GWT Introduction

1. If you haven't already installed GWT and the Google Eclipse plug-in on your development computer, do so now. (This is for those using their own PCs. The class PCs already has all the software installed.)

2. Open your favorite browser and bookmark the main GWT documentation page at http://code.google.com/webtoolkit/overview.html
   Click on "Reference" on the left, then click on "SDK API Reference (Javadoc)", and bookmark that also. Use this latter link for details on methods to call in the Button class, TextBox class, ClickListener class, and so forth.

3. Make a new GWT project. Either do File > New > Web Application Project, or click on the blue "g" at the top of Eclipse.

4. Run the project in development mode. (R-click, Run As, Web Application. Then Rclick on the displayed URL, Copy, and paste it into your browser. If your browser doesn't already have the Google browser plug-in, install it now.)

5. Change the <title> of the HTML page to say "Hello, GWT". Change the main <h1> heading to match it. Reload the page in the browser to see the change.

6. Change the main Java class (src/package.client.AppName.java) so that the default textfield value is "Jane Java" instead of "GWT User". Change the button in the popup dialog box from "Close" to "Close dialog box". Reload the page in the browser to see the change.

7. Run the project in production mode. First R-click project and do Google > GWT Compile. Then change the URL in the browser by deleting the "?" and all text after it. If you have a browser that does not have the Google browser plugin installed, you can verify that you are really running in pure JavaScript by cutting/pasting the URL into that browser.

8. If you have a Java server like Tomcat on your PC and you know how to use it, deploy your app to that server. On the class PCs, the renamed "war" folder should go in C:\apache-tomcat-6.xxx\webapps, and C:\apache-tomcat-6-xxx\bin has the start and stop scripts.

GWT Basics

1. Make a new GWT app called “BasicsExercises” or some such. (Note that my solution project is called GwtBasicsExercises, so don’t use that name.)

2. This app won’t use server side code, so delete all extraneous files accordingly.

3. Make a push button (GWT class: Button) that, when pressed, inserts the current date into the page. You can base this code on the example in the GwtApp1 project. If you still have the previous project running, go to the “Development Mode” tab and click on the square red stop button. Run the new app in development mode.

4. Make two textfields (GWT class: TextBox) that contain numbers. Make a push button, that when pressed, reads the values from the two textfields, adds them together, and inserts the sum into the page. Run the app in development mode.

5. Run the app in production mode.

6. If you have a Java server that you know how to deploy to, deploy and run the app there.

Widget Event Handling

Make a new Google Web Application Project called “ExercisesEvents” (or some such the solution set is called GwtEventsExercises). Since you are not using any of the code in the “server” package, edit war/WEB-INF/web.xml and remove the servlet and servletmapping tags at the top. Then, delete all the classes in the “server” package.

1. Make a TextBox and a Button. What happens when you press the button should depend on what is in the textfield. If the textfield contains “English” when the button is pressed, “Hello” should be inserted in the page. If the textfield contains “Spanish” when the button is pressed, “Hola” should be inserted in the page. Otherwise, “Error” should be inserted. Test in development mode.

2. Make two textfields for numbers as in the previous exercises. But, this time omit the pushbutton. Instead, when the user types a number into either one, insert the sum of the two numbers into the page. Use a colored error message if the user enters an illegal entry. Test in development mode.

3. Make a list box that contains four choices (plus “Choose Range” at the top): 0-10, 0-100, 0-1000, and 0-10,000. When the user chooses one of the four choices, insert a random number in the specified range into the page. Test in development mode. What is the main problem with using list boxes for this type of action?

4. Make three radio buttons with labels Default, Red, Blue. Make Default the initial value. When the user selects one of the radio buttons, change the page background accordingly. Test in development mode.

5. Test the project in production mode.

6. Test the project in deployed mode.

GWT RCP

Make a new Google Web Application Project called “ExercisesRpc” (or some such).
Note that my solution set is called GwtRpcExercises.

1. Make a push button that, when pressed, inserts the current date into the page. Get the date from the server. Test in development mode.

2. Repeat the example from the first exercises where you have two textfields and a button, and after pressing the button, you get the sum inserted into the page. But, this time, send the two numbers to the server, and have it return the sum. Test in development mode.

3. Look in the “Customers” project, and you will see two classes for this problem: Customer and CustomerUtils. Copy Customer into the “client” subpackage of your project, and copy CustomerUtils into the server subpackage. You will have to edit both classes slightly to make them legal for the purposes of this exercise. The CustomerUtils class has a static method (CustomerUtils.getRichestCustomer()) that returns the customer with the highest bank account balance. Make a push button that, when pressed, inserts the id, first name, last name, and balance of the richest customer. Get the customer info from the server.

4. Make a textfield to collect a customer id, and pushbutton to send it to the server. When the button is pressed, print out the info for the customer you find. On the server, use CustomerUtils.getCustomer(id). Note that the legal ids are a1234, a1235, etc. Test in development mode.

5. Collect a list of customer ids in a textfield or text area. Send them to the server and return an array or List of customer objects. Make a table of the results. Test in development mode.

6. Test in production mode.

7. Test in deployed mode.

GWT Panels & Layouts

1. Make four tabbed panels. Start off with each panel containing an h1 heading that displays a random number. The number in each panel should be different, but the numbers won't change each time you reopen the same panel. Test in development mode.

2. Make a second set of tabbed panels. But this time, the content of each panel should be a HorizontalPanel containing, in order, an h1 heading with a random number, a Button, a TextBox, and a <ul> list. Test in development mode.

3. Make a third set of tabbed panels. It should be the same as the last one except that instead of a Button in the second entry, you should have a vertical column of four Buttons. Test in development mode.

4. Make a Grid with five rows and three columns. The first column in each row should be HTML that says "Enter value i" (where i matches the row number). The second column in each row should be a textfield. The last column in each row should be a button labeled "Do Something Cool with Value". Test in development mode.

5. Test in production mode.

6. Test in deployed mode.

GWT Widgets

Test all exercises in development mode.

1. Make a button that, when pressed, pops up a dialog box that contains <h1>Wow! Button was pressed!</h1>

2. Copy the code for one of the examples from your last set of exercises that used tabbed panels. Change it to use stacked panels (aka accordion panels) instead.

3. Make a date-input field and a button. When the button is pressed, take the date that the user entered, look up the year, and insert that into the page.

4. Make a Java array that contains a few city names beginning with A and B. (See http://www.usatoday.com/weather/resources/climate/wusaclim.htm for a short list.) Make an auto-completing text box that lets the user choose a city name. For bonus "credit", send the city name to Google when the user presses a button.

5. Make a menu bar where the top-level entries are "M", "N", and "O". When you click on an entry, you should get a list of the US states starting with that letter. Here are the states:
    - M: Maine, Maryland, Massachusetts, Michigan, Minnesota, Mississippi, Missouri, Montana
    - N: Nebraska, Nevada, New Hampshire, New Jersey, New Mexico, New York, North Carolina, North Dakota
    - O: Ohio, Oklahoma, Oregon Try the simple case first, where clicking on the letter shows the list of states, and clicking on a state pops up an alert (Window.alert) simply saying "A state was selected".

6. Update your state menu bar in two ways.
    - First, if the state has a common first word, just show that word, but clicking on the word should show a submenu. For example, under N, you would have Nebraska, Nevada, New, and North. Then under New and North you would have the full state names.
    - Second, if the user clicks on a full state name, pop up a dialog box or simple alert that says "A state was selected". If you are feeling more ambitious, you could have the popup show the state name. If you are feeling very ambitious, you could have the popup show the state name and the postal service abbreviation (e.g., MD for Maryland).

GWT and JSNI

1. Make a JavaScript function that generates a random number and puts the result in an alert (dialog) box. Make a GWT button that invokes the JavaScript function.

2. Make a JavaScript function that takes a number as an argument. It then generates a random number, multiplies it times the argument, and pops up the result in an alert box. Make a GWT button that reads a value from a textfield, converts it to a Double (use Double. parseDouble with a try/catch block for NumberFormatException), and passes that number to the JavaScript function.

3. Make a Java static method that takes a number as an argument. It then generates a random number, multiplies it times the argument, and returns the result. Make a JavaScript function that calls this Java method. Make a GWT button that reads a value from a textfield, converts it to a Double (use Double. parseDouble with a try/ catch block for NumberFormatException), and passes that number to the JavaScript function. The JavaScript function passes the number to the Java method, gets the result back, and puts the result in an alert box.