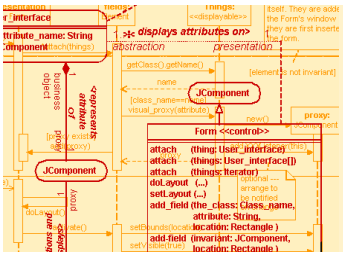


The Google Web Toolkit

Allen I. Holub
Holub Associates
www.holub.com
allen@holub.com

This Talk

- The point of this talk is to give you an overview of GWT suitable for evaluating it.
- This is not a how-to talk for existing GWT developers.
- Hard-core-programming details, examples, tutorials, etc. are available at:
  http://code.google.com/webtoolkit/

Background

Web 1.0

- Pull based. All real work is done on the server
  - Server creates and lays out page.
  - Server validates data
    - Must re-serve page if there's an error.
    - New page often omits previously-entered data
- All user input must be passed through sever.
  - Entire page is submitted, new page returned.
- All user interaction must be form based.
- Slow response times are inherent.
- Reasonably easy to secure.
- Inherently procedural.

Web 2.0

- Client/Server Architecture
  - Most of the U/I work is done client side in JavaScript
  - JavaScript programming is excruciatingly difficult
- Can be OO
  - Both client-side and server-side objects can exist, and can communicate with one another.
- Can be push/pull.
- Most data validation & collection done client side.
  - UI is much more responsive.
- Dynamic pages change with user input.
  - Not form based (though it can be).
  - Drag and drop, etc., is easy.
- More difficult to secure (larger attack surface)

In a nutshell

- A web 2.0 application is effectively:
  - A client-side program, written in JavaScript.
  - Uses the server primarily as a data repository
  - Communicates with the server frequently
    - Uses RPC or equivalent
    - All important user activity is typically reported to server immediately
    - Validation usually done as soon as field is entered, and error is reported (or fixed) immediately.
    - Many small, lightweight, server requests rather than large ones.

The Dark Underbelly

The most complicated part of the app,
the part of the app most visible to the user,
the part of the app on which you spend the most time,

IS WRITTEN IN JAVASCRIPT,

the world's worst programming language.

High-Level Languages

- Standard, portable.
- Focus on the problem, not on the hardware / OS.
- Language does high-level tasks for you (extensive libraries).
- Finds bugs.
  - E.g. Compiler vs. Run-time error
- Promotes good program organization through language structure.
  - E.g. Modules, classes
- Large, platform independent ecosystem
  - E.g. Eclipse

Assembly Languages

- Machine dependent, not portable.
- Forces you to worry about machine architecture. Focus on the language, not the work.
- Does nothing for you. Libraries are primitive.
- Does not help find bugs.
  - Language complexity increases bug counts.
  - No static typing & other bug-reduction features.
- Promotes bad program organization through language structure.
  - E.g. global variables, goto
- Hodge-podge of Micky-Mouse vendor-supplied development tools

Examples

- High-level Languages
  - C++
  - C#
  - **Java**
- Assembly Languages:
  - 80x86 assembler, etc.
  - C
  - **JavaScript**

Translate Java to JavaScript!

"If all you have is a hammer, everything looks like a nail." *- Bernard Baruch*

- Sometimes, though, a hammer is the right tool!

Ultimate Geeks "Multi-tool Hammer"

Java-to-JavaScript --- Why?

- Standard development tools work on both browser & server code (Eclipse)
- Static type checking, etc., moves bugs to compile time
- Good editor support
- Automated refactoring is possible
- Testing is easy using standard tools (junit, log4j)

System is OO front to back.

- You have a distributed application, with some server-side objects and some client-side objects, talking to each other over a remote-procedure-call mechanism.
- Solves most of the problems that (over) complex technologies like JSF and struts try (unsuccessfully) to solve.

Download From

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

GWT is …

- A Java-to-JavaScript compiler.
- Libraries
  - Widgets
  - JRE Emulations (java.lang, java.util)
- Jetty-based web-server functionality
  - runs under Eclipse, so you can set server-side breakpoints.
- Plugins for all major browsers
  - let you run your client-side code in the browser (Firebug works), but under Eclipse (so you can set client-side breakpoints, etc.).
- An Eclipse plugin for compiling, syntax help, etc.
- A few minor support tools that let you create Eclipse projects, etc.

Compatibility (output)

- Compiled JavaScript runs fine in:
  - IE 7.0+ (IE 6 support is marginal)
  - Firefox (all versions)
  - Safari (2.x 3.x …)
  - Chrome
  - Opera
- Native widgets used.
- CSS is browser dependent
  - HTML5/CSS3 works
- Server side is Java servlet
  - optional

Compatibility (input)

- Windows, Linux, Mac OS X
- Java 1.6
  - Including generics, for(x:y), etc.
- byte, char, short, int, long, float, double, String
  - Some methods of String class are missing.
  - JavaScript regular exception syntax is used.
  - No strictfp
- try/catch/finally supported
- No synchronization, reflection, finalization
- No serialization
  - but can send/return objects in RPC.

Library Support

- java.lang
  - Object, String, Exception
    - Some methods of String are missing.
- java.util
  - ArrayList, Arrays, Collections, Data, HashMap, HashSet, Stack, Vector, Iterator, & associated interfaces and Exceptions.
  - Fully<paramaterized>
- java.io.Serializable

Create a Project

Use create-GWT-project button in Eclipse, or manually:

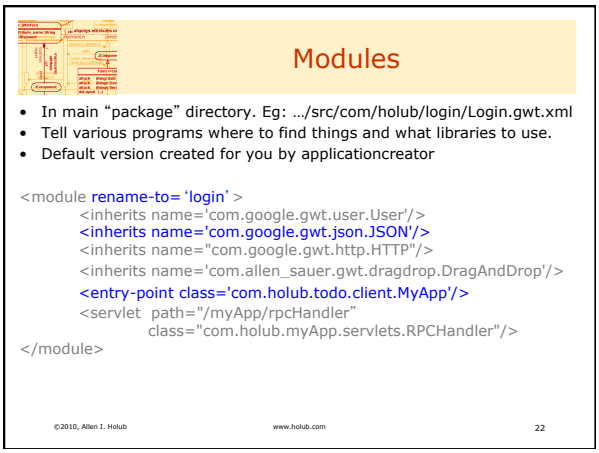
```
~/Foo> projectCreator -ant Foo -eclipse Foo          create Eclipse project
    Created directory src
    Created directory test
    Created file Foo.ant.xml
    Created file .project
    Created file .classpath

                                                     create skeleton application
 ~/Foo> applicationCreator -eclipse Foo com.example.foo.client.Foo
    Created directory src/com/example/foo/client
    Created directory src/com/example/foo/public
    Created file src/com/example/foo/Foo.gwt.xml
    Created file src/com/example/foo/public/Foo.html
    Created file src/com/example/foo/client/Foo.java
    Created file Foo.launch
    Created file Foo-shell                           launch in hosted mode
    Created file Foo-compile                         create final (compiled) ver.
```

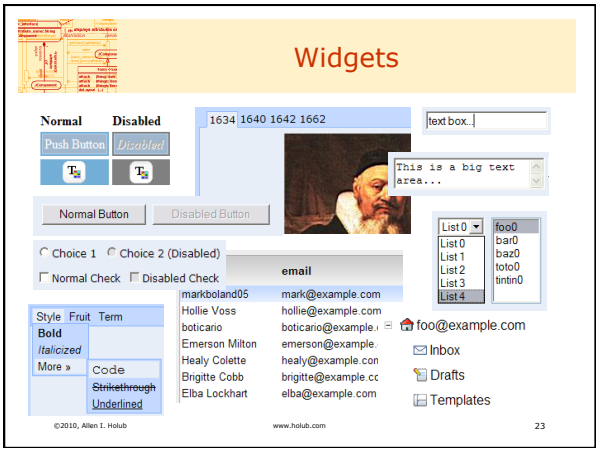

Uses Standard War Layout

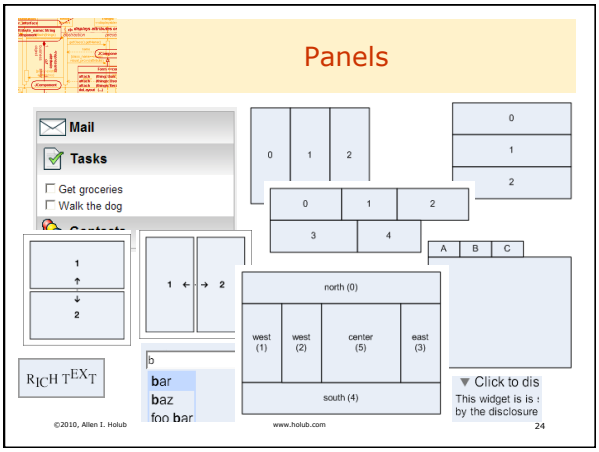
- /myproject
- /myproject/src (sources go here)
- /myproject/war (can use jar to distribute)
  - MyProject.html (root page for your app)
  - MyProject.css
- /myproject/war/WEB-INF
  - classes/ (compiler puts output here)
  - lib/ (you put .jar files that you need here)
  - web.xml (standard servlet config)

Project Structure

- Directory/package structure is proscribed
  - …/src
    - Files in this directory will be in (server-side) application class path.
  - …/src/com/holub/myApp (com.holub.myApp)
    - "Module" XML files have to go here
  - …/src/com/holub/myApp/client (com.holub.myApp.**client**)
    - Client-side classes (and shared client-sever classes such as constant definitions) go here or in subpackages.
  - …/src/com/holub/myApp/server (com.holub.myApp.**server**)
    - Server-side classes go here
    - May use classes in client-side packages
  - …/src/com/holub/myApp/**public**
    - static resources (.html, images, css, etc.)
    - These files end up in a subdirectory directory of the war file.







CSS

- All Widgets (and all the panels are Widgets) support: `w.setStyleName("css-style").`
- CSS is not abstracted,
  - is browser dependent.
- Include style sheet in the normal way:

```
<link rel="stylesheet" type="text/css"
                                href="todo.css" >
```

File is src/com/holub/myApp/public/todo.css

Creating a widget

```
TextBox b = new TextBox();
b.setStyleName("box-style"); // must be in .css
b.addChangeHandler
( new ChangeHandler()
  {  public void onChange(ChangeEvent event)
     {
          //...
     }
  }
);
```


FlexTable

```
FlexTable layout = new FlexTable();
layout.setWidget( 0, 0, new HTML("Your email"));
layout.setWidget( 1, 0, returnAddress );
layout.setWidget( 2, 0, new HTML("Subject:"));
layout.setWidget( 3, 0, subject );
layout.setWidget( 4, 0, new HTML("Message:"));
layout.setWidget( 5, 0, message );
```

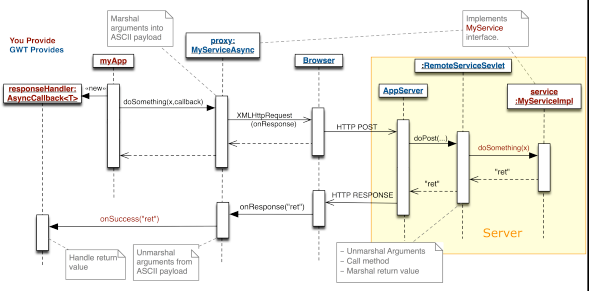
Installing into a Panel

```
Panel centerPanel = new HorizontalPanel();
      centerPanel.add( Toolbar.getInstance() );
      centerPanel.add( listArea );

Panel mainFrame = new VerticalPanel();
      mainFrame.add( new HTML("Hello") );
      mainFrame.add( centerPanel );

RootPanel.get("application").add(mainFrame);
```

In app.html:

```
<div id="application"></div>
```


RPC



RPC Implementation (server side)

```
public interface MyService
{    public String hello( String msg ); }


public class MyServiceImplemenation implements MyService
                              extends RemoteServiceServlet
{    public String hello( String msg )
     {      return "Received" + msg;
     }
}
```

RPC Implemenation (client side)

```
interface MyServiceAsync {
   public void hello( String msg, AsyncCallback
   callback ); }

class MyClientSideClass
{
  MyServiceAsynch server;
  static {
  String moduleRelativeURL = GWT.getModuleBaseURL() +
                                "/myapp/MyServlet" ;
  server = (MyServiceAsync) GWT.create(MyService.class);
  ((ServiceDefTarget) server).
            setServiceEntryPoint(moduleRelativeURL);
   }
   //. . .
```


RPC implementation (client side)

```
void f()
{  server.hello( "Foo",
       new AsychCallback()
       {   public void onFailure(final Throwable caught)
           {  Window.alert("call failed:" + caught );
           }
           public void onSuccess(Object result)
           {  String s = (String) result;
           }
       }
   );
}
```


RPC method arguments

- Primitives (char, byte, int, etc.)
- String, Date, Character, Byte, Integer, …
- Serializable or IsSerializable objects
- Arrays of Serializable or primitives
- Has at least on Serializable sublcass.
- Polymorphic returns & args are okay.
- Your *Serializable* objects:
  - All non-final, non-transient fields must be Serializable
  - Must have a public no-arg constructor.

Exceptions under RPC

- Service-interface methods may have `throws` clauses and throw defined exceptions.
- They are passed to the `AsynchCallback`'s `onFailure` method.
- `InvocationException` is passed if the network connection breaks.

JUnit Integration

```
public class FooTest extends GWTTestCase {

  /*
   * Specifies a module to use when running this test case. The returned
   * module must cause the source for this class to be included.
   *
   * @see com.google.gwt.junit.client.GWTTestCase#getModuleName()
   */
  public String getModuleName() {
    return "com.example.foo.Foo";
  }

  public void testStuff() {
    assertTrue(2 + 2 == 4);
  }
}
```

junitCreator builds test cases for you

JSNI

```
public static native void alert(String msg) /*-{
  $wnd.alert(msg);
}-*/;
```

- Acessing Methods:

```
[instance-expr.]@class-name::method-name
          (param-signature)(arguments)
```

- Accessing Fields:

```
[instance-expr.]@class-name::field-name
```

```
this.@com.google.gwt.examples.JSNIExample::
          instanceFoo(Ljava/lang/String;)(s);
```

JSNI

```
public static native void alert(String msg) /*-{
  $wnd.alert(msg);
}-*/;
```


Other JSNI Issues

- It's possible to pass Java objects (including null) into JavaScript and vice-versa.
- Exceptions thrown from JavaScript can be caught as a `JavaScriptException`

Drag And Drop

- Not supported by GWT, but easy to add
  - http://code.google.com/p/gwt-dnd/

Implementing DnD

```
private PickupDragController dragController =
     new PickupDragController(RootPanel.get(), false);

dragController.makeDraggable( someWidget );

dragController.registerDropController( dropController );
```


A Drop Controller

```
private class TrashDropController
                              extends SimpleDropController
{public TrashDropController() {super(TrashIcon.this); }
    public void onDrop(DragContext context)
    {super.onDrop(context);
    //...
}
public void onEnter(DragContext context) { }
public void onLeave(DragContext context) { }
public void onPreviewDrop(DragContext context) ... {}
}
```


Q&A

Allen Holub
www.holub.com

get slides from
http://www.holub.com/publications/notes_and_slides/