Building Web Applications with GWT

Solving common application problems and improving productivity with Google's Web Toolkit

Douglas Bullard

September 16, 2008

What is GWT?

What is GWT?

- Google Web Toolkit (GWT) is a framework for creating rich web applications that run in any browser with no plugins required.

What is GWT?

- Google Web Toolkit (GWT) is a framework for creating rich web applications that run in any browser with no plugins required.
  - GWT applications are

What is GWT?

- Google Web Toolkit (GWT) is a framework for creating rich web applications that run in any browser with no plugins required.
  - GWT applications are
  - Written in Java

What is GWT?

- Google Web Toolkit (GWT) is a framework for creating rich web applications that run in any browser with no plugins required.
  - GWT applications are
    - Written in Java
    - Use CSS for display formatting

What is GWT?

- Google Web Toolkit (GWT) is a framework for creating rich web applications that run in any browser with no plugins required.
  - GWT applications are
    - Written in Java
    - Use CSS for display formatting
    - Are converted to JavaScript for runtime

What is GWT?

- Google Web Toolkit (GWT) is a framework for creating rich web applications that run in any browser with no plugins required.
  - GWT applications are
    - Written in Java
    - Use CSS for display formatting
    - Are converted to JavaScript for runtime
    - Are debuggable in Java with your favorite IDE

What is GWT?

- Web applications with a rich client experience

What is GWT?

- Web applications with a rich client experience
- Works with existing JavaScript libraries

What is GWT?

- Web applications with a rich client experience
  - Works with existing JavaScript libraries
  - Works with your existing JavaScript

What is GWT?

- Web applications with a rich client experience
  - Works with existing JavaScript libraries
  - Works with your existing JavaScript
- Handles browser compatibility issues

What is GWT?

- Web applications with a rich client experience
  - Works with existing JavaScript libraries
  - Works with your existing JavaScript
  - Handles browser compatibility issues
  - Works with other web frameworks (Seam, Struts, Spring, etc)

What is GWT?

- Web applications with a rich client experience
  - Works with existing JavaScript libraries
  - Works with your existing JavaScript
  - Handles browser compatibility issues
  - Works with other web frameworks (Seam, Struts, Spring, etc)
  - Allows for simplification of our front-end technology stack

Rich Client Applications

Rich Client Applications

- Everybody likes rich clients on web - except developers

Rich Client Applications

- Everybody likes rich clients on web - except developers
  - Why aren't more web apps using rich clients?

Rich Client Applications

- Everybody likes rich clients on web - except developers
  - Why aren't more web apps using rich clients?
    - UI interaction is hard to code

Rich Client Applications

- Everybody likes rich clients on web - except developers
  - Why aren't more web apps using rich clients?
    - UI interaction is hard to code
    - Mixing of JavaScript, JSP, Java to get needed functionality

Rich Client Applications

- Everybody likes rich clients on web - except developers
  - Why aren't more web apps using rich clients?
    - UI interaction is hard to code
      - Mixing of JavaScript, JSP, Java to get needed functionality
      - Usually isn't done - server side validation and error detection requires round-trip and page refresh

Rich Client Applications

- Everybody likes rich clients on web - except developers
  - Why aren't more web apps using rich clients?
    - UI interaction is hard to code
      - Mixing of JavaScript, JSP, Java to get needed functionality
      - Usually isn't done - server side validation and error detection requires round-trip and page refresh
    - UI interaction is hard to debug

Rich Client Applications

- GWT makes rich client experiences easy

Rich Client Applications

- GWT makes rich client experiences easy
  - UI design is done in Java, like Swing

Rich Client Applications

- GWT makes rich client experiences easy
  - UI design is done in Java, like Swing
    - UI components (buttons, dropdowns, labels, etc)

Rich Client Applications

- GWT makes rich client experiences easy
  - UI design is done in Java, like Swing
    - UI components (buttons, dropdowns, labels, etc)
    - Reusable panels for layout

Rich Client Applications

- GWT makes rich client experiences easy
  - UI design is done in Java, like Swing
    - UI components (buttons, dropdowns, labels, etc)
    - Reusable panels for layout
  - Events are all done in Java

Rich Client Applications

- GWT makes rich client experiences easy
  - UI design is done in Java, like Swing
    - UI components (buttons, dropdowns, labels, etc)
    - Reusable panels for layout
  - Events are all done in Java
    - Add event listeners, just like Swing

Rich Client Applications

- GWT makes rich client experiences easy
  - UI design is done in Java, like Swing
    - UI components (buttons, dropdowns, labels, etc)
    - Reusable panels for layout
  - Events are all done in Java
    - Add event listeners, just like Swing
    - All UI interaction can be debugged in Java

GWT and Java Emulation

GWT and Java Emulation

- GWT doesn't really run Java on the client side - it runs JavaScript.

GWT and Java Emulation

- GWT doesn't really run Java on the client side - it runs JavaScript.
- GWT converts selected libraries to JavaScript

GWT and Java Emulation

- GWT doesn't really run Java on the client side - it runs JavaScript.
- GWT converts selected libraries to JavaScript
- What version of Java does GWT support?

GWT and Java Emulation

- GWT doesn't really run Java on the client side - it runs JavaScript.
- GWT converts selected libraries to JavaScript
- What version of Java does GWT support?
  - GWT supports Java 5, with generics and annotations

GWT and Java Emulation

- GWT doesn't really run Java on the client side - it runs JavaScript.
- GWT converts selected libraries to JavaScript
- What version of Java does GWT support?
  - GWT supports Java 5, with generics and annotations
    - The GWT JRE emulation engine supports a limited subset of Java, but includes collections, arrays, dates, and many other types

GWT and Java Emulation

- GWT supports most of java.lang, java.lang.annotation, java.util, parts of java.io and java.sql

GWT and Java Emulation

- GWT supports most of java.lang, java.lang.annotation, java.util, parts of java.io and java.sql
- Full JRE emulation list is at: http://code.google.com/docreader/#p=google-web-toolkit-doc-1-5&s=google-web-toolkit-doc-1-5&t=RefJreEmulation

Basic GWT App Design

Basic GWT App Design

- Here's a basic web page, where everything is done in GWT:

```
<html>
    <head>
        <title>Catalog Data Management</title>
        <meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
        <link rel="stylesheet" href="ExampleApp.css">
    </head>
    <body>
        <script type="text/javascript" language="javascript" src="com.abc.ExampleApp.nocache.js"></script>
        <!-- OPTIONAL: include this if you want history support -->
        <iframe src="javascript:'' id="__gwt_historyFrame" style="width:0;height:0;border:0"/>
    </body>
</html>
```

Basic GWT App Design

- Here's a basic web page, where everything is done in GWT:

```
<html>
    <head>
        <title>Catalog Data Management</title>
        <meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
        <link rel="stylesheet" href="ExampleApp.css">
    </head>
    <body>
        <script type="text/javascript" language="javascript" src="com.abc.ExampleApp.nocache.js"></script>
        <!-- OPTIONAL: include this if you want history support -->
        <iframe src="javascript:''" id="__gwt_historyFrame" style="width:0;height:0;border:0"/>
    </body>
</html>
```

Basic GWT App Design

- Here's a simple class to put a button on the page:

Basic GWT App Design

- Here's a simple class to put a button on the page:

```
public class GWTPresentationApp implements EntryPoint
{
    public void onModuleLoad()
    {
        Panel panel = new FlowPanel();
        RootPanel.get().add(panel);
        final Button button = new Button("Click me");
        panel.add(button);
        button.addStyleName("clickButton");

        button.addClickListener(new ClickListener()
        {
            public void onClick(Widget sender)
            {
                label.setText(label.getText()+" Click!");
            }
        });
    }
}
```

Basic GWT App Design

- Here's a simple class to put a button on the page:

```
public class GWTPresentationApp implements EntryPoint
{
    public void onModuleLoad()
    {
        Panel panel = new FlowPanel();
        RootPanel.get().add(panel);
        final Button button = new Button("Click me");
        panel.add(button);
        button.addStyleName("clickButton");

        button.addClickListener(new ClickListener()
        {
            public void onClick(Widget sender)
            {
                label.setText(label.getText()+" Click!");
            }
        });
    }
}
```

Basic GWT App Design

- Here's a simple class to put a button on the page:

```
public class GWTPresentationApp implements EntryPoint
{
    public void onModuleLoad()
    {
        Panel panel = new FlowPanel();
        RootPanel.get().add(panel);
        final Button button = new Button("Click me");
        panel.add(button);
        button.addStyleName("clickButton");

        button.addClickListener(new ClickListener()
        {
            public void onClick(Widget sender)
            {
                label.setText(label.getText()+" Click!");
            }
        });
    }
}
```

Basic GWT App Design

- Here's a simple class to put a button on the page:

```
public class GWTPresentationApp implements EntryPoint
{
    public void onModuleLoad()
    {
        Panel panel = new FlowPanel();
        RootPanel.get().add(panel);
        final Button button = new Button("Click me");
        panel.add(button);
        button.addStyleName("clickButton");

        button.addClickListener(new ClickListener()
        {
            public void onClick(Widget sender)
            {
                label.setText(label.getText()+" Click!");
            }
        });
    }
}
```

Basic GWT App Design

- Here's a simple class to put a button on the page:

```
public class GWTPresentationApp implements EntryPoint
{
    public void onModuleLoad()
    {
        Panel panel = new FlowPanel();
        RootPanel.get().add(panel);
        final Button button = new Button("Click me");
        panel.add(button);
        button.addStyleName("clickButton");

        button.addClickListener(new ClickListener()
        {
            public void onClick(Widget sender)
            {
                label.setText(label.getText()+" Click!");
            }
        });
    }
}
```

Basic GWT App Design

- Here's a simple class to put a button on the page:

```
public class GWTPresentationApp implements EntryPoint
{
    public void onModuleLoad()
    {
        Panel panel = new FlowPanel();
        RootPanel.get().add(panel);
        final Button button = new Button("Click me");
        panel.add(button);
        button.addStyleName("clickButton");

        button.addClickListener(new ClickListener()
        {
            public void onClick(Widget sender)
            {
                label.setText(label.getText()+" Click!");
            }
        });
    }
}
```

Basic GWT App Design

- Here's another basic web page, where layout is done in the web page itself (using GWT with Struts/JSP/etc):

Basic GWT App Design

- Here's another basic web page, where layout is done in the web page itself (using GWT with Struts/JSP/etc):

```
<html>
    <head>
        <title>Catalog Data Management</title>
        <meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
        <link rel="stylesheet" href="ExampleApp.css">
    </head>
    <body>
        <script type="text/javascript" language="javascript" src="com.abc.ExampleApp.nocache.js"></scr
        <!-- OPTIONAL: include this if you want history support -->
        <iframe src="javascript:''" id="__gwt_historyFrame" style="width:0;height:0;border:0"/>
        <table align=center>
            <tr>
                <td id="slot1"></td><td id="slot2"></td>
            </tr>
            <tr>
                <td id="slot3"></td><td id="slot4"></td>
            </tr>
        </table>
    </body>
</html>
```

Basic GWT App Design

- Here's another basic web page, where layout is done in the web page itself (using GWT with Struts/JSP/etc):

```
<html>
    <head>
        <title>Catalog Data Management</title>
        <meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
        <link rel="stylesheet" href="ExampleApp.css">
    </head>
    <body>
        <script type="text/javascript" language="javascript" src="com.abc.ExampleApp.nocache.js"></scr
        <!-- OPTIONAL: include this if you want history support -->
        <iframe src="javascript:''" id="__gwt_historyFrame" style="width:0;height:0;border:0"/>
        <table align=center>
            <tr>
                <td id="slot1"></td><td id="slot2"></td>
            </tr>
            <tr>
                <td id="slot3"></td><td id="slot4"></td>
            </tr>
        </table>
    </body>
</html>
```

Basic GWT App Design

- Here's the class to put things into those slots:

Basic GWT App Design

- Here's the class to put things into those slots:

```
public class GWTPresentationApp implements EntryPoint
{
    public void onModuleLoad()
    {
        final Button button = new Button("Click me");
        RootPanel.get("slot1").add(button);
        button.addStyleName("clickButton");

        RootPanel.get("slot2").add(new Button("Me, too!");
        RootPanel.get("slot3").add(new Button("Me, three!");
        RootPanel.get("slot4").add(new Button("Don't forget me!");

        button.addClickListener(new ClickListener()
        {
            public void onClick(Widget sender)
            {
                label.setText(label.getText()+" Click!");
            }
        });
    }
}
```

Basic GWT App Design

- Here's the class to put things into those slots:

```
public class GWTPresentationApp implements EntryPoint
{
    public void onModuleLoad()
    {
        final Button button = new Button("Click me");
        RootPanel.get("slot1").add(button);
        button.addStyleName("clickButton");

        RootPanel.get("slot2").add(new Button("Me, too!");
        RootPanel.get("slot3").add(new Button("Me, three!");
        RootPanel.get("slot4").add(new Button("Don't forget me!");

        button.addClickListener(new ClickListener()
        {
            public void onClick(Widget sender)
            {
                label.setText(label.getText()+" Click!");
            }
        });
    }
}
```

Remote Procedure Calls in GWT

Remote Procedure Calls in GWT

- GWT talks to the server via remote procedure calls (RPC)

Remote Procedure Calls in GWT

- GWT talks to the server via remote procedure calls (RPC)
- Results from these calls are returned asynchronously

Remote Procedure Calls in GWT

- GWT talks to the server via remote procedure calls (RPC)
- Results from these calls are returned asynchronously
- No page refresh results from the call - JavaScript and AJAX are used to update the desired component

Remote Procedure Calls in GWT

- GWT talks to the server via remote procedure calls (RPC)
- Results from these calls are returned asynchronously
- No page refresh results from the call - JavaScript and AJAX are used to update the desired component
- Anything can be used to back up the RPC - Spring, Seam, EJB, etc.

Simple RPC example

Simple RPC example

- Our button example now gets it's text from an RPC:

Simple RPC example

```
public class GWTPresentationApp implements EntryPoint
{
    public void onModuleLoad()
    {
        Panel panel = new FlowPanel();
        RootPanel.get().add(panel);
        final Button button = new Button("Click me");
        panel.add(button);
        button.addStyleName("clickButton");

        button.addClickListener(new ClickListener()
        {
            public void onClick(Widget sender)
            {
                ButtonService.App.getInstance().getText(new ButtonAsyncCallback(button));
            }
        });
     }
    class ButtonAsyncCallback implements AsyncCallback
     {
        private Button button;

        ButtonAsyncCallback(Button button)
        {
            this. button = button;
        }

        public void onFailure(Throwable caught) { }

        public void onSuccess(Object result)
        {
            String value= (String) result;
            button.setText(value);
        }
     }
}
```

Simple RPC example

```
public class GWTPresentationApp implements EntryPoint
{
    public void onModuleLoad()
    {
        Panel panel = new FlowPanel();
        RootPanel.get().add(panel);
        final Button button = new Button("Click me");
        panel.add(button);
        button.addStyleName("clickButton");

        button.addClickListener(new ClickListener()
        {
            public void onClick(Widget sender)
            {
                ButtonService.App.getInstance().getText(new ButtonAsyncCallback(button));
            }
        });
     }
    class ButtonAsyncCallback implements AsyncCallback
     {
        private Button button;

        ButtonAsyncCallback(Button button)
        {
            this. button = button;
        }

        public void onFailure(Throwable caught) { }

        public void onSuccess(Object result)
        {
            String value= (String) result;
            button.setText(value);
        }
     }
}
```

Simple RPC example

```
public class GWTPresentationApp implements EntryPoint
{
    public void onModuleLoad()
    {
        Panel panel = new FlowPanel();
        RootPanel.get().add(panel);
        final Button button = new Button("Click me");
        panel.add(button);
        button.addStyleName("clickButton");

        button.addClickListener(new ClickListener()
        {
            public void onClick(Widget sender)
            {
                ButtonService.App.getInstance().getText(new ButtonAsyncCallback(button));
            }
        });
     }
    class ButtonAsyncCallback implements AsyncCallback
     {
        private Button button;

        ButtonAsyncCallback(Button button)
        {
            this. button = button;
        }

        public void onFailure(Throwable caught) { }

        public void onSuccess(Object result)
        {
            String value= (String) result;
            button.setText(value);
        }
     }
}
```

Simple RPC example

```
public class GWTPresentationApp implements EntryPoint
{
    public void onModuleLoad()
    {
        Panel panel = new FlowPanel();
        RootPanel.get().add(panel);
        final Button button = new Button("Click me");
        panel.add(button);
        button.addStyleName("clickButton");

        button.addClickListener(new ClickListener()
        {
            public void onClick(Widget sender)
            {
                ButtonService.App.getInstance().getText(new ButtonAsyncCallback(button));
            }
        });
     }
    class ButtonAsyncCallback implements AsyncCallback
     {
        private Button button;

        ButtonAsyncCallback(Button button)
        {
            this. button = button;
        }

        public void onFailure(Throwable caught) { }

        public void onSuccess(Object result)
        {
            String value= (String) result;
            button.setText(value);
        }
     }
}
```

RPC Code

RPC Code

- What does the service layer look like?

RPC Code

- What does the service layer look like?

```
public interface ButtonServiceAsync
{
    void getText(AsyncCallback async);
}
```

RPC Code

- What does the service layer look like?

```
public interface ButtonServiceAsync
{
    void getText(AsyncCallback async);
}
```

RPC Code

- What does the service layer look like?

```
public interface ButtonService extends RemoteService
{

    String getText();

    /** Utility/Convenience class. Use ButtonService.App.getInstance() to access static instance
        of ButtonServiceAsync */
    class App
    {
        private static final ButtonServiceAsync ourInstance;

        private App()
        {
        }

        static
        {
            ourInstance = (ButtonServiceAsync) GWT.create(ButtonService.class);
            ((ServiceDefTarget) ourInstance).setServiceEntryPoint(GWT.getModuleBaseURL()
              +  "com.abc.cdm.ExampleApp/ButtonService");
        }

        public static ButtonServiceAsync getInstance()
        {
            return ourInstance;
        }
    }
}
```

RPC Code

- What does the service layer look like?

```
public interface ButtonService extends RemoteService
{

    String getText();

    /** Utility/Convenience class. Use ButtonService.App.getInstance() to access static instance
        of ButtonServiceAsync */
    class App
    {
        private static final ButtonServiceAsync ourInstance;

        private App()
        {
        }

        static
        {
            ourInstance = (ButtonServiceAsync) GWT.create(ButtonService.class);
            ((ServiceDefTarget) ourInstance).setServiceEntryPoint(GWT.getModuleBaseURL()
              +  "com.abc.cdm.ExampleApp/ButtonService");
        }

        public static ButtonServiceAsync getInstance()
        .
            return ourInstance;
        }
    }
}
```

RPC Code

- What does the service layer look like?

```
public class ButtonServiceImpl extends ButtonServiceServlet implements ButtonService
{
    private static final long   serialVersionUID = -21260570371947985591;

    public String getText() { return "SomeText"; }
}
```

RPC Code

- What does the service layer look like?

```
public class ButtonServiceImpl extends ButtonServiceServlet implements ButtonService
{
    private static final long   serialVersionUID = -21260570371947985591;

    public String getText() { return "SomeText"; }
}
```

RPC Code

- What does the service layer look like?

ButtonApp.gwt.xml

```
<module>
    <inherits name='com.google.gwt.user.User'/>

    <entry-point class='com.abc.cdm.client.ButtonApp'/>

    <servlet path="/com.abc.cdm.ExampleApp/ButtonService" class="com.abc.cdm.server.ButtonServiceImpl"/>

</module>
```

RPC Code

- What does the service layer look like?

ButtonApp.gwt.xml

```
<module>
    <inherits name='com.google.gwt.user.User'/>

    <entry-point class='com.abc.cdm.client.ButtonApp'/>

    <servlet path="/com.abc.cdm.ExampleApp/ButtonService" class="com.abc.cdm.server.ButtonServiceImpl"/>

</module>
```

RPC Code

- That seems pretty ugly, and a lot of hand-coding

RPC Code

- That seems pretty ugly, and a lot of hand-coding
  - Not really - GWT generates all the RPC stuff for you.

RPC Code

- That seems pretty ugly, and a lot of hand-coding
  - Not really - GWT generates all the RPC stuff for you.
  - All the IDEs call GWT to generate this via their plugins

RPC Code

- That seems pretty ugly, and a lot of hand-coding
  - Not really - GWT generates all the RPC stuff for you.
  - All the IDEs call GWT to generate this via their plugins
  - GWT provides command-line executables for generating them

RPC Code

- That seems pretty ugly, and a lot of hand-coding
  - Not really - GWT generates all the RPC stuff for you.
  - All the IDEs call GWT to generate this via their plugins
  - GWT provides command-line executables for generating them
  - GWT even creates an entire sample project for you

Widgets

Widgets

- What widgets does GWT come with?

Widgets

- What widgets does GWT come with?
  - Basic layout and component widgets (panels, buttons, dropdowns, listboxes, etc).

Widgets

- What widgets does GWT come with?
  - Basic layout and component widgets (panels, buttons, dropdowns, listboxes, etc).
- What if I want something more?

Widgets

- What widgets does GWT come with?
  - Basic layout and component widgets (panels, buttons, dropdowns, listboxes, etc).
- What if I want something more?
  - EXT-JS is a major supplier of JavaScript widgets

Widgets

- What widgets does GWT come with?
  - Basic layout and component widgets (panels, buttons, dropdowns, listboxes, etc).
- What if I want something more?
  - EXT-JS is a major supplier of JavaScript widgets
    - GWT-EXT-JS is a GWT port of their JavaScript widgets - all written in Java, with source code available

Widgets

- What widgets does GWT come with?
  - Basic layout and component widgets (panels, buttons, dropdowns, listboxes, etc).
- What if I want something more?
  - EXT-JS is a major supplier of JavaScript widgets
    - GWT-EXT-JS is a GWT port of their JavaScript widgets - all written in Java, with source code available
  - Many more component vendors and open-source projects to choose from

GWT and 3rd Party GWT Libraries

GWT and 3rd Party GWT Libraries

- How do we use someone else's GWT libraries? I'll use GWT-EXT-JS as an example

GWT and 3rd Party GWT Libraries

- How do we use someone else's GWT libraries? I'll use GWT-EXT-JS as an example
  - Add the line
    ```
    <inherits name='com.extjs.gxt.ui.GXT'/>
    ```
    to the YourApp.gwt.xml config file

GWT and 3rd Party GWT Libraries

- How do we use someone else's GWT libraries? I'll use GWT-EXT-JS as an example
  - Add the line
    ```
    <inherits name='com.extjs.gxt.ui.GXT'/>
    ```
    to the YourApp.gwt.xml config file
  - Add the line
    ```
    <link rel="stylesheet" type="text/css" href="css/ext-all.css" />
    ```
    to your HTML page

GWT and 3rd Party GWT Libraries

- How do we use someone else's GWT libraries? I'll use GWT-EXT-JS as an example
  - Add the line
    ```
    <inherits name='com.extjs.gxt.ui.GXT'/>
    ```
    to the YourApp.gwt.xml config file
  - Add the line
    ```
    <link rel="stylesheet" type="text/css" href="css/ext-all.css" />
    ```
    to your HTML page
  - Add the GXT.jar to your project's library dependencies

GWT and 3rd Party GWT Libraries

- How do we use someone else's GWT libraries? I'll use GWT-EXT-JS as an example
  - Add the line
    ```
    <inherits name='com.extjs.gxt.ui.GXT'/>
    ```
    to the YourApp.gwt.xml config file
  - Add the line
    ```
    <link rel="stylesheet" type="text/css" href="css/ext-all.css" />
    ```
    to your HTML page
  - Add the GXT.jar to your project's library dependencies
  - That's it!

GWT and 3rd Party JavaScript Libraries

GWT and 3rd Party JavaScript Libraries

- How do we use someone else's JavaScript libraries?

GWT and 3rd Party JavaScript Libraries

- How do we use someone else's JavaScript libraries?
- In the GWT module xml file, add

```
<script src="runtime/lib/aw.js"></script>
```

GWT and 3rd Party JavaScript Libraries

- How do we use someone else's JavaScript libraries?
  - In the GWT module xml file, add

```
<script src="runtime/lib/aw.js"></script>
```

  - Their JavaScript is injected into your page

GWT and 3rd Party JavaScript Libraries

- How do we use someone else's JavaScript libraries?
  - In the GWT module xml file, add

```
<script src="runtime/lib/aw.js"></script>
```

  - Their JavaScript is injected into your page
  - Or, add the JavaScript into your HTML as normal

GWT and 3rd Party JavaScript Libraries

- How does a GWT method talk to someone else's JavaScript libraries?

GWT and 3rd Party JavaScript Libraries

- How does a GWT method talk to someone else's JavaScript libraries?
  - Create a JavaScript Native Interface method (JSNI):

GWT and 3rd Party JavaScript Libraries

- How does a GWT method talk to someone else's JavaScript libraries?
  - Create a JavaScript Native Interface method (JSNI):

```
native JavaScriptObject init(JavaScriptObject
        myColumns,JavaScriptObject myData)/*-{
  try{
    $wnd.mygrid = new $wnd.AW.UI.Grid;
    $wnd.mygrid.setSize(750, 350);
    $wnd.mygrid.setRowCount(myData.length);
    $wnd.mygrid.setColumnCount(myColumns.length);
    return $wnd.mygrid;
  }
  catch(e){
    $wnd.alert(e.description);
  }
  return null;
}-*/;
```

GWT and 3rd Party JavaScript Libraries

- How does a GWT method talk to someone else's JavaScript libraries?
  - Create a JavaScript Native Interface method (JSNI):

```
native JavaScriptObject init(JavaScriptObject
        myColumns,JavaScriptObject myData)/*-{
  try{
    $wnd.mygrid = new $wnd.AW.UI.Grid;
    $wnd.mygrid.setSize(750, 350);
    $wnd.mygrid.setRowCount(myData.length);
    $wnd.mygrid.setColumnCount(myColumns.length);
    return $wnd.mygrid;
  }
  catch(e){
    $wnd.alert(e.description);
  }
  return null;
}-*/;
```

GWT and 3rd Party JavaScript Libraries

- How does a GWT method talk to someone else's JavaScript libraries?
  - Create a JavaScript Native Interface method (JSNI):

```
native JavaScriptObject init(JavaScriptObject
        myColumns,JavaScriptObject myData)/*-{
  try{
    $wnd.mygrid = new $wnd.AW.UI.Grid;
    $wnd.mygrid.setSize(750, 350);
    $wnd.mygrid.setRowCount(myData.length);
    $wnd.mygrid.setColumnCount(myColumns.length);
    return $wnd.mygrid;
  }
  catch(e){
    $wnd.alert(e.description);
  }
  return null;
}-*/;
```

GWT and 3rd Party JavaScript Libraries

- How does a GWT method talk to someone else's JavaScript libraries?
  - Create a JavaScript Native Interface method (JSNI):

```
native JavaScriptObject init(JavaScriptObject
        myColumns,JavaScriptObject myData)/*-{
  try{
    $wnd.mygrid = new $wnd.AW.UI.Grid;
    $wnd.mygrid.setSize(750, 350);
    $wnd.mygrid.setRowCount(myData.length);
    $wnd.mygrid.setColumnCount(myColumns.length);
    return $wnd.mygrid;
  }
  catch(e){
    $wnd.alert(e.description);
  }
  return null;
}-*/;
```

GWT and 3rd Party JavaScript Libraries

- JSNI can be used to:

GWT and 3rd Party JavaScript Libraries

- JSNI can be used to:
  - Implement a Java method directly in JavaScript.

GWT and 3rd Party JavaScript Libraries

- JSNI can be used to:
  - Implement a Java method directly in JavaScript.
  - Wrap type-safe Java method signatures around existing JavaScript.

GWT and 3rd Party JavaScript Libraries

- JSNI can be used to:
  - Implement a Java method directly in JavaScript.
  - Wrap type-safe Java method signatures around existing JavaScript.
  - Call from JavaScript into Java code and vice versa.

GWT and 3rd Party JavaScript Libraries

- JSNI can be used to:
  - Implement a Java method directly in JavaScript.
  - Wrap type-safe Java method signatures around existing JavaScript.
  - Call from JavaScript into Java code and vice versa.
  - Throw exceptions across Java/JavaScript boundaries.

GWT and 3rd Party JavaScript Libraries

- JSNI can be used to:
  - Implement a Java method directly in JavaScript.
  - Wrap type-safe Java method signatures around existing JavaScript.
  - Call from JavaScript into Java code and vice versa.
  - Throw exceptions across Java/JavaScript boundaries.
  - Read and write Java fields from JavaScript.

GWT and 3rd Party JavaScript Libraries

- JSNI can be used to:
  - Implement a Java method directly in JavaScript.
  - Wrap type-safe Java method signatures around existing JavaScript.
  - Call from JavaScript into Java code and vice versa.
  - Throw exceptions across Java/JavaScript boundaries.
  - Read and write Java fields from JavaScript.
  - Use hosted mode to debug both Java source (with a Java debugger) and JavaScript (with a script debugger).

Internationalization with GWT

Internationalization with GWT

- Two methods of doing this in GWT

Internationalization with GWT

- Two methods of doing this in GWT
  - Constant's interface

Internationalization with GWT

- Two methods of doing this in GWT
  - Constant's interface
  - Message's interface

Internationalization: Constant's Interface

Internationalization: Constant's Interface

- Create property files (for this example, the base file is called ExampleAppConstants.properties):

Internationalization: Constant's Interface

- Create property files (for this example, the base file is called ExampleAppConstants.properties):

```
appTitle    = File Management Tool
editLabel   = Edit File
```

Internationalization: Constant's Interface

- Create property files (for this example, the base file is called ExampleAppConstants.properties):

```
appTitle    = File Management Tool
editLabel   = Edit File
```

- Create an interface called ExampleAppConstants:

Internationalization: Constant's Interface

- Create property files (for this example, the base file is called ExampleAppConstants.properties):

```
appTitle   = File Management Tool
editLabel  = Edit File
```

- Create an interface called ExampleAppConstants:

```
public interface ExampleAppConstants extends Constants {
    String appTitle();
    String editLabel();
}
```

Internationalization: Constant's Interface

- Create property files (for this example, the base file is called ExampleAppConstants.properties):

```
appTitle   = File Management Tool
editLabel  = Edit File
```

- Create an interface called ExampleAppConstants:

```
public interface ExampleAppConstants extends Constants {
    String appTitle();
    String editLabel();
}
```

- Get the contents at runtime like this:

Internationalization: Constant's Interface

- Create property files (for this example, the base file is called ExampleAppConstants.properties):

```
appTitle    = File Management Tool
editLabel   = Edit File
```

- Create an interface called ExampleAppConstants:

```
public interface ExampleAppConstants extends Constants {
    String appTitle();
    String editLabel();
}
```

- Get the contents at runtime like this:

```
ExampleAppConstants
constants=(ExampleAppConstants)GWT.create(ExampleAppConstants.class);
Button editButton = new Button(constants.editLabel());
```

Internationalization: Constant's Interface

- Advantages

Internationalization: Constant's Interface

- Advantages
  - Compilation performs code optimization, inlining

Internationalization: Constant's Interface

- Advantages
  - Compilation performs code optimization, inlining
  - Type-safe

Internationalization: Constant's Interface

- Advantages
  - Compilation performs code optimization, inlining
  - Type-safe
  - Strongly bound based on name

Internationalization: Constant's Interface

- Advantages
  - Compilation performs code optimization, inlining
  - Type-safe
  - Strongly bound based on name
  - Type or name errors caught at compile time (not run time!)

Internationalization: Constant's Interface

- Advantages
  - Compilation performs code optimization, inlining
  - Type-safe
  - Strongly bound based on name
  - Type or name errors caught at compile time (not run time!)
- Disadvantages

Internationalization: Constant's Interface

- Advantages
  - Compilation performs code optimization, inlining
  - Type-safe
  - Strongly bound based on name
  - Type or name errors caught at compile time (not run time!)
- Disadvantages
  - Limited to one value, cannot change over time

Internationalization: Constant's Interface

- Advantages
  - Compilation performs code optimization, inlining
  - Type-safe
  - Strongly bound based on name
  - Type or name errors caught at compile time (not run time!)
- Disadvantages
  - Limited to one value, cannot change over time
  - Use the `Messages` interface for these

Internationalization: Constant's Interface

- Constant Types

Internationalization: Constant's Interface

- Constant Types
  - String - editLabel = Edit

Internationalization: Constant's Interface

- Constant Types
  - String - editLabel = Edit
  - String[] - buttonLabels = New, Edit, Save

Internationalization: Constant's Interface

- Constant Types
  - String - editLabel = Edit
  - String[] - buttonLabels = New, Edit, Save
  - int - timeout = 30

Internationalization: Constant's Interface

- Constant Types
  - String - editLabel = Edit
  - String[] - buttonLabels = New, Edit, Save
  - int - timeout = 30
  - float, double - price = 65.00

Internationalization: Constant's Interface

- Constant Types
  - String - editLabel = Edit
  - String[] - buttonLabels = New, Edit, Save
  - int - timeout = 30
  - float, double - price = 65.00
  - boolean - debugMode = false

Internationalization: Constant's Interface

- Constant Types
  - String - editLabel = Edit
  - String[] - buttonLabels = New, Edit, Save
  - int - timeout = 30
  - float, double - price = 65.00
  - boolean - debugMode = false
  - Map - map of other properties - newLabel=New
    editLabel=Edit
    saveLabel=Save
    buttonLabels=newLabel, editLabel, saveLabel

Internationalization: Messages' Interface

Internationalization: Messages' Interface

- Create property files with parameters:

Internationalization: Messages' Interface

- Create property files with parameters:

```
appTitle   = File Management Tool, version {0}.{1}
editLabel  = Edit File
```

Internationalization: Messages' Interface

- Create property files with parameters:

```
appTitle    = File Management Tool, version {0}.{1}
editLabel   = Edit File
```

- Create an interface called ExampleAppMessages:

Internationalization: Messages' Interface

- Create property files with parameters:

```
appTitle   = File Management Tool, version {0}.{1}
editLabel  = Edit File
```

- Create an interface called ExampleAppMessages:

```
public interface ExampleAppMessages extends Messages {
    String appTitle(int majorVersion, int minorVersion);
    String editLabel();
}
```

Internationalization: Messages' Interface

- Create property files with parameters:

```
appTitle   = File Management Tool, version {0}.{1}
editLabel  = Edit File
```

- Create an interface called ExampleAppMessages:

```
public interface ExampleAppMessages extends Messages {
    String appTitle(int majorVersion, int minorVersion);
    String editLabel();
}
```

- Get the contents at runtime like this:

Internationalization: Messages' Interface

- Create property files with parameters:

```
appTitle   = File Management Tool, version {0}.{1}
editLabel  = Edit File
```

- Create an interface called ExampleAppMessages:

```
public interface ExampleAppMessages extends Messages {
    String appTitle(int majorVersion, int minorVersion);
    String editLabel();
}
```

- Get the contents at runtime like this:

```
ExampleAppMessages messages=(ExampleAppMessages)GWT.create(
ExampleAppMessages.class);
Label appTitle = new Label(messages.appTitle(10,0));
```

Internationalization: Messages' Interface

- Plural forms

Internationalization: Messages' Interface

- Plural forms
  - Different messages based on plurality

Internationalization: Messages' Interface

- Plural forms
- Different messages based on plurality

```
public interface ExampleAppMessages extends Messages{
    @DefaultMessage("You have {0} messages")
    @PluralText({"one","You have 1 message"})
    String catalogCount(@PluralCount int count);
}
```

Internationalization: Messages' Interface

- Plural forms
- Different messages based on plurality

```
public interface ExampleAppMessages extends Messages{
    @DefaultMessage("You have {0} messages")
    @PluralText({"one","You have 1 message"})
    String catalogCount(@PluralCount int count);
}


ExampleAppMessages messages = GWT.create(ExampleAppMessages.class);
Window.alert(catalogCount(catalogs.size());
```

Internationalization: Messages' Interface

- Plural forms
  - Different messages based on plurality

```
public interface ExampleAppMessages extends Messages{
    @DefaultMessage("You have {0} messages")
    @PluralText({"one","You have 1 message"})
    String catalogCount(@PluralCount int count);
}


ExampleAppMessages messages = GWT.create(ExampleAppMessages.class);
Window.alert(catalogCount(catalogs.size());
```

Internationalization: Messages' Interface

- Advantages

Internationalization: Messages' Interface

- Advantages
- Parameter substitution

Internationalization: Messages' Interface

- Advantages
  - Parameter substitution
  - Number of args enforced at compile time

Internationalization: Messages' Interface

- Advantages
  - Parameter substitution
  - Number of args enforced at compile time
- Disadvantages

Internationalization: Messages' Interface

- Advantages
  - Parameter substitution
  - Number of args enforced at compile time
- Disadvantages
  - Doesn't have the same range of types, only ints and Strings

Internationalization: Messages' Interface

- Advantages
  - Parameter substitution
  - Number of args enforced at compile time
- Disadvantages
  - Doesn't have the same range of types, only ints and Strings
- In both Constants and Messages, GWT can automatically generate the interface based on the properties file (i18nCreator):

Internationalization: Messages' Interface

- Advantages
  - Parameter substitution
  - Number of args enforced at compile time
- Disadvantages
  - Doesn't have the same range of types, only ints and Strings
- In both Constants and Messages, GWT can automatically generate the interface based on the properties file (i18nCreator):

```
i18nCreator ExampleAppApp -createMessages com.abc.ExampleApp.ExampleAppMessages
```

GWT and REST

GWT and REST

- Can GWT work with RESTful services?

GWT and REST

- Can GWT work with RESTful services?
  - Yes, if you use 3rd party components

GWT and REST

- Can GWT work with RESTful services?
  - Yes, if you use 3rd party components
    - Restlet (http://wiki.restlet.org/docs1.1/g1/13-restlet/28-restlet/144-restlet.html)

GWT and REST

- Can GWT work with RESTful services?
  - Yes, if you use 3rd party components
    - Restlet (http://wiki.restlet.org/docs1.1/g1/13-restlet/28-restlet/144-restlet.html)
    - gwt-rest (http://code.google.com/p/gwt-rest)

GWT and Seam

GWT and Seam

- Can GWT work with Seam?

GWT and Seam

- Can GWT work with Seam?
  - Yes. Seam can act as the back-end for the GWT service

GWT and Seam

- Can GWT work with Seam?
  - Yes. Seam can act as the back-end for the GWT service
    - Create the synchronous and asynchronous service interfaces as normal (we'll use "askIt" as an example)

GWT and Seam

- Can GWT work with Seam?
  - Yes. Seam can act as the back-end for the GWT service
    - Create the synchronous and asynchronous service interfaces as normal (we'll use "askIt" as an example)
    - Create a Seam component that implements the synchronous interface

GWT and Seam

```
@Name("org.jboss.seam.example.remoting.gwt.client.MyService")
public class ServiceImpl implements MyService {


   @WebRemote
   public String askIt(String question) {
      if (!validate(question)) {
         throw new IllegalStateException("Hey, this shouldn't happen, I checked"
         + " the client, but its always good to double check.");
      }
      return "42. Its the real question that you seek now.";
   }

   public boolean validate(String q) {
      ValidationUtility util = new ValidationUtility();
      return util.isValid(q);
   }
}
```

GWT and Seam

```
@Name("org.jboss.seam.example.remoting.gwt.client.MyService")
public class ServiceImpl implements MyService {


   @WebRemote
   public String askIt(String question) {
      if (!validate(question)) {
         throw new IllegalStateException("Hey, this shouldn't happen, I checked"
         + " the client, but its always good to double check.");
      }
      return "42. Its the real question that you seek now.";
   }

   public boolean validate(String q) {
      ValidationUtility util = new ValidationUtility();
      return util.isValid(q);
   }
}
```

GWT and Seam

```
@Name("org.jboss.seam.example.remoting.gwt.client.MyService")
public class ServiceImpl implements MyService {

   @WebRemote ← Needed for all web-remotable methods
   public String askIt(String question) {
      if (!validate(question)) {
         throw new IllegalStateException("Hey, this shouldn't happen, I checked"
         + " the client, but its always good to double check.");
      }
      return "42. Its the real question that you seek now.";
   }

   public boolean validate(String q) {
      ValidationUtility util = new ValidationUtility();
      return util.isValid(q);
   }
}
```

GWT and Seam

- Hooking up a GWT widget to the Seam component

GWT and Seam

- Hooking up a GWT widget to the Seam component
  - In the method which returns the asynchronous interface, obtain a reference to the asynchronous client stub:

GWT and Seam

- Hooking up a GWT widget to the Seam component
  - In the method which returns the asynchronous interface, obtain a reference to the asynchronous client stub:

```
private MyServiceAsync getService() {
   String endpointURL = GWT.getModuleBaseURL() + "seam/resource/gwt";
   MyServiceAsync svc = (MyServiceAsync) GWT.create(MyService.class);
   ((ServiceDefTarget) svc).setServiceEntryPoint(endpointURL);
   return svc;
}
```

GWT and Seam

- Hooking up a GWT widget to the Seam component
  - In the method which returns the asynchronous interface, obtain a reference to the asynchronous client stub:

```
private MyServiceAsync getService() {
   String endpointURL = GWT.getModuleBaseURL() + "seam/resource/gwt";
   MyServiceAsync svc = (MyServiceAsync) GWT.create(MyService.class);
   ((ServiceDefTarget) svc).setServiceEntryPoint(endpointURL);
   return svc;
}
```

GWT and Seam

- Hooking up a GWT widget to the Seam component
  - In the method which returns the asynchronous interface, obtain a reference to the asynchronous client stub:

```
private MyServiceAsync getService() {
   String endpointURL = GWT.getModuleBaseURL() + "seam/resource/gwt";
   MyServiceAsync svc = (MyServiceAsync) GWT.create(MyService.class);
   ((ServiceDefTarget) svc).setServiceEntryPoint(endpointURL);
   return svc;
}
```

GWT and Seam

- Hooking up a GWT widget to the Seam component
  - In the method which returns the asynchronous interface, obtain a reference to the asynchronous client stub:

```
private MyServiceAsync getService() {
   String endpointURL = GWT.getModuleBaseURL() + "seam/resource/gwt";
   MyServiceAsync svc = (MyServiceAsync) GWT.create(MyService.class);
   ((ServiceDefTarget) svc).setServiceEntryPoint(endpointURL);
   return svc;
}
```

That's it! GWT is now talking to Seam on the back-end

GWT and Seam

- More information available at: http://docs.jboss.org/seam/latest/reference/en-US/html/gwt.html

GWT and Login Security

GWT and Login Security

- Username entered in TextBox, Password entered in PasswordBox

GWT and Login Security

- Username entered in TextBox, Password entered in PasswordBox
- Username and hashed password sent to server

GWT and Login Security

- Username entered in TextBox, Password entered in PasswordBox
- Username and hashed password sent to server
- A session ID is generated and sent back to the client

GWT and Login Security

- Username entered in TextBox, Password entered in PasswordBox
- Username and hashed password sent to server
- A session ID is generated and sent back to the client

```
String sessionID = /*(Get sessionID from server's response to your login request.)*/;
final long DURATION = 1000 * 60 * 60 * 24 * 14; //duration remembering login. 2 weeks.
Date expires = new Date(System.currentTimeMillis() + DURATION);
Cookies.setCookie("sid", sessionID, expires, null, "/", false);
```

GWT and Login Security

- Username entered in TextBox, Password entered in PasswordBox
- Username and hashed password sent to server
- A session ID is generated and sent back to the client

```
String sessionID = /*(Get sessionID from server's response to your login request.)*/;
final long DURATION = 1000 * 60 * 60 * 24 * 14; //duration remembering login. 2 weeks.
Date expires = new Date(System.currentTimeMillis() + DURATION);
Cookies.setCookie("sid", sessionID, expires, null, "/", false);
```

- In the EntryPoint code,

GWT and Login Security

- Username entered in TextBox, Password entered in PasswordBox
- Username and hashed password sent to server
- A session ID is generated and sent back to the client

```
String sessionID = /*(Get sessionID from server's response to your login request.)*/;
final long DURATION = 1000 * 60 * 60 * 24 * 14; //duration remembering login. 2 weeks.
Date expires = new Date(System.currentTimeMillis() + DURATION);
Cookies.setCookie("sid", sessionID, expires, null, "/", false);
```

- In the EntryPoint code,

```
String sessionID = Cookies.getCookie("sid");
if ( sessionID != null ) {
    checkWithServerIfSessionIdIsStillLegal();
}else {
    displayLoginBox();
}
```

GWT and Login Security

- Never rely on the sessionID sent to your server in the cookie header

GWT and Login Security

- Never rely on the sessionID sent to your server in the cookie header
  - Look only at the sessionID that your GWT app sends explicitly in the payload of messages to your server.

GWT and Login Security

- Never rely on the sessionID sent to your server in the cookie header
  - Look only at the sessionID that your GWT app sends explicitly in the payload of messages to your server.
- Full article at: http://code.google.com/p/google-web-toolkit-incubator/wiki/LoginSecurityFAQ

GWT and Login Security

- Never rely on the sessionID sent to your server in the cookie header
  - Look only at the sessionID that your GWT app sends explicitly in the payload of messages to your server.
- Full article at: http://code.google.com/p/google-web-toolkit-incubator/wiki/LoginSecurityFAQ
- LDAP authentication dealt with in "GWT in Practice"

GWT Portlets

GWT Portlets

- Seems to be doable, but no documentation on Google's site

GWT Portlets

- Seems to be doable, but no documentation on Google's site
- Useful blog at http://xantorohara.blogspot.com/2007/07/portlets-and-gwt.html

What's new in GWT 1.5?

What's new in GWT 1.5?

- Java 5 language support

What's new in GWT 1.5?

- Java 5 language support
  - Generics

What's new in GWT 1.5?

- Java 5 language support
  - Generics
  - Enumerated types

What's new in GWT 1.5?

- Java 5 language support
  - Generics
  - Enumerated types
  - Annotations

What's new in GWT 1.5?

- Java 5 language support
  - Generics
  - Enumerated types
  - Annotations
  - Enhanced for-loop syntax

What's new in GWT 1.5?

- Java 5 language support
  - Generics
  - Enumerated types
  - Annotations
  - Enhanced for-loop syntax
  - Static imports

What's new in GWT 1.5?

- JavaScriptObject subclassing

What's new in GWT 1.5?

- JavaScriptObject subclassing
  - Lets you model JS objects as proper Java types without any memory or speed overhead

What's new in GWT 1.5?

- JavaScriptObject subclassing
  - Lets you model JS objects as proper Java types without any memory or speed overhead
  - Allows code completion, refactoring, inlining, etc.

What's new in GWT 1.5?

- JavaScriptObject subclassing
  - Lets you model JS objects as proper Java types without any memory or speed overhead
  - Allows code completion, refactoring, inlining, etc.
- Method inlining

What's new in GWT 1.5?

- JavaScriptObject subclassing
  - Lets you model JS objects as proper Java types without any memory or speed overhead
  - Allows code completion, refactoring, inlining, etc.
- Method inlining
  - Compiler selectively inlines bodies of Java and JSNI methods.

What's new in GWT 1.5?

- JavaScriptObject subclassing
  - Lets you model JS objects as proper Java types without any memory or speed overhead
  - Allows code completion, refactoring, inlining, etc.
- Method inlining
  - Compiler selectively inlines bodies of Java and JSNI methods.
    - Reduces code size, always improves performance

What's new in GWT 1.5?

- Assertion support

What's new in GWT 1.5?

- Assertion support
  - For debugging purposes, by default turned off in Production mode

What's new in GWT 1.5?

- Assertion support
  - For debugging purposes, by default turned off in Production mode
- Long emulation

What's new in GWT 1.5?

- Assertion support
  - For debugging purposes, by default turned off in Production mode
- Long emulation
  - JavaScript doesn't have 64-bit types

What's new in GWT 1.5?

- Assertion support
  - For debugging purposes, by default turned off in Production mode
- Long emulation
  - JavaScript doesn't have 64-bit types
  - GWT compiler generates smarter but slower code

What's new in GWT 1.5?

- Assertion support
  - For debugging purposes, by default turned off in Production mode
- Long emulation
  - JavaScript doesn't have 64-bit types
  - GWT compiler generates smarter but slower code
- A whole new slew of UI changes

GWT Pros/Cons

GWT Pros/Cons

- Pros

GWT Pros/Cons

- Pros
  - One language to work with - Java. Otherwise, one has to know HTML/DHTML, Javascript, JSP, JSTL, AJAX, etc.

GWT Pros/Cons

- Pros
  - One language to work with - Java. Otherwise, one has to know HTML/DHTML, Javascript, JSP, JSTL, AJAX, etc.
  - Debugging - IDE debugging in both client and server. No need for separate tools for different browsers (IE vs. Safari vs. Firefox)

GWT Pros/Cons

- Pros
  - One language to work with - Java. Otherwise, one has to know HTML/DHTML, Javascript, JSP, JSTL, AJAX, etc.
  - Debugging - IDE debugging in both client and server. No need for separate tools for different browsers (IE vs. Safari vs. Firefox)
  - Only POJOs - no JSON/XML/DOM stuff. Can leverage typical OO design patterns.

GWT Pros/Cons

- Pros

GWT Pros/Cons

- Pros
  - Support for many AJAX widgets

GWT Pros/Cons

- Pros
  - Support for many AJAX widgets
  - No browser specific code. GWT generates it.

GWT Pros/Cons

- Pros
  - Support for many AJAX widgets
  - No browser specific code. GWT generates it.
  - Code size is smaller and execution speed faster than hand-coding JavaScript.

GWT Pros/Cons

- Pros
  - Support for many AJAX widgets
  - No browser specific code. GWT generates it.
  - Code size is smaller and execution speed faster than hand-coding JavaScript.
  - Faster time to delivery and faster bug fixing than typical JSP/JavaScript apps.

GWT Pros/Cons

- Pros
  - Support for many AJAX widgets
  - No browser specific code. GWT generates it.
  - Code size is smaller and execution speed faster than hand-coding JavaScript.
  - Faster time to delivery and faster bug fixing than typical JSP/JavaScript apps.
  - Unit testing - client and server side unit tests like any other Java app

GWT Pros/Cons

- Pros
  - Support for many AJAX widgets
  - No browser specific code. GWT generates it.
  - Code size is smaller and execution speed faster than hand-coding JavaScript.
  - Faster time to delivery and faster bug fixing than typical JSP/JavaScript apps.
  - Unit testing - client and server side unit tests like any other Java app
  - Similar to Flex in server interaction

GWT Pros/Cons

- Pros

GWT Pros/Cons

- Pros
  - Instantiations makes GWT Designer

GWT Pros/Cons

- Pros
  - Instantiations makes GWT Designer
    - Plugin for Eclipse

GWT Pros/Cons

- Pros
  - Instantiations makes GWT Designer
    - Plugin for Eclipse
    - Drag-n-Drop UI design for GWT components

GWT Pros/Cons

- Pros
  - Instantiations makes GWT Designer
    - Plugin for Eclipse
    - Drag-n-Drop UI design for GWT components
    - CSS rules editor

GWT Pros/Cons

- Cons

GWT Pros/Cons

- Cons
  - Full app is loaded at page load

GWT Pros/Cons

- Cons
  - Full app is loaded at page load
    - But, only 1-2 K per page

GWT Pros/Cons

- Cons
  - Full app is loaded at page load
    - But, only 1-2 K per page
    - GWT has a tool to join all images into one for fast loading

GWT Pros/Cons

- Cons
  - Full app is loaded at page load
    - But, only 1-2 K per page
    - GWT has a tool to join all images into one for fast loading
  - GWT is only as bug-free and memory efficient as your least-efficient component (just like regular JavaScript)

GWT Pros/Cons

- Cons
  - Full app is loaded at page load
    - But, only 1-2 K per page
    - GWT has a tool to join all images into one for fast loading
  - GWT is only as bug-free and memory efficient as your least-efficient component (just like regular JavaScript)
  - GWT is a paradigm shift from “regular” web-apps - application vs. web app. Not all developers get it.

Tips for GWT application development

Tips for GWT application development

- Don’t mix layout in CSS and GWT

Tips for GWT application development

- Don’t mix layout in CSS and GWT
  - Do layout with GWT, use CSS for colors, font, etc.

Tips for GWT application development

- Don’t mix layout in CSS and GWT
  - Do layout with GWT, use CSS for colors, font, etc.
- Get to know the third-party GWT components

Tips for GWT application development

- Don’t mix layout in CSS and GWT
  - Do layout with GWT, use CSS for colors, font, etc.
- Get to know the third-party GWT components
  - More versatile than stock GWT components

Tips for GWT application development

- Don’t mix layout in CSS and GWT
  - Do layout with GWT, use CSS for colors, font, etc.
- Get to know the third-party GWT components
  - More versatile than stock GWT components
  - More layout options

Tips for GWT application development

- Don’t mix layout in CSS and GWT
  - Do layout with GWT, use CSS for colors, font, etc.
- Get to know the third-party GWT components
  - More versatile than stock GWT components
  - More layout options
- Figure out how you want to deal with “pages”

Tips for GWT application development

- Don't mix layout in CSS and GWT
  - Do layout with GWT, use CSS for colors, font, etc.
- Get to know the third-party GWT components
  - More versatile than stock GWT components
  - More layout options
- Figure out how you want to deal with "pages"
  - Cache? Refresh each time? Cache data?

Tips for GWT application development

- Don't mix layout in CSS and GWT
  - Do layout with GWT, use CSS for colors, font, etc.
- Get to know the third-party GWT components
  - More versatile than stock GWT components
  - More layout options
- Figure out how you want to deal with "pages"
  - Cache? Refresh each time? Cache data?
- Watch out for new GWT releases

Tips for GWT application development

- Don't mix layout in CSS and GWT
  - Do layout with GWT, use CSS for colors, font, etc.
- Get to know the third-party GWT components
  - More versatile than stock GWT components
  - More layout options
- Figure out how you want to deal with "pages"
  - Cache? Refresh each time? Cache data?
- Watch out for new GWT releases
- Use an IDE that supports GWT!

What's in the future for GWT?

What's in the future for GWT?

- In incubator status now

What's in the future for GWT?

- In incubator status now
  - More widgets - DatePicker, GlassPanel, FastTree, SliderBar/ProgressBar/Spinner

What's in the future for GWT?

- In incubator status now
  - More widgets - DatePicker, GlassPanel, FastTree, SliderBar/ProgressBar/Spinner
  - Bulk rendering for tables (10x faster), scrolling table, paging tables, editable tables

What's in the future for GWT?

- In incubator status now
  - More widgets - DatePicker, GlassPanel, FastTree, SliderBar/ProgressBar/Spinner
  - Bulk rendering for tables (10x faster), scrolling table, paging tables, editable tables
  - GWT logging suite - just like Java logging

What's in the future for GWT?

- In incubator status now
  - More widgets - DatePicker, GlassPanel, FastTree, SliderBar/ProgressBar/Spinner
  - Bulk rendering for tables (10x faster), scrolling table, paging tables, editable tables
  - GWT logging suite - just like Java logging
  - CssResource - enables CSS source files to be fully commented and modularized without a performance hit.

What's in the future for GWT?

- In incubator status now
  - More widgets - DatePicker, GlassPanel, FastTree, SliderBar/ProgressBar/Spinner
  - Bulk rendering for tables (10x faster), scrolling table, paging tables, editable tables
  - GWT logging suite - just like Java logging
  - CssResource - enables CSS source files to be fully commented and modularized without a performance hit.
  - GWTCanvas Widget - vector graphics and image manipulation

More GWT Information

More GWT Information

- Visit http://code.google.com/webtoolkit/

More GWT Information

- Visit http://code.google.com/webtoolkit/
- Blog at: http://googlewebtoolkit.blogspot.com/

More GWT Information

- Visit http://code.google.com/webtoolkit/
- Blog at: http://googlewebtoolkit.blogspot.com/
- Groups: http://groups.google.com/group/Google-Web-Toolkit

More GWT Information

- Visit http://code.google.com/webtoolkit/
- Blog at: http://googlewebtoolkit.blogspot.com/
- Groups: http://groups.google.com/group/Google-Web-Toolkit
- Full docs at: http://code.google.com/webtoolkit/overview.html

Demo of GWT

Demo of GWT

- “Skin” of existing app

Demo of GWT

- "Skin" of existing app
  - Facsimile of app's web UI

Demo of GWT

- "Skin" of existing app
  - Facsimile of app's web UI
  - Goals

Demo of GWT

- "Skin" of existing app
  - Facsimile of app's web UI
  - Goals
    - Determine ease of use of GWT

Demo of GWT

- "Skin" of existing app
  - Facsimile of app's web UI
  - Goals
    - Determine ease of use of GWT
    - Get an idea of page weight

Demo of GWT

- "Skin" of existing app
  - Facsimile of app's web UI
  - Goals
    - Determine ease of use of GWT
    - Get an idea of page weight
    - See what problems are lurking

Demo of GWT

- "Skin" of existing app
  - Facsimile of app's web UI
  - Goals
    - Determine ease of use of GWT
    - Get an idea of page weight
    - See what problems are lurking
- Findings

Demo of GWT

- "Skin" of existing app
  - Facsimile of app's web UI
  - Goals
    - Determine ease of use of GWT
    - Get an idea of page weight
    - See what problems are lurking
- Findings
  - Very easy to use (easier than JSPs)

Demo of GWT

- "Skin" of existing app
  - Facsimile of app's web UI
  - Goals
    - Determine ease of use of GWT
    - Get an idea of page weight
    - See what problems are lurking
- Findings
  - Very easy to use (easier than JSPs)
  - 3 pages, with interaction - 5K

Demo of GWT

- "Skin" of existing app
  - Facsimile of app's web UI
  - Goals
    - Determine ease of use of GWT
    - Get an idea of page weight
    - See what problems are lurking
- Findings
  - Very easy to use (easier than JSPs)
  - 3 pages, with interaction - 5K
  - Complex interaction with UI easy to code

Demo of GWT

- "Skin" of existing app
  - Facsimile of app's web UI
  - Goals
    - Determine ease of use of GWT
    - Get an idea of page weight
    - See what problems are lurking
- Findings
  - Very easy to use (easier than JSPs)
  - 3 pages, with interaction - 5K
  - Complex interaction with UI easy to code
  - I really liked it!