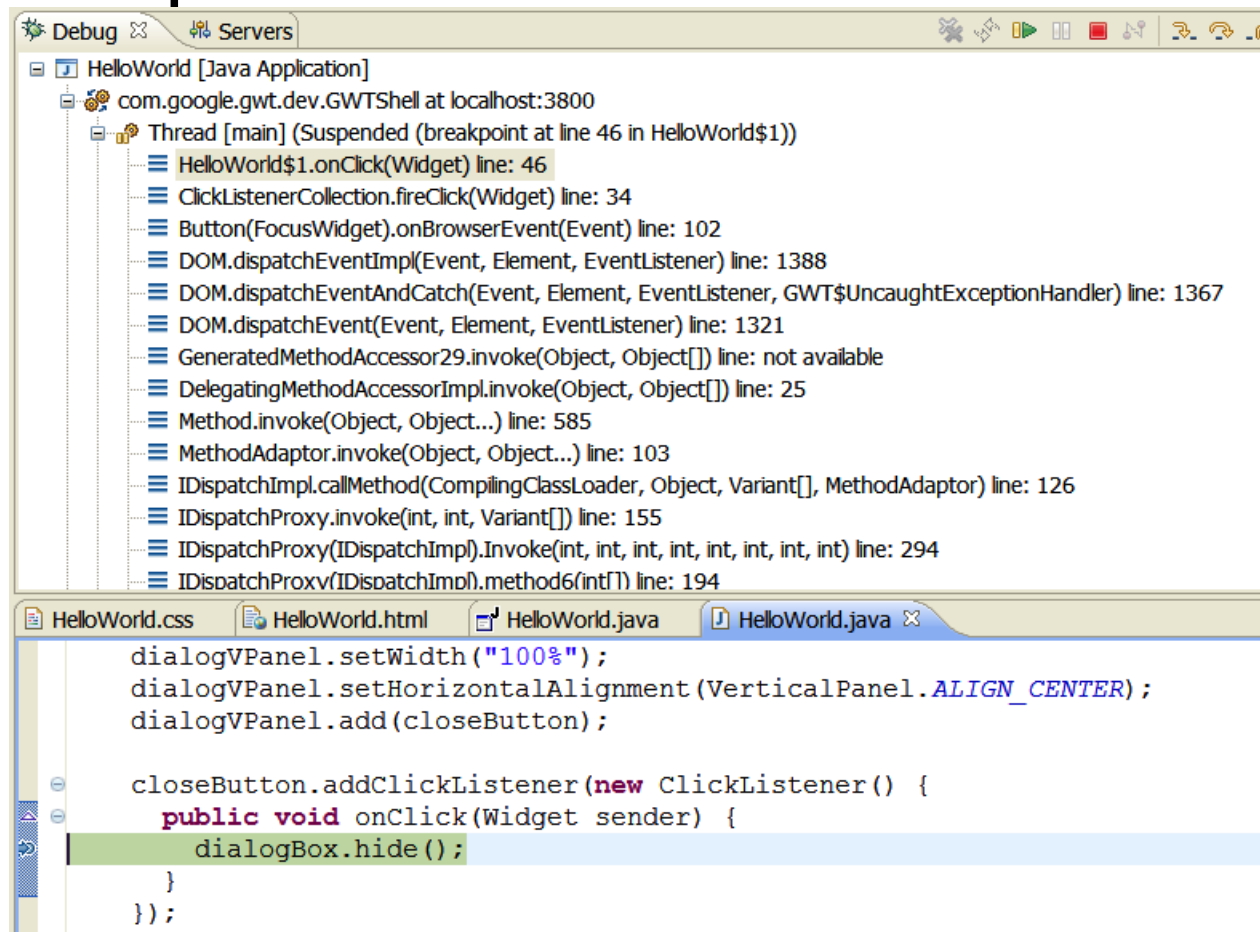
Pimp My Webapp

- The result (in hostedmode browser)



Pimp My Webapp

- Debug in Eclipse





Pimp My Webapp

- Java to JavaScript

- The following GWT Java application

```
public class Sample1 implements EntryPoint {  
    private Button clickMeButton;  
    public void onModuleLoad() {  
        RootPanel rootPanel = RootPanel.get("gwtstuffhere");  
  
        clickMeButton = new Button();  
        rootPanel.add(clickMeButton);  
        clickMeButton.setText("Click me!");  
        clickMeButton.addClickListener(new ClickListener() {  
            public void onClick(Widget sender) {  
                Window.alert("Hello, CSS 2008!");  
            }  
        });  
    }  
}
```

Pimp My Webapp

- Gets translated into the following set of files





Pimp My Webapp

- Adding 3rd party libraries
 - All 3rd party libraries must be defined in the GWT project description file (xxx.gwt.xml) using the inherits tag
 - Jar must be added to classpath

```
<module>  
<inherits name="com.google.gwt.user.User" />  
<!-- Add in 3rd party library -->  
<inherits name="org.gwtwidgets.WidgetLibrary" />  
<entry-point class="css2008.gwt.client.RequestBuilder" />  
</module>
```



Pimp My Webapp

- Adding to an existing web page
 - Start out by analyzing your existing web page, breaking it into areas of information or functionality
 - Surrounding those areas with div tags makes things simpler
 - Mark the areas (pref. Div's) with ID attributes



Pimp My Webapp

- In your code you grab the marked areas by calling the `RootPanel.get("id")` method
 - `RootPanel rootPanel = RootPanel.get("yourid");`
- Now you can create your widgets and add them to the selected tag
 - `clickMeButton = new Button();`
 - `rootPanel.add(clickMeButton);`
- This will add a button to an existing tag
- To clear it first use the clear method
 - `rootPanel.clear();`



Pimp My Webapp

- Creating reusable GWT components
 - Just like a regular Java project.
 - Compile it to a jar.
 - Must contain source.
 - Used by the JavaScript compiler.



Pimp My Webapp

- Writing Native JavaScript Methods:
 - Declare just like a regular Java method in a Java class.
 - Declare it using following syntax:

```
public static native void methodname(Javatype paramname) /*-{  
$javascriptobject. javascriptmethodname(paramname);  
}-*/;
```
 - Objects can be shared between Java source and JavaScript



Pimp My Webapp

- Internationalization – I18N
 - I18NCreator [-eclipse projectName] [-out dir] [-overwrite] [-createConstantsWithLookup] [-createMessages] [-ignore] interfaceName
 - -eclipse Creates an i18n update launch config for the named eclipse project



Pimp My Webapp

- Locale properties consist of a minimum two sets of files
 - A file extending `com.google.gwt.i18n.client.Constants` for parameterless strings.
- Or
 - A file extending `com.google.gwt.i18n.client.Messages` for parameterized messages.



Pimp My Webapp

- Samples

```
public interface AppConstants extends Constants {  
    String clickMeButton_text();  
    String hello_text();  
}
```

```
public interface AppMessages extends Messages {  
  
    String sayHelloString(String brukernavn);  
    String errorMessage(String errortext, Integer errorcode);  
}
```





Pimp My Webapp

➤ A set of localized properties files

- Constants

```
clickMeButton_text=Click me!
```

```
clickMeButton2_text=Click me to see message!
```

```
hello_text=Hello, GWT World!
```

- Messages

```
sayHelloString=Hello {0}
```

```
errorMessage=An error with errorcode: {1} and message: {0}
```



Pimp My Webapp

- Add entries to myapp.gwt.xml
 - `<extend-property name="locale" values="en" />`
 - `<extend-property name="locale" values="no" />`
 - `<extend-property name="locale" values="de" />`
 - `<extend-property name="locale" values="fr" />`

- Add localized versions of your AppConstants and AppMessages
 - AppConstants-en.properties
 - *etc.*



Pimp My Webapp

- Add entry to myapp.html to support all international characters
 - `<meta http-equiv="content-type" content="text/html; charset=UTF-8">`
- To change locale add query parameter to URL
 - `?locale=en`
- Or have multiple locale versions of myapp.xml with metatag specifying default locale
 - `<meta name="gwt:property" content="locale=en">`



Pimp My Webapp

- History support
 - A reliable way of controlling the browser's history cache.
 - Provides bookmarking.
 - Use History tokens in code.
 - History is stored in an IFRAME.
 - Must implement the HistoryListener interface.



Pimp My Webapp

- HistorySample



Pimp My Webapp

- Deferred binding
 - Much like Java's reflection
 - Used to overcome the different browsers' ways of implementing JavaScript and rendering.
 - Happens at compile time.
 - Results in different script files for different browsers.
 - Internationalization uses it.



Pimp My Webapp

- Talking to a server

- RPC

- GWT built-in mechanisms

- ✓ AsyncCallback

- ❖ 2 client side files (interfaces)

- ❖ Clientside interface with two methods

- ❖ -onSuccess

- ❖ -onFailure

- ❖ Serverside servlet(s) (define in web.xml)–
Extends RemoteServiceServlet

- ❖ Serialization policy file



Pimp My Webapp

- RPCSample



Pimp My Webapp

➤ RequestBuilder

- Post and Get
- You build the query string
- SOP (Same origin policy)



Pimp My Webapp

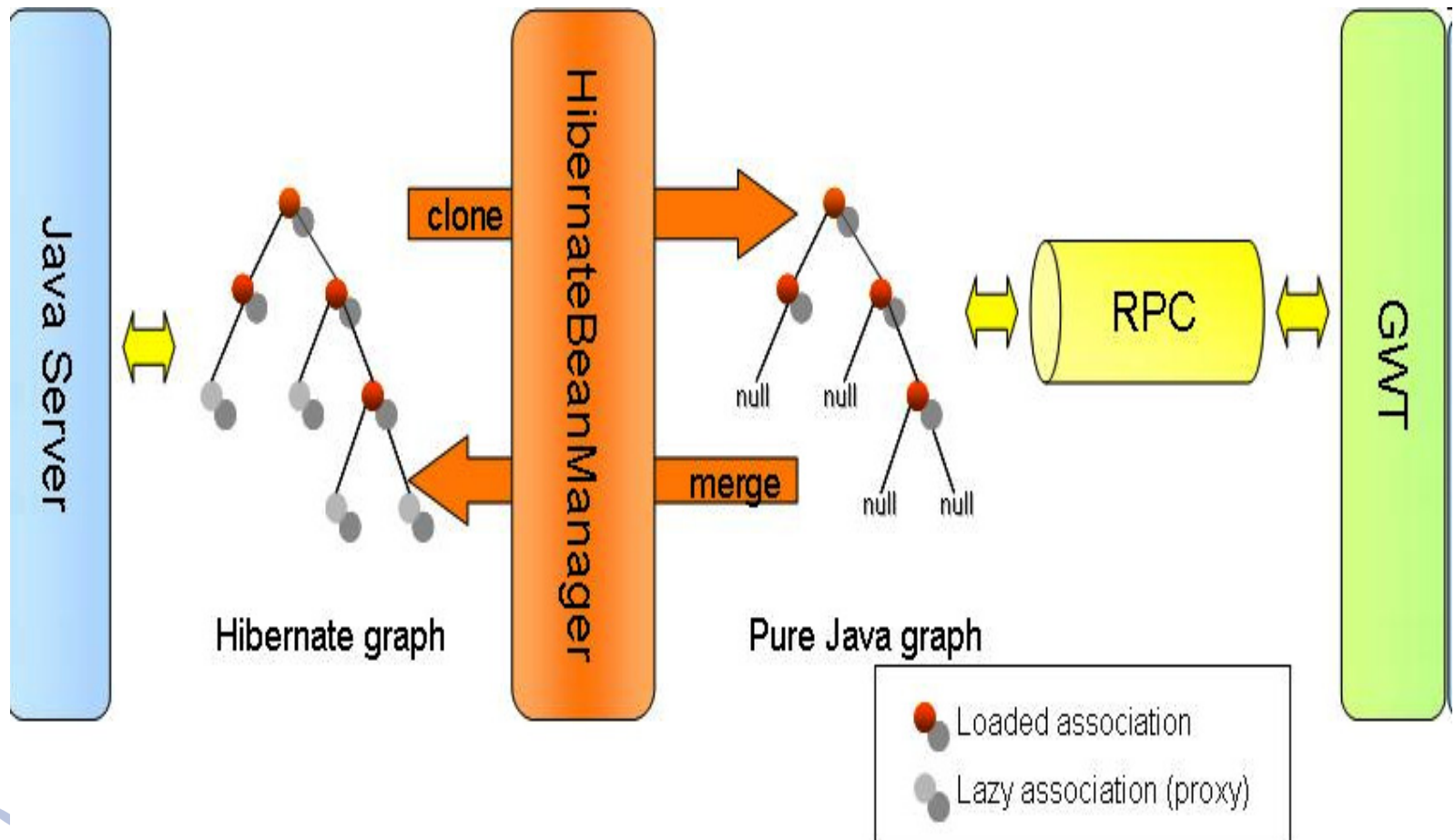
- RequestBuilderSample



Pimp My Webapp

- Integrating with Hibernate
 - Several libraries available
 - Hibernate4GWT + GWT-SL + Spring
 - Supports Lazy POJOS.
 - Removes Hibernate specific things (proxies) from POJOS.
 - HibernateBeanManager – The class that does the dirtywork.
 - Hibernate domainbeans must extend LazyGwtPojo

Pimp My Webapp





Pimp My Webapp

- GWT-SL provides the mapping between the incoming request and business POJO method to be called.
- Uses
`org.springframework.web.servlet.handler.
SimpleUrlHandlerMapping`
- Business POJOS are managed by
`org.gwtwidgets.server.spring.hb4gwt.HB4
GWTRPCServiceExporter`



Pimp My Webapp

- HibernateSample



Pimp My Webapp

- Integrating with Acegi
 - Uses same setup as for integrating with Hibernate (Spring)
 - Acegi adds cookie to session which GWT remembers and adds to all requests to the server.



Pimp My Webapp

- AcegiSample



Pimp My Webapp

- Pimping the application
 - Adding a better menu
 - Refreshing parts of the page
 - *etc.*



Pimp My Webapp

- Debug in production!
 - To debug an application in production is not always that easy.
 - The hosted mode browser and an IDE makes makes it possible.
 - Demo



Pimp My Webapp

- Alternatives

- Many RIA's out there

- Opensource

- ✓ XAP

- ✓ ThinWire

- ✓ *etc.*

- Commercial

- ✓ Flex

- ✓ Backbase

- ✓ *etc.*



Pimp My Webapp

- According to Gartner, by 2010 some 60% of new applications will be RIAs, creating a richer experience for the end user.



Pimp My Webapp

- References:

- <http://code.google.com/webtoolkit/>
- <http://gwt-widget.sourceforge.net/?q=node/51>
- <http://hibernate4gwt.sourceforge.net/index.html>
- <http://code.google.com/p/gwt-ent/wiki/IntegrationGWTWithAcegi>



Pimp My Webapp





Pimp My Webapp

- Remember to fill out the evaluations
- All sample code will be on the post-conference CD.