The Google Web Toolkit (GWT): The Basics

Originals of Slides and Source Code for Examples:
http://courses.coreservlets.com/Course-Materials/ajax.html

Topics in This Section

- **Pros and cons of GWT**
- **Installing GWT**
- **Development process**
  - Making a project
  - Editing auto-generated HTML file
  - Editing auto-generated application class
- **Testing process**
  - Hosted mode
  - Web mode
- **Client-side listeners**
- **Custom Java classes**

Overview and Installation

Overview of the Google Web Toolkit (GWT)

- **Big Idea**
  - You write *both* client-side and server-side code in Java
  - Client-side code
    - Uses an API similar to Swing
    - Most basic JDK 1.4 constructs and classes supported
    - Gets compiled into JavaScript that runs in your browser
  - Server-side code
    - Client uses a callback API and specifies data source URL
      - Once you define callback, you are mostly using regular Java method calls with complex arguments and return values
    - Server extends special class and defines regular methods
      - These are not servlet-style doXxx methods linked to HTTP
      - Arguments and return values can be
        - » Primitives, Strings, arrays, a few java.util collections, Serializable custom classes

Advantages of GWT

- **No JavaScript syntax errors**
  - Use a reliable, strongly-typed language (Java) for development and debugging
  - No JavaScript programming at all!
- **Can use complex Java on the client**
  - Turned into JavaScript, but you still use String, array, Math class, ArrayList, HashMap, custom classes, etc.
  - Full IDE-based Java support for development/debugging
- **Can send complex Java types from the server**
  - Data gets serialized across network
- **Standalone test environment**
  - Can test within Eclipse without installing a server
- **Support by major company**
  - From the company that popularized Ajax in the first place
  - Company won't go away like perhaps with AjaxTags

Disadvantages of GWT

- **Big learning curve**
  - Java developers can deploy with AjaxTags in just a few minutes, whereas it takes much longer to get anything running with GWT.
- **Cumbersome deployment**
  - Clumsy and poorly documented process to deploy on a regular Java-based Web server.
- **Nonstandard approach to integrate JavaScript**
  - You *never* put direct JavaScript in your HTML. Instead, you use JSNI to wrap JavaScript in Java.
    - Very powerful in the long run, but hard to get used to at first.
- **Only for Java developers**
  - Most Ajax environments do JavaScript on the client and have a choice for the server. GWT is based entirely around Java.
- **Unusual approach**
  - Fundamentally different strategy than all other Ajax environments makes evaluation and management buyoff harder

Sites that Use GWT

- **Google Sites**
  - Google Base, Google Checkout, Google Web Content Manager
- **dismoiou.com**
- **gpokr.com**
- **queplix.com**
- **etripbuilder.com**
- **omnispense.com**
- **tipit.to**

Gpokr - Free Texas Holdem Poker - Microsoft Internet Explorer
DONKEY_JR Table
gpokr
dismoiou.fr - Microsoft Internet Explorer
dismoiou
le monde en partage (bêta)
moi
lieux
membres
Paris

Installation

- **Downloading**
  - Binaries
    - http://code.google.com/webtoolkit/download.html
  - Source code
    - http://code.google.com/p/google-web-toolkit/source
- **Installation**
  - Download main file
  - Unzip into a directory of your choice
  - Optional: set your PATH (not CLASSPATH) to include this directory
    - Or, you can just run the scripts with an absolute path or from this directory

Documentation

- **Developer Guide**
  - http://code.google.com/webtoolkit/documentation/
- **Getting Started Guide**
  - http://code.google.com/webtoolkit/gettingstarted.html
- **Class Reference**
  - http://code.google.com/webtoolkit/documentation/gwt.html
- **Widget API**
  - http://code.google.com/webtoolkit/documentation/com.google.gwt.user.client.ui.html
    - This is the one you will use the most
- **Developer Forum**
  - http://groups.google.com/group/Google-Web-Toolkit
    - Pretty active with experts to answer questions

Application Development Steps (Eclipse Version)

- **Create an Eclipse application**
  - Use projectCreator and applicationCreator scripts
  - You can make non-Eclipse projects also, but the examples in this tutorial will assume Eclipse
- **Edit auto-generated HTML file**
  - Called MainNameApp.html
    - Under src/package.../public folder in Eclipse
  - Give id's to regions where controls will be placed
- **Edit auto-generated Java application class**
  - Class: package...client.MainNameApp
    - Under src folder in Eclipse
  - Method: onModuleLoad
  - Create controls and give them event handlers
  - Insert controls into HTML page
    - RootPanel.get("htmlID").add(controlReference);

Creating (Eclipse) Projects

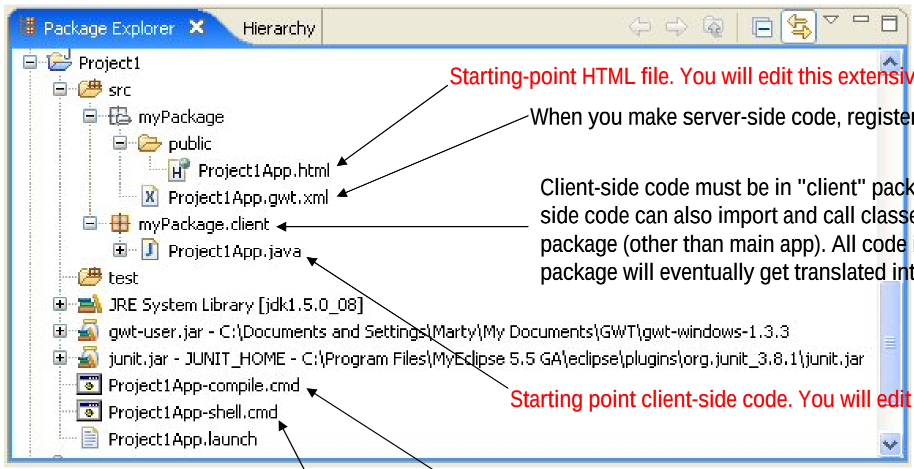
Setting Up a Project (Eclipse Version)

1. **Make sure PATH includes GWT installation directory**
   – Or specify full path for scripts
2. **Make a temporary directory for your project**
   – DOS> mkdir Project1
   – DOS> cd Project1
3. **Make a blank Eclipse project**
   – DOS> projectCreator –eclipse Project1
4. **Make a starting-point application**
   – DOS> applicationCreator –eclipse Project1 myPackage.client.Project1App
5. **Start Eclipse and import project**
   – File > Import > Existing Projects into Workspace
      - Browse to temporary directory from (2) above and click "Finish"
      - If you specify "copy files" (usual approach), then delete the temporary directory from (2) above

Names must match

Name of driver class that will be created. Rightmost package name must be "client".

Resulting Eclipse Project



```
Project1
  src
    myPackage
      public
        Project1App.html
      Project1App.gwt.xml
    myPackage.client
      Project1App.java
  test
  JRE System Library [jdk1.5.0_08]
  gwt-user.jar - C:\Documents and Settings\Marty\My Documents\GWT\gwt-windows-1.3.3
  junit.jar - JUNIT_HOME - C:\Program Files\MyEclipse 5.5 GA\eclipse\plugins\org.junit_3.8.1\junit.jar
  Project1App-compile.cmd
  Project1App-shell.cmd
  Project1App.launch
```

Starting-point HTML file. You will edit this extensively.

When you make server-side code, register servlet in here.

Client-side code must be in "client" package. Server-side code can also import and call classes in this package (other than main app). All code in this package will eventually get translated into JavaScript.

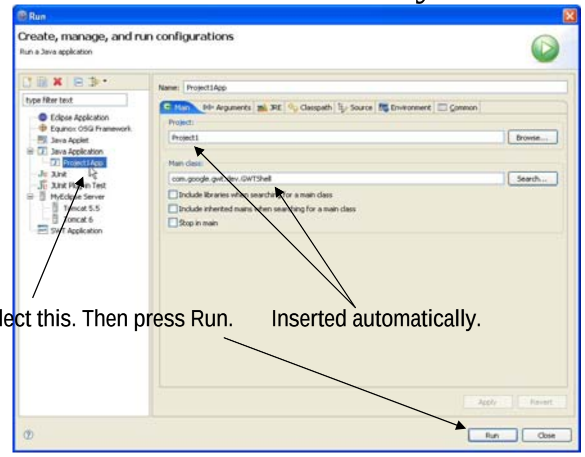
Starting point client-side code. You will edit this extensively.

After you have finished testing and debugging, you can double-click this to generate www directory containing HTML and JavaScript that can run in a regular browser. But an easier way to build this files is to run in hosted mode and then click Compile/Browse (see later slides).

Not needed if you run within an IDE.

Testing the Project in Hosted Mode

- **New projects have "HelloWorld" functionality built in**
  - Two buttons and two labels created automatically
- **Run the project in hosted mode**
  - In Eclipse: Run> Run> Java Application



Name: Project1App

Project: Project1

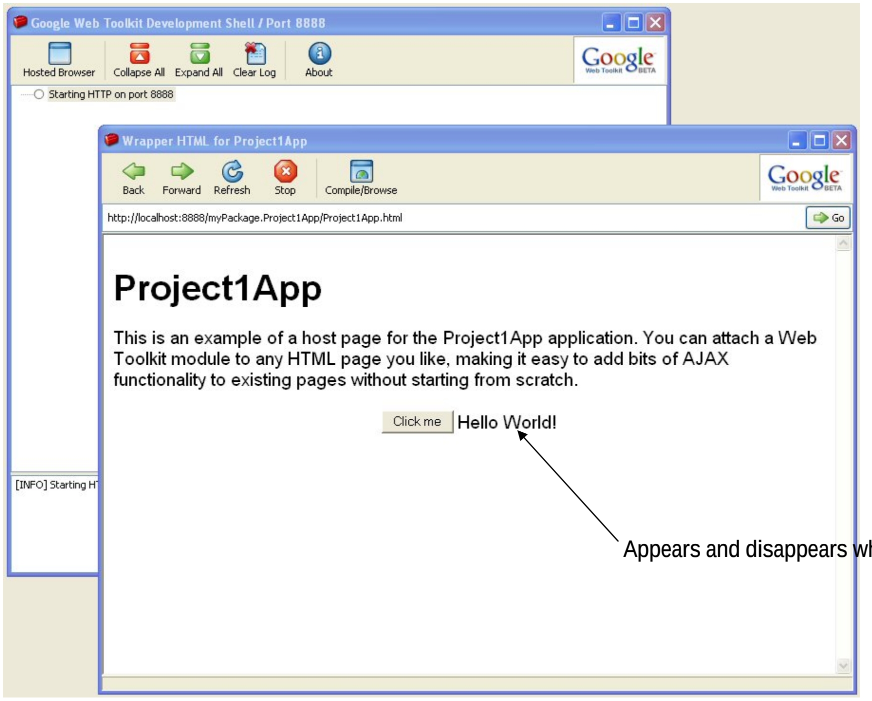
Main class: com.google.gwt.dev.GWTShell

Select this. Then press Run.

Inserted automatically.

Note: "hosted mode" means running entirely within IDE in Java.
- Code almost always runs identically once deployed in a regular browser, but deployment is cumbersome when there is server-side code, and will be covered in next section of tutorial.

Testing the Project in Hosted Mode: Result



Starting HTTP on port 8888

[INFO] Starting H

http://localhost:8888/myPackage.Project1App/Project1App.html

Project1App

This is an example of a host page for the Project1App application. You can attach a Web Toolkit module to any HTML page you like, making it easy to add bits of AJAX functionality to existing pages without starting from scratch.

Hello World!

Appears and disappears when you press button.

Developing GWT Applications

GWT Development Steps

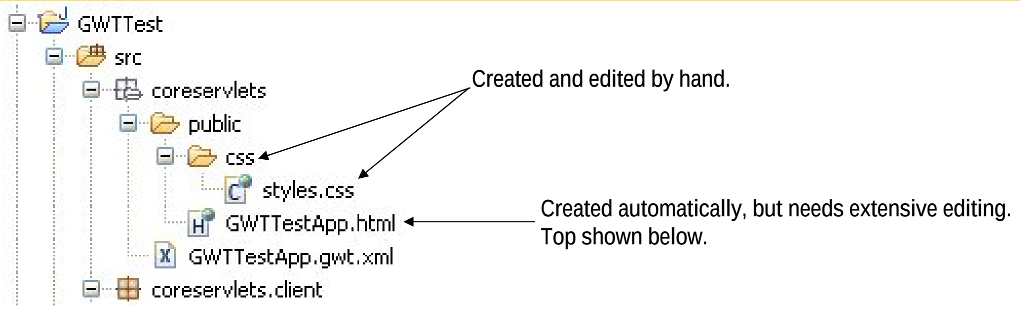
- **Edit HTML file and name sections**
  - Give id's to sections where buttons, textfields, and output will go
    - Usually div, span, td, or th elements
      - `<span id="sectionForButton1"></span>`
- **Edit main application class**
  - Class name given to applicationCreator (e.g., BlahApp.java)
    - Under src folder in Eclipse
    - Code goes in auto-generated onModuleLoad method
      - Delete all code that is in onModuleLoad to start with

  1. Create a control
     - Button, Checkbox, RadioButton, TextBox, TextArea, Label (plain text content), HTML (HTML content), etc.
       - `Button button1 = new Button("Press Here");`
  2. Give it an event handler
     - E.g., Button has ClickListener
       - `button1.addClickListener(new Button1Listener());`
  3. Insert it in HTML page
       - `RootPanel.get("sectionForButton1").add(button1);`

Example: Button that Generates Random Number (On Client)

- **Created GWTTest Eclipse project**
  - DOS> mkdir GWTTest
  - DOS> cd GWTTest
  - DOS> projectCreator –eclipse GWTTest
  - DOS> applicationCreator –eclipse GWTTest coreservlets.client.GWTTestApp
  - Imported into Eclipse with File > Import > etc.
- **HTML: GWTTestApp.html**
  - Need regions named randomNumberButton and randomNumberResult
- **Java: GWTTestApp.java**
  - Button named randomNumberButton
  - HTML (label) named randomNumberResult
  - Buttons event handler should insert value into HTML

HTML File: Details


GWTTest
src
coreservlets
public
css
styles.css
GWTTestApp.html
GWTTestApp.gwt.xml
coreservlets.client
Created and edited by hand.
Created automatically, but needs extensive editing. Top shown below.

```
<!DOCTYPE ...>
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>GWTTestApp</title>
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
<meta name='gwt:module' content='coreservlets.GWTTestApp'/>
</head>
<body>
<script language="javascript" src="gwt.js"></script>
```

This code edited by hand. Use normal HTML (usually xhtml).

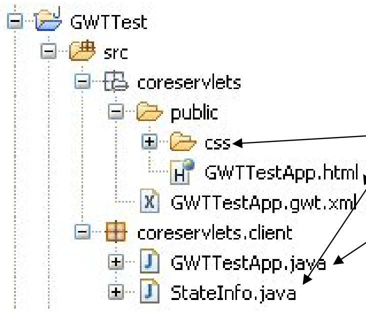
These two lines inserted automatically. Do not remove them. Format is different in GWT 1.4, but just use whatever gets built for you.

HTML File: Continued

```
...
<fieldset>
<legend>Client-Side Data</legend>
<table border="1">
  <tr><th>User Control</th>
      <th>Result</th></tr>
  <tr><td id="randomNumberButton"></td>
      <td id="randomNumberResult"></td></tr>
  ...
</table><br/>
</fieldset>
...
</body>
</html>
```

These names will be referred to in Java code. In these simple examples we create one HTML section for each low-level GWT widget. But in more advanced applications, it is common to put GWT panels into HTML sections, and then put multiple GWT widgets in the panels. It is even possible to build a GUI completely with GWT, where the HTML just has one blank section that will contain the main GWT panel.

Main Application Class: Details



Created by hand. Will be used/shown later in this tutorial.

Created automatically, but needs extensive editing. Top shown below.

```
package coreservlets.client;

import com.google.gwt.core.client.*;
import com.google.gwt.user.client.*;
import com.google.gwt.user.client.ui.*;
import com.google.gwt.user.client.rpc.*;

public class GWTTestApp implements EntryPoint {
  ...
  public void onModuleLoad() {
```

Created automatically.

Main Application Class: Continued

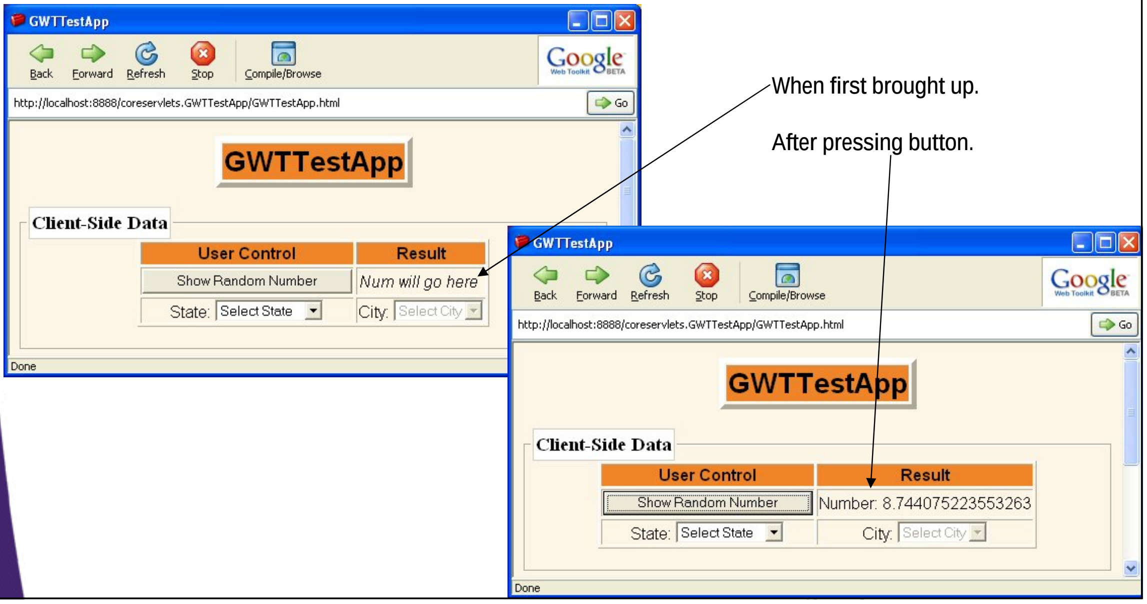
```
public class GWTTestApp implements EntryPoint {
  private HTML randomNumberResult;
  ...
  public void onModuleLoad() {
    Button randomNumberButton =
      new Button("Show Random Number");
    randomNumberResult = new HTML("<i>Num will go here</i>");
    randomNumberButton.addClickListener(new RanNumListener());
    RootPanel.get("randomNumberButton").add(randomNumberButton);
    RootPanel.get("randomNumberResult").add(randomNumberResult);
  }

  private class RanNumListener implements ClickListener {
    public void onClick(Widget sender) {
      randomNumberResult.setText("Number: " + Math.random()*10);
    }
  }
  ...
}
```

Matches id's in the HTML

Testing in Hosted Mode

- **Run app within Eclipse**
  - Run > Run > Java Application > GWTTestApp



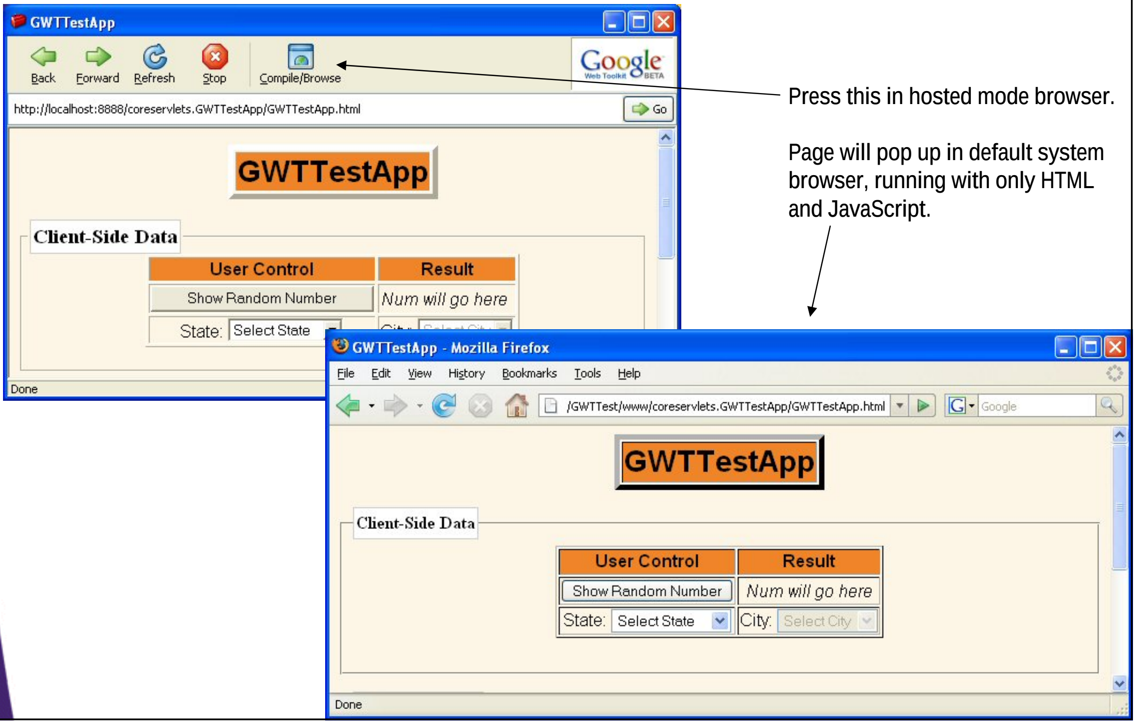
When first brought up.

After pressing button.

Testing in Web Mode

- **Idea**
  - Running in Web mode means running in a regular browser with all client-side Java code converted to JavaScript
- **Steps**
  - Run the application in hosted mode
  - Click the "Compile/Browse" button at top of hosted-mode browser. Result:
    - Creates a folder in your project called www
      - Note: in Eclipse, right-click on main project name and select "Refresh" to see the newly created directories and files
    - Creates a folder inside www matching module name
      - E.g., coreservlets.GWTTestApp
    - Puts HTML and translated JavaScript files in this folder
    - Loads the main HTML file in your default browser

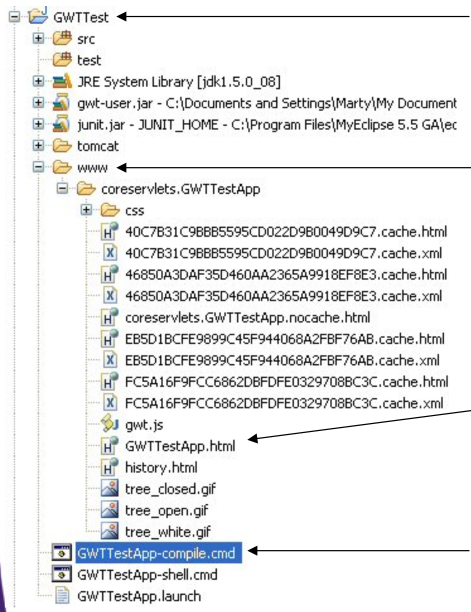
Testing in Web Mode


GWTTestApp
Back
Forward
Refresh
Stop
Compile/Browse
http://localhost:8888/coreservlets.GWTTestApp/GWTTestApp.html
Go
GWTTestApp
Client-Side Data
User Control
Result
Show Random Number
Num will go here
State: Select State
Done
GWTTestApp - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/GWTTest/www/coreservlets.GWTTestApp/GWTTestApp.html
City: Select City

Press this in hosted mode browser.

Page will pop up in default system browser, running with only HTML and JavaScript.

Testing in Web Mode: Resultant Files


```
GWTTest
  src
  test
  JRE System Library [jdk1.5.0_08]
  gwt-user.jar - C:\Documents and Settings\Marty\My Document
  junit.jar - JUNIT_HOME - C:\Program Files\MyEclipse 5.5 GA\ec
  tomcat
  www
    coreservlets.GWTTestApp
      css
      40C7B31C9BBB5595CD022D9B0049D9C7.cache.html
      40C7B31C9BBB5595CD022D9B0049D9C7.cache.xml
      46850A3DAF35D460AA2365A9918EF8E3.cache.html
      46850A3DAF35D460AA2365A9918EF8E3.cache.xml
      coreservlets.GWTTestApp.nocache.html
      EB5D1BCFE9899C45F944068A2FBF76AB.cache.html
      EB5D1BCFE9899C45F944068A2FBF76AB.cache.xml
      FC5A16F9FCC6862DBFDFE0329708BC3C.cache.html
      FC5A16F9FCC6862DBFDFE0329708BC3C.cache.xml
      gwt.js
      GWTTestApp.html
      history.html
      tree_closed.gif
      tree_open.gif
      tree_white.gif
  GWTTestApp-compile.cmd
  GWTTestApp-shell.cmd
  GWTTestApp.launch
```

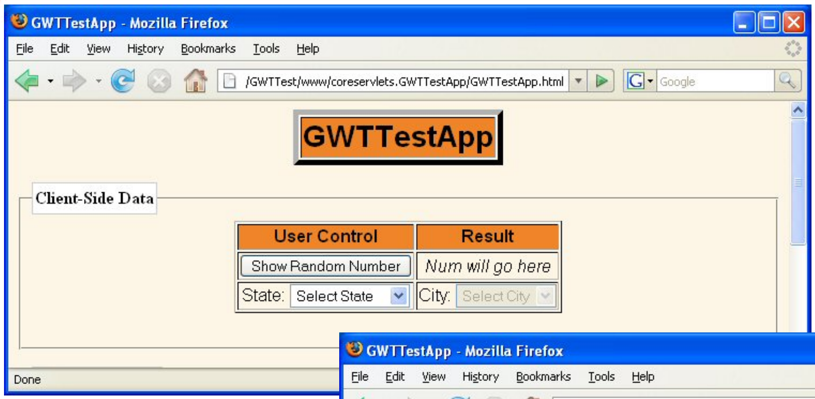
If you want to see the files that get built for Web mode, right-click here and select Refresh.

This folder was built when you pressed Compile/Browse.
Notes:
- To see these files in Eclipse, right-click on main project name (GWTTest at top) and choose Refresh
- Don't edit files in this folder. They will be overwritten next time you rebuild the client-side code.

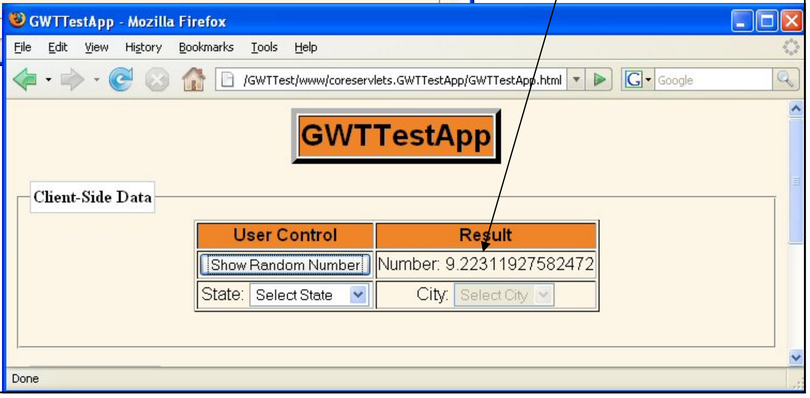
You can also navigate to the www folder outside Eclipse and load GWTTestApp.html in a browser.

You can also build the www folder and all client-side code by double-clicking this file. You can do so from within Eclipse.

Testing in Web Mode


GWTTestApp - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/GWTTest/www/coreservlets.GWTTestApp/GWTTestApp.html
Google
GWTTestApp
Client-Side Data

User Control	Result
Show Random Number	Num will go here
State: Select State	City: Select City


GWTTestApp - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/GWTTest/www/coreservlets.GWTTestApp/GWTTestApp.html
Google
GWTTestApp
Client-Side Data

User Control	Result
Show Random Number	Number: 9.22311927582472
State: Select State	City: Select City

Same functionality as in hosted mode, but code is now running entirely in JavaScript.

Using Auxiliary Java Classes

Big Idea

- **Use regular Java for client-side code**
  - Normal Java 1.4 syntax
    - Classes, methods, constructors, loops, control statements, etc.
  - Supporting classes
    - Static methods in the Math class and a few common data structures
      - array, ArrayList, Date, HashMap, HashSet, String, and Vector
  - Custom GWT classes
    - Class defined for each type of standard HTML control
      - Button, Checkbox, Image, Hyperlink, RadioButton, etc.
    - GWT also provides classes for groups of HTML controls
      - SimplePanel, TabPanel, ScrollPanel, Tree, FlexTable, PopupPanel, etc.
- **Java gets translated into JavaScript**
  - Click Compile/Browser or run BlahApp-compile.cmd to generate JavaScript
  - But you develop and test using Java only
- **Restrictions**
  - No Java 5 or Java 6 features
    - No for/each, generics, varargs, printf (or String.format), etc.
  - Custom classes must be placed in "...client" package.

GWT Development Steps

- **Edit HTML file and name sections**
  - Give id's to sections where buttons, textfields, and output will go

```
<tr><td>State: <span id="stateList"></span></td>
  <td>City: <span id="cityList"></span></td></tr>
```

- **Edit main application class**
  - Class name given to applicationCreator
    - Code goes in auto-generated onModuleLoad method

1. Create controls

```
stateList = new ListBox();
cityList = new ListBox();
```

2. Define event handlers

```
stateList.addChangeListener(new StateListener());
```

3. Insert in HTML page

```
RootPanel.get("stateList").add(stateList);
RootPanel.get("cityList").add(cityList);
```

Example: Linked Comboboxes
(Choosing State Results in List of Associated Cities)

- **Created GWTTest Eclipse project**
  - Same project as in previous example
- **HTML: GWTTestApp.html**
  - Need regions named stateList and cityList
- **Java**
  - Main Application class
    - Defines two listboxes
    - Attaches event handler to listbox that stores state names
  - StateInfo class (must be in ...client package)
    - Associates state name with array of cities
    - Defines static method with list of common states
- **Note**
  - In AjaxTags section, we did this process using server-side code. Using Java greatly simplifies client-side code.

HTML File

- **Same basic structure**
  - Still called GWTTestApp.html
  - Still need auto-generated meta and script tags
- **Body**

```
<fieldset>
<legend>Client-Side Data</legend>
<table border="1">
  <tr><th>User Control</th>
      <th>Result</th></tr>
  <tr><td id="randomNumberButton"></td>
      <td id="randomNumberResult"></td></tr>
  <tr><td>State: <span id="stateList"></span></td>
      <td>City: <span id="cityList"></span></td></tr>
</table><br/>
</fieldset>
```

Main Application Class

- **Same basic structure**
  - Still called coreservlets.client.GWTTestApp
    - As given to applicationCreator
  - Still has auto-generated features
    - implements EntryPoint
    - onModuleLoad method

```
package coreservlets.client;

import com.google.gwt.core.client.*;
import com.google.gwt.user.client.*;
import com.google.gwt.user.client.ui.*;
import com.google.gwt.user.client.rpc.*;

public class GWTTestApp implements EntryPoint {
  ...
  public void onModuleLoad() {
```

Main Application Class

```
public class GWTTestApp implements EntryPoint {
  private ListBox stateList, cityList;
  ...
  public void onModuleLoad() {
    stateList = new ListBox();
    populateStateList(stateList);
    stateList.setVisibleItemCount(1);
    cityList = new ListBox();
    cityList.addItem("Select City");
    cityList.setVisibleItemCount(1);
    cityList.setEnabled(false);
    stateList.addChangeListener
                          (new StateListener());
    RootPanel.get("stateList").add(stateList);
    RootPanel.get("cityList").add(cityList);
  }
```

Matches id's in the HTML

Main Application Class (Continued)

```
  private void populateStateList(ListBox stateList) {
    stateList.addItem("Select State");
    StateInfo[] nearbyStates =
      StateInfo.getNearbyStates();
    for(int i=0; i<nearbyStates.length; i++) {
      String stateName =
        nearbyStates[i].getStateName();
      stateList.addItem(stateName);
    }
  }
```

Main Application Class (Continued)

```
private class StateListener implements ChangeListener {
  public void onChange(Widget sender) {
    int index = stateList.getSelectedIndex();
    String state = stateList.getItemText(index);
    StateInfo[] nearbyStates =
      StateInfo.getNearbyStates();
    String[] cities =
      StateInfo.findCities(nearbyStates, state);
    cityList.clear();
    for(int i=0; i<cities.length; i++) {
      cityList.addItem(cities[i]);
    }
    cityList.setEnabled(true);
  }
}
```

Helper Class (StateInfo)

```
package coreservlets.client;

public class StateInfo {
  private String stateName;
  private String[] cities;

  public StateInfo(String stateName, String[] cities) {
    setStateName(stateName);
    setCities(cities);
  }
  public String getStateName() {
    return(stateName);
  }
  public void setStateName(String stateName) {
    this.stateName = stateName;
  }
  public String[] getCities() {
    return(cities);
  }
  public void setCities(String[] cities) {
    this.cities = cities;
  }
```

Must be in same package as main app. (...client). Only classes in this package are translated into JavaScript.

Helper Class (StateInfo, Continued)

```
private static String[] mdCities =
  {"Baltimore", "Frederick","Gaithersburg","Rockville"};
private static String[] vaCities =
  {"Virginia Beach","Norfolk","Chesapeake","Arlington"};
private static String[] paCities =
  {"Philadelphia","Pittsburgh","Allentown","Erie"};
private static String[] njCities =
  {"Newark", "Jersey City","Paterson","Elizabeth"};
private static String[] nyCities =
  {"New York", "Buffalo","Rochester","Yonkers"};
private static StateInfo[] nearbyStates =
  { new StateInfo("Maryland", mdCities),
    new StateInfo("Virginia", vaCities),
    new StateInfo("Pennsylvania", paCities),
    new StateInfo("New Jersey", njCities),
    new StateInfo("New York", nyCities)
  };
```

Can use arrays, ArrayList, HashMap, String, custom classes, etc.

Helper Class (StateInfo, Continued)

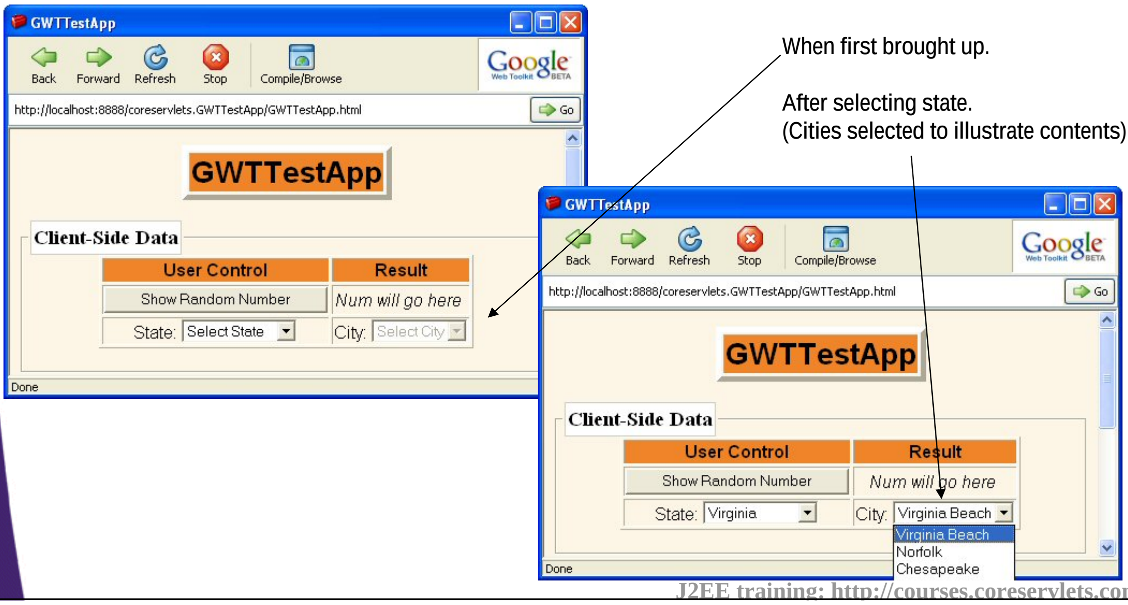
```
  public static StateInfo[] getNearbyStates() {
    return(nearbyStates);
  }

  public static String[] findCities(StateInfo[] states,
                                    String stateName) {
    for(int i=0; i<states.length; i++) {
      if(states[i].getStateName().equals(stateName)) {
        return(states[i].getCities());
      }
    }
    String[] unknown = {"Unknown state"};
    return(unknown);
  }
}
```

Cannot use new-style for/each loop from Java 5/Java 6 for client-side code. However, for server-side code (see later section), you can use whatever Java version your server supports.

Testing in Hosted Mode

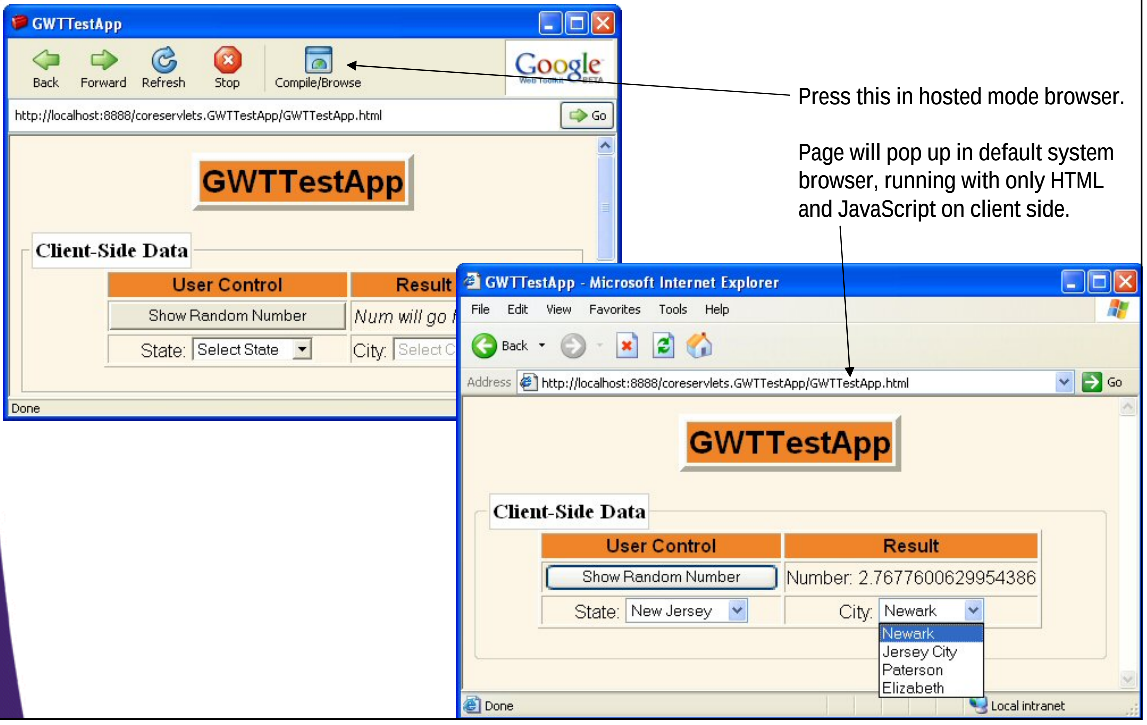
- **Run app within Eclipse**
  - Run > Run > Java Application > GWTTestApp



When first brought up.

After selecting state.
(Cities selected to illustrate contents)

Testing in Web Mode



Press this in hosted mode browser.

Page will pop up in default system browser, running with only HTML and JavaScript on client side.

Next Section: RPC and Server-Side Data

- **Big idea**
  - Event handlers can get data from server
  - Client defines data source and callbacks
  - Callbacks specify normal methods on server
    - No explicit networking
  - Server-side code just writes the methods
    - No explicit servlet methods, request handling, or response generation
    - Server-side code can use Java 5 or Java 6
    - Server-side code can import and call client classes
  - Server can return complex Java data structures
    - Including custom classes if they implement IsSerializable
- **Next tutorial section**
  - Illustrate development process and testing in hosted mode
  - Show steps needed to deploy in Web mode

Summary

- **Create an Eclipse application**
  - Use projectCreator and applicationCreator scripts
- **Edit auto-generated HTML file (BlahApp.html)**
  - Give id's to regions where controls will be placed
- **Edit auto-generated Java class (BlahApp.java)**
  - Edit onModuleLoad
    - Create controls
    - Give them event handlers
    - Insert controls into HTML page
- **Test in hosted mode**
  - Run > Run > Java application
- **Test in Web mode**
  - Press Compile/Browse in hosted mode browser
  - Or, load www/packageName.BlahApp/BlahApp.html in browser

Questions?