Google Web Toolkit (GWT)

St. Louis Java SIG
April 12, 2007

Brad Busch <brad.busch@gmail.com>

Andrew Prunicki <prunand@iit.edu>

What is GWT?

- GWT is a much different way to develop web applications from the past.
- GWT applications are:
  - Written in plain-old Java
  - Compiled into Javascript to be run on the browser
  - Developed using a standard JRE, the GWT workbench, and your IDE of choice

Why Java?

- Testing, debugging and profiling of browser code is easily accomplished using the same tools you are using today.
- Static type checking.
- Java-based OO designs are easier to communicate and understand.
- Java tools are very mature and feature rich.

Why GWT?

- Dynamic web development is painful with traditional tools.
- Too many disparate technologies and frameworks. HTML, XHTML, XML, *CSS*, *JavaScript*, **Java**, Java-EL, JSTL, AJAX, JSP, Taglibs, JSF, Struts, Tiles, Shale, Tapestry, RIFE, Seam, Spring MVC/Web Flow, Stripes, WebWork, Wicket
- Avoid browser incompatibilities.
- GWT 1.3 is now fully open source.

Key GWT Definitions

- Hosted Mode: Application runs as Java bytecode within JVM. (Debugging)
- Web Mode: Application runs as Javascript and HTML compiled using GWT's Java to Javascript compiler.
- Module: An XML configuration file with the extension .gwt.xml. Modules are used to specify entry point class(es), source and public path entries, other inherited modules and resource injection.

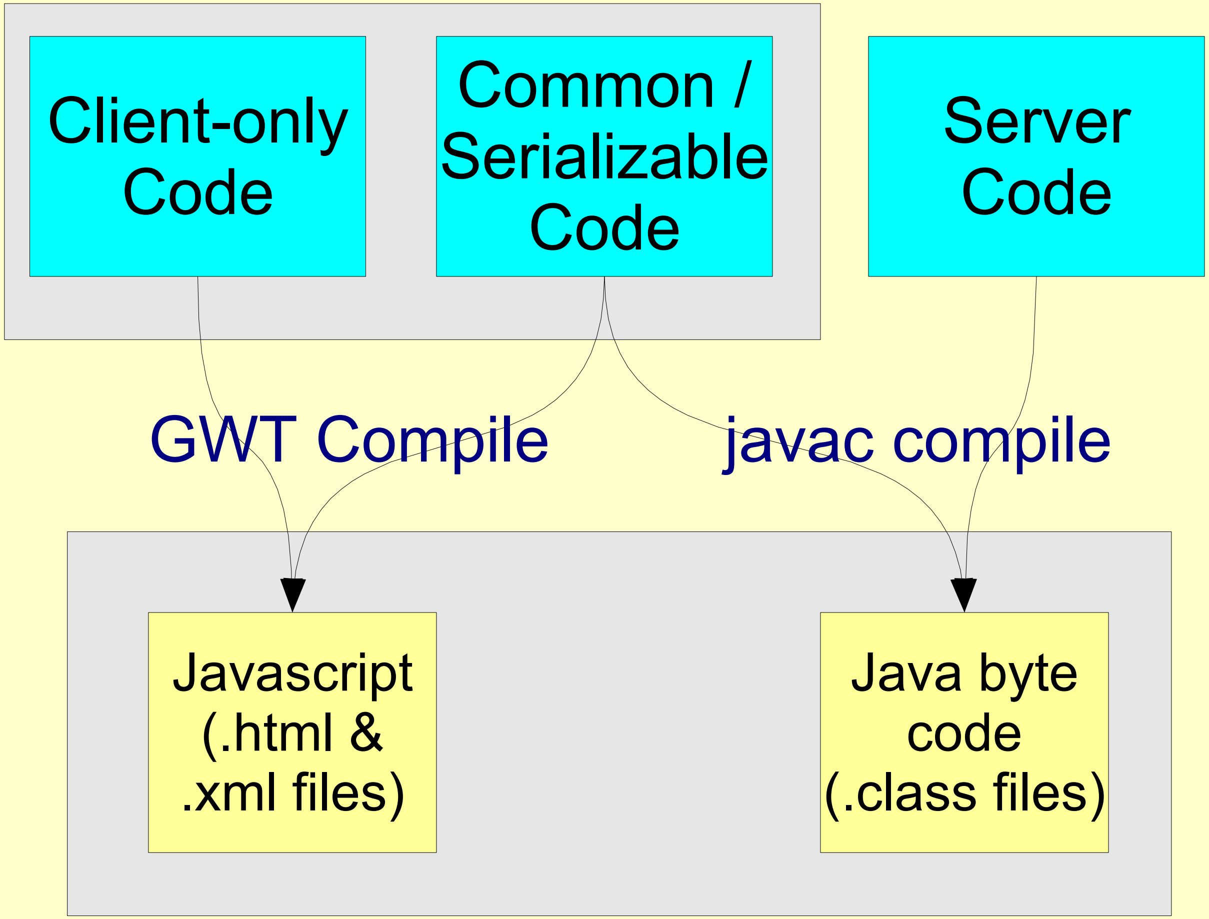
Quick Demo

Module Definition

- Look how easy it is to configure GWT:

```
<module>

  <!-- Inherit the core Web Toolkit stuff.                  -->
  <inherits name='com.google.gwt.user.User'/>

  <!-- Specify the app entry point class.                   -->
  <entry-point class='javasig.stl.demo.client.Chat'/>

  <!-- Define chat servlet -->
  <servlet path="/service/Chat" class="javasig.stl.demo.server.ChatServiceImpl"/>

</module>
```

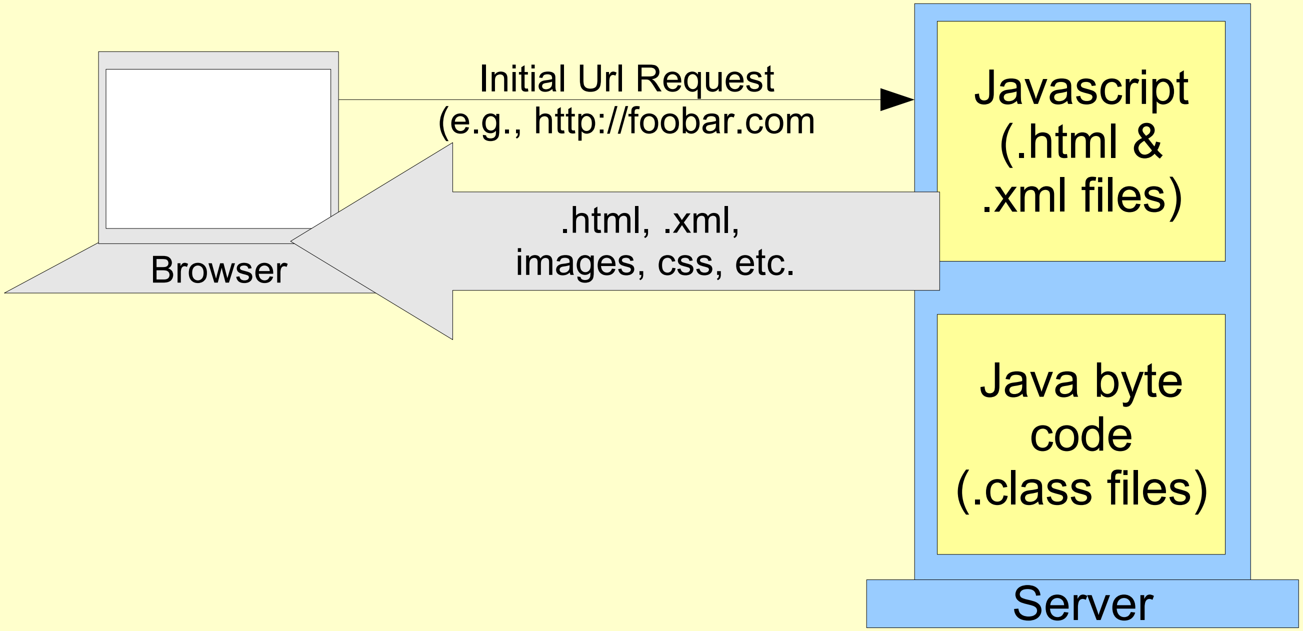
Compilation


Client-only Code
Common / Serializable Code
Server Code
GWT Compile
javac compile
Javascript (.html & .xml files)
Java byte code (.class files)

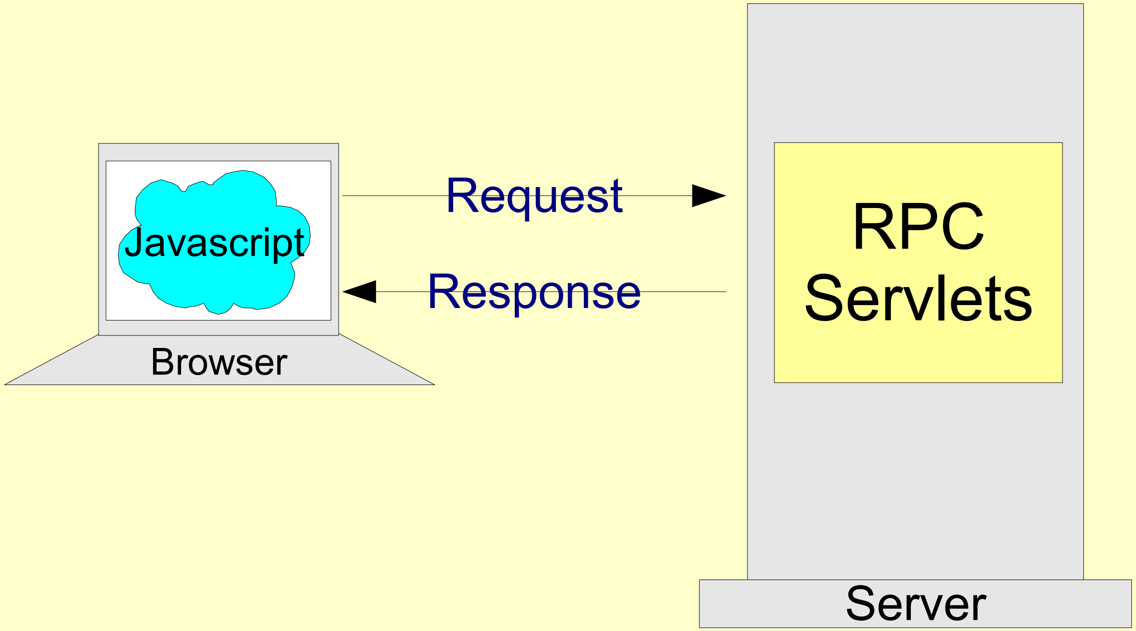
Compilation Options

- Different compilation options are supported:
  - Obfuscated – Can't read / Small file size
  - Pretty – Easy to read / 2x obfuscated file size
  - Detailed – Info overload / 4x obfuscated file size
- Compiled code is Obfuscated by default.

Runtime Architecture



RPC Architecture



Javascript

Browser

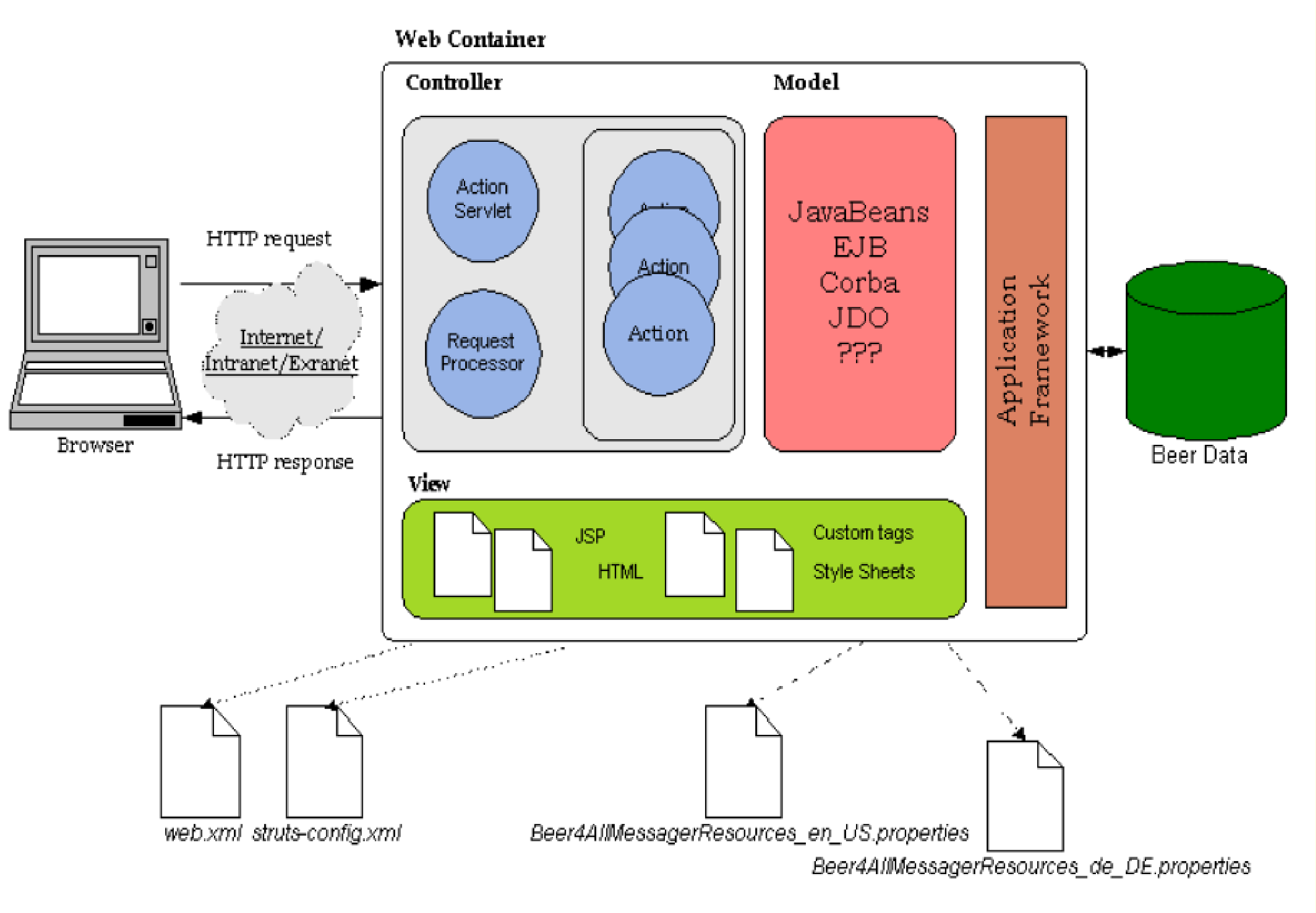
Request

Response

RPC Servlets

Server

Remember Struts?



Feel the Pain!

```
<!-- ========== Form Bean Definitions ============ -->
<form-beans>
     <form-bean name="login" type="test.struts.LoginForm" />
</form-beans>

<!-- ========== Action Mapping Definitions ======== -->
<action-mappings>
    <action
          path="/login"
          type="test.struts.LoginAction" >
      <forward name="valid" path="/jsp/MainMenu.jsp" />
      <forward name="invalid" path="/jsp/LoginView.jsp" />
    </action>
</action-mappings>
```

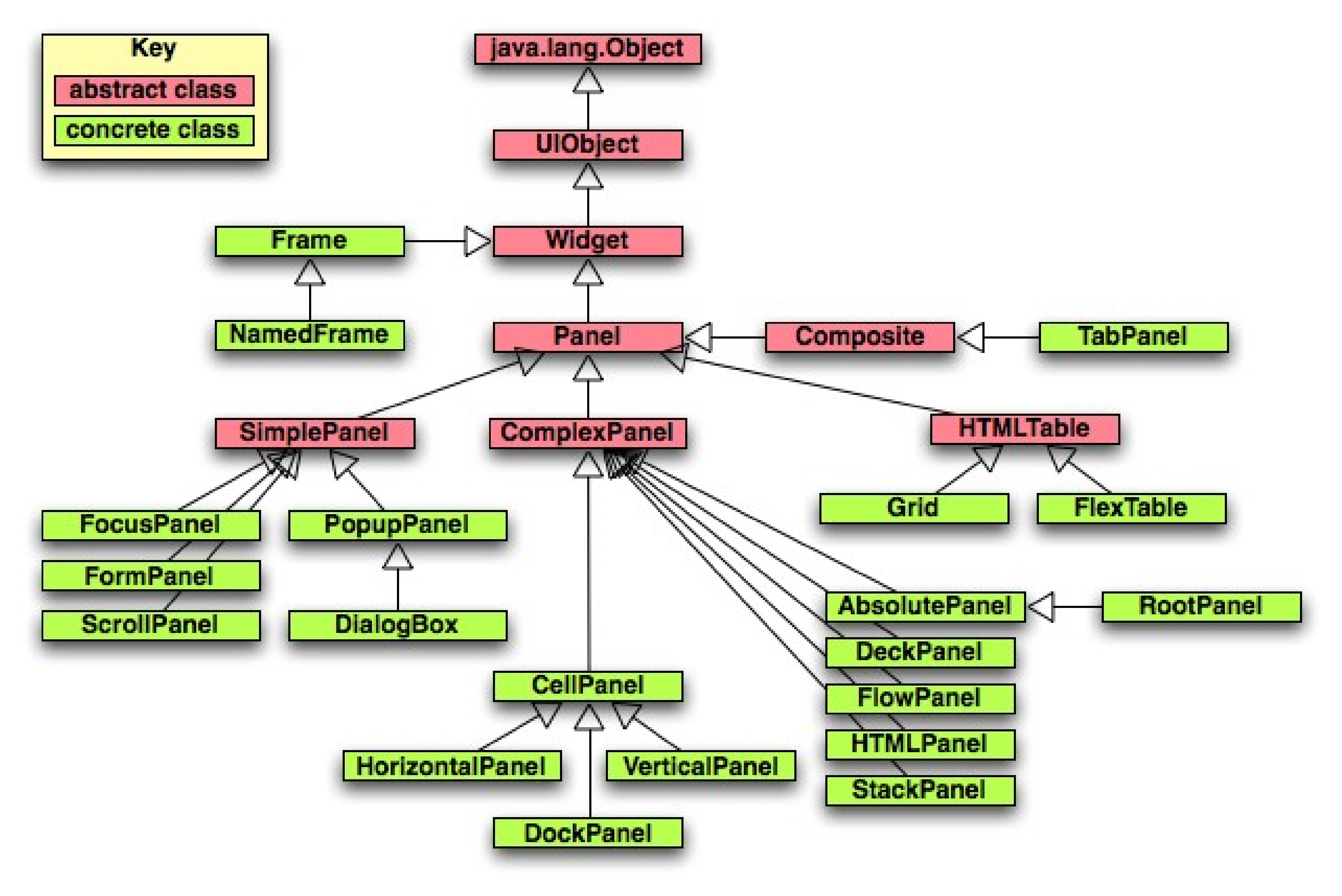
Client-side Development

- Since you are now developing Java code for the browser, you can now develop OO on the client. True code sharing and re-use!!
- GWT is similar to other windowing toolkits (e.g., Swing, SWT)
  - Event-driven
  - Component-based

Laying out a page

- GWT Pages are layed out with containers rather than layout managers.
- A rich set of containers is included. One container for example, FlexLayout, approximates GridBagLayout.
- Tools are becoming available to do WYSIWYG layout (e.g., GWT Designer) under commercial licenses.

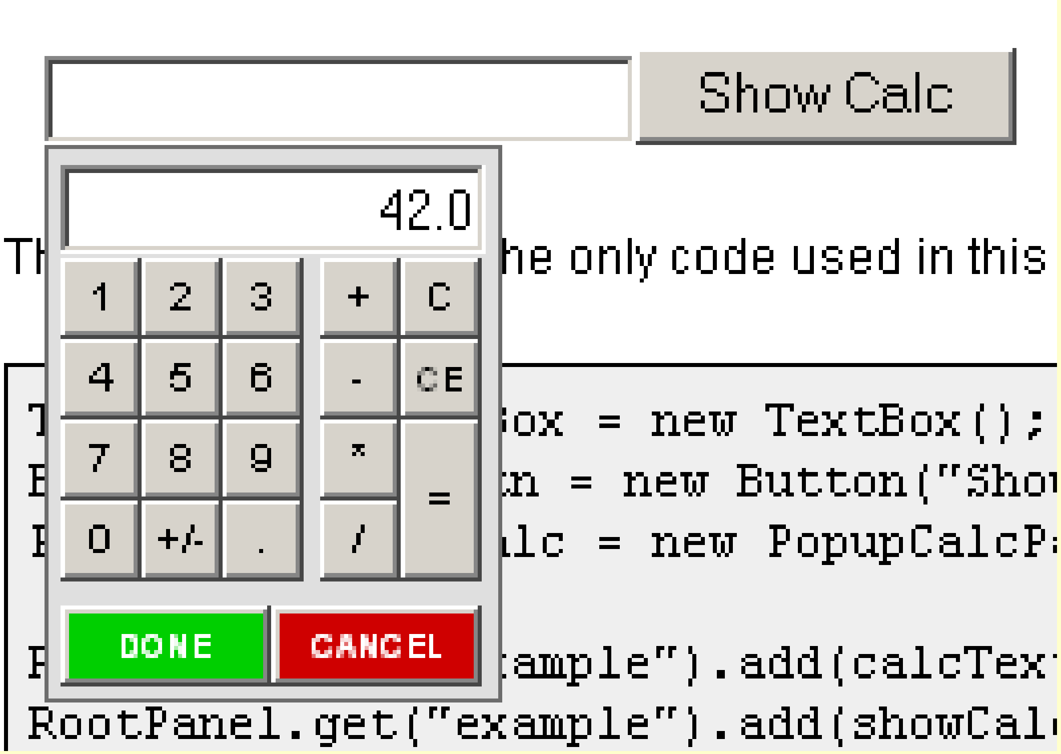
Container (Panel) Library



Widgets

- The basic set of widgets are included. For example:
  - Label
  - Button
  - Listbox
  - Textbox
  - Textarea
- Other (non-Google) widget libraries:
  - http://gwt-widget.sourceforge.net/ - the GWT Widget Library. A large collection of widgets including edittable labels, image buttons and Scriptaculous integration amongst others.
  - http://www.gwtwindowmanager.org/ GWT Window manager. Very nice looking windows.
  - http://gwtpowered.org - not a widget library, but contains a list of widgets, tools, and resources available for GWT

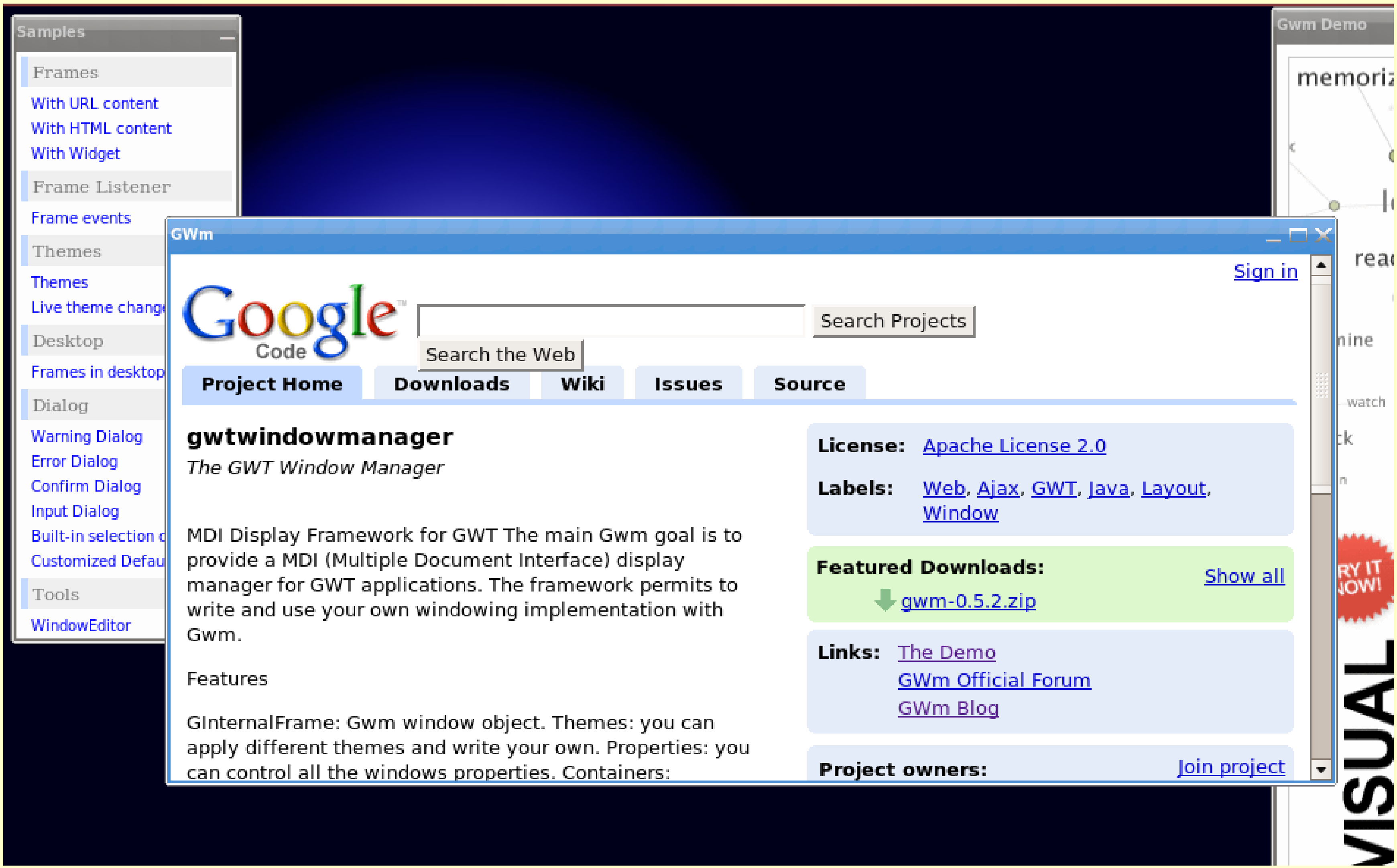
Sample 3rd Party Widgets



Th he only code used in this

```
T          ox = new TextBox();
B          n = new Button("Show
P          lc = new PopupCalcP

R          ample").add(calcTex
RootPanel.get("example").add(showCal
```

Sample 3rd Party Widgets



gwtwindowmanager

The GWT Window Manager

MDI Display Framework for GWT The main Gwm goal is to provide a MDI (Multiple Document Interface) display manager for GWT applications. The framework permits to write and use your own windowing implementation with Gwm.

Features

GInternalFrame: Gwm window object. Themes: you can apply different themes and write your own. Properties: you can control all the windows properties. Containers:

License: Apache License 2.0

Labels: Web, Ajax, GWT, Java, Layout, Window

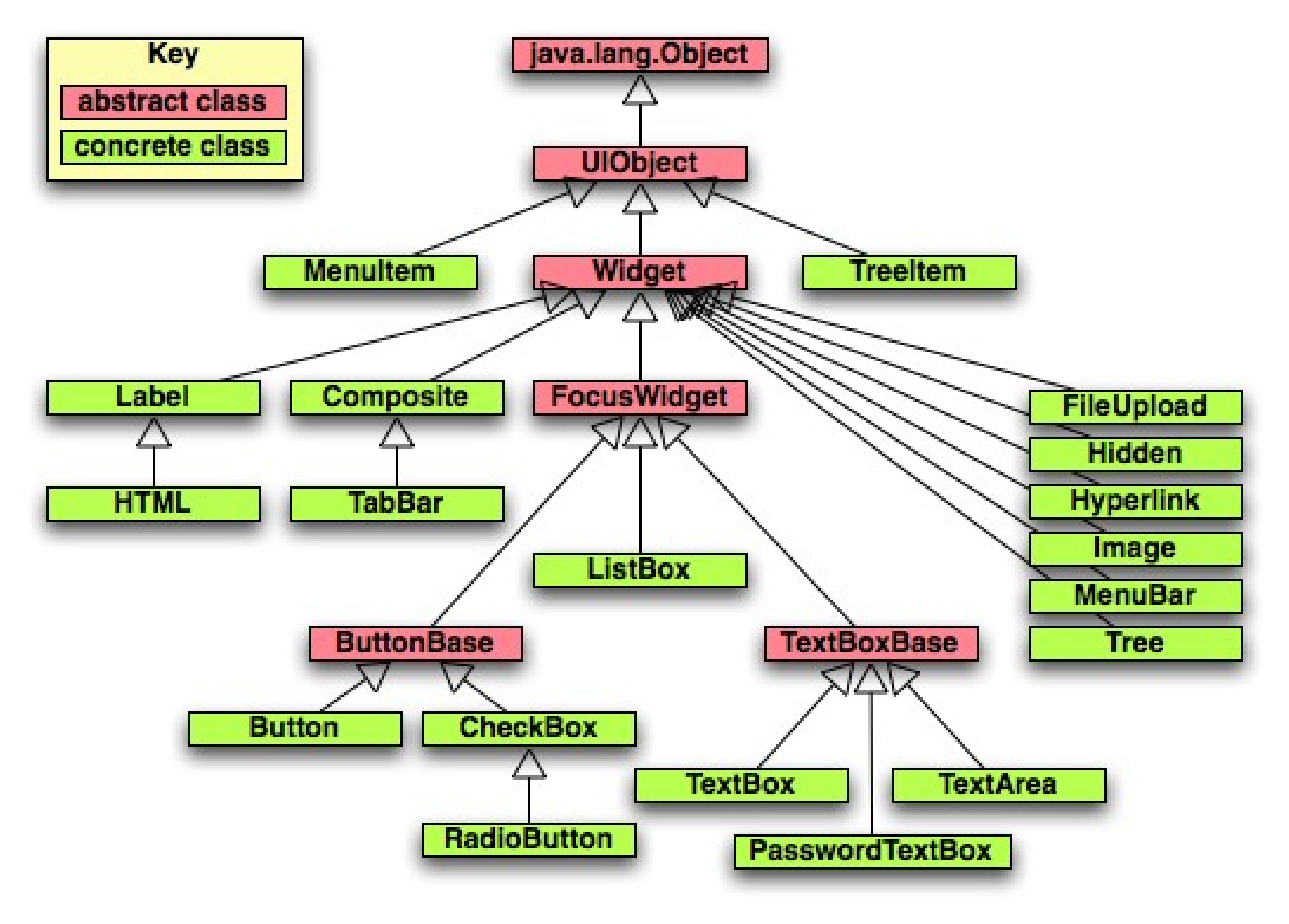
Featured Downloads: Show all

gwm-0.5.2.zip

Links: The Demo
GWm Official Forum
GWm Blog

Project owners: Join project

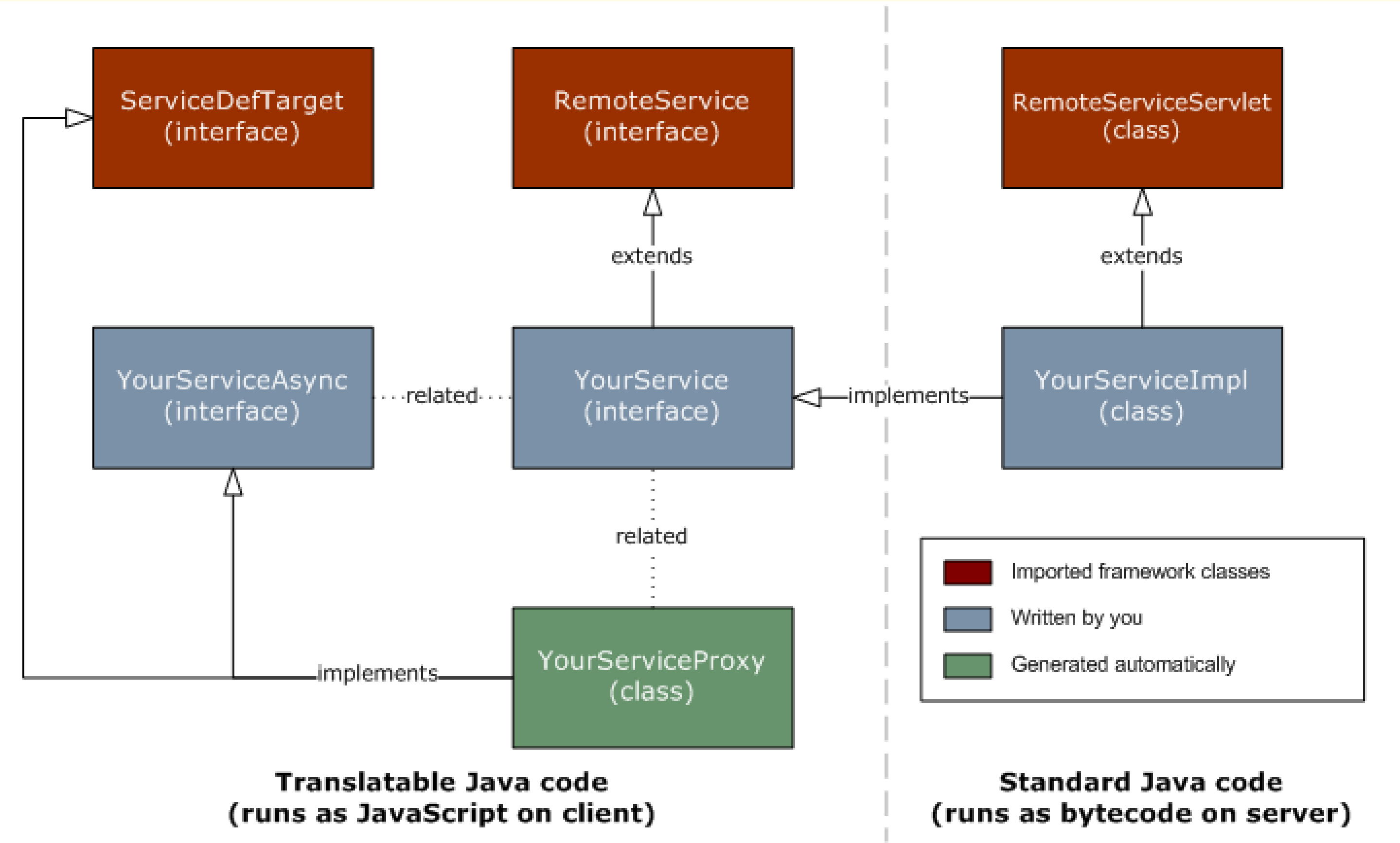
Standard Widget Library



Calling the server

- Client calls to the server are made using the GWT RPC mechanism.
- Can be confusing at times.
- Tools coming of age to hide the complexity (Googlipse, IntelliJ plugin)

RPC Services



Service Implementation

```
public interface ChatService extends RemoteService {

  public int sendMessage(User user, Friend friend, String message);

}



public class ChatServiceImpl extends RemoteServiceServlet implements ChatService {

  public int sendMessage(User user, Friend friend, String message);
    //Some code ...
  }

}
```

Service Call

```
public interface ChatServiceAsync {

  void sendMessage(User user, Friend friend, String message, AsyncCallback callback);
}


ChatServiceAsync serviceProxy = (ChatServiceAsync) GWT.create(ChatService.class);

ServiceDefTarget target = (ServiceDefTarget) serviceProxy;
target.setServiceEntryPoint(GWT.getModuleBaseURL()+”/service/Chat”);

AsyncCallback callback = new AsyncCallback() {

  public void onFailure(Throwable caught) {
    Window.alert("Problem sending message (" + caught.getMessage() + ")");
  }

  public void onSuccess(Object result) {
    //Do something fabulous...
  }
};

serviceProxy.sendMessage(_user, _friend, message, callback);
```

Module Definition

```
<module>

  <!-- Inherit the core Web Toolkit stuff.                  -->
  <inherits name='com.google.gwt.user.User'/>

  <!-- Specify the app entry point class.                   -->
  <entry-point class='javasig.stl.demo.client.Chat'/>

  <!-- Define chat servlet -->
  <servlet path="/service/Chat" class="javasig.stl.demo.server.ChatServiceImpl"/>

</module>
```

Demo RPC Debugging

Integration

- GWT integrates well with other frameworks.
- Client-side Integration:
  - Largely at the layer of embedded pages via IFrames, although custom Javascript integration is possible via JSNI.
- Server-side Integration
  - Anything is possible really, as RPC classes are just servlets with some method dispatching from GWT framework.

JSNI

- JSNI methods is a powerful technique, but:
  - less portable across browsers
  - more likely to leak memory
  - less amenable to Java tools
  - hard for the compiler to optimize.

```
public static native void alert(String msg) /*–{

  $wnd.alert(msg);

}–*/;
```

Security

- Be careful not to put too much on the client side, as you are exposing some inner-workings of your application.
- User input must be validated on the server, just as with any framework.
- Due to the fine-grained nature of exposed services, there are more points of entry exposed to malicious use.

Security

- GWT == Javascript on the client. Consequently, Javascript vulnerabilities effect GWT.
- Cross-Site Scripting (XSS)
- Request Forging (XSRF)

Unit Testing

- Test cases extend GWTTestCase
- Two ways to create unit tests:
  - junitCreator shell script
  - through IDE
- Two ways to run unit tests:
  - gwt – generated unit test script
  - through IDE

Limitations

- Java 1.4 compatibility
- Only a subset of the base Java classes are supported. (May be less of an issue as Java goes open source).
- Server-side objects may not be re-useable on the client if they have references to any code that will not / should not be on the browser. --> Consider using DTO's.

Consequences of GWT Development

- Developer required skill-sets can be different depending on the role:
  - Greater usage of OO & patterns
  - Experience with event-driven programming (e.g., Swing, SWT, even VB)
  - Less HTML & Javascript.
- UI development may actually go faster
- Greater UI consistency

Stuff You might want to know

- GWT is completely open-source (Apache 2.0)
- GWT development tools are available for Windows, Linux and Mac OS X
- Compiled GWT code is cross-platform and supports the following browsers:
  - Firefox, IE 5/6/7, Safari, Opera
- GWT# for .Net is in development.

GWT Roadmap

- Things to come:
  - RPC simplification
  - Drag and Drop support
  - Java 5 language support
  - Vector graphics library support: canvas, SVG, VML
- See more at:
  - http://code.google.com/webtoolkit/makinggwtbetter.html

Resources

http://code.google.com/webtoolkit

http://code.google.com/webtoolkit/makinggwtbetter.html

http://groups.google.com/group/Google-Web-Toolkit/web/security-for-gwt-a

http://www.ajug.org/meetings/download/struts.pdf

http://www.ociweb.com/mark/programming/GWT.html