**AJAX application development process using GWT:**

|  |  |  |
| --- | --- | --- |
| **Tasks  What you'll do** | **Concepts  What you'll learn** | **GWT Tools and APIs What you'll use** |
| 1. [Create a GWT Project](http://www.gwtproject.org/doc/latest/tutorial/create.html) | Generate the files and directories you need to get started. | * Google Plugin for Eclipse * GWT command-line tool webAppCreator * Development Mode |
| 2. [Design the Application](http://www.gwtproject.org/doc/latest/tutorial/design.html) | Identify requirements, constraints, and implementation strategies. | Language constraints |
| 3. [Build the User Interface](http://www.gwtproject.org/doc/latest/tutorial/buildui.html) | Lay out the visual design and add user interface components. | GWT widgets and panels, the Root panel |
| 4. [Manage Events on the Client](http://www.gwtproject.org/doc/latest/tutorial/manageevents.html) | Handling mouse and keyboard events. | ClickHandler and KeyPressHandler interfaces |
| 5. [Code Functionality on the Client](http://www.gwtproject.org/doc/latest/tutorial/codeclient.html) | Maintain one code base for multiple browser implementations. Leveraging your Java IDE's features such as refactoring and code completion. | various GWT methods |
| 6. [Debug a GWT Application](http://www.gwtproject.org/doc/latest/tutorial/debug.html) | Debug the Java code before compiling it into JavaScript. Leverage your Java IDE's debugging tools by running the application in development mode. | Development Mode |
| 7. [Apply Style](http://www.gwtproject.org/doc/latest/tutorial/style.html) | Apply visual style to the application. Define the visual style in CSS. Set the class attributes on HTML elements programmatically. Change styles dynamically. Include static elements, such as image files. | * GWT module * GWT themes * application style sheet * GWT methods: addStyleName, addStyleDependentName,setStyleName * automatic resource inclusion |
| 8. [Compile a GWT Application](http://www.gwtproject.org/doc/latest/tutorial/compile.html) | Compile your client-side Java code into JavaScript. Test in production mode. Learn about the benefits of deferred binding. | GWT compiler |

## Creating a GWT application:

The Google Plugin for Eclipse contains a wizard for creating GWT applications. Here are steps for creating a starter application.

1. In the toolbar, click the New Web Application Project buttonicon.
2. Fill out the project details:
   1. Enter the project name "StockWatcher".
   2. Enter the package "com.google.gwt.sample.stockwatcher".
   3. Make sure Use Google Web Toolkit is checked and that Use default SDK (GWT) is selected.
   4. (Optional) If you are using Google App Engine, make sure Use Google App Engine is checked and that Use default SDK (App Engine) is selected.
   5. If you did not install the SDKs when you installed the Google Plugin for Eclipse, you should click Configure SDKs... to specify the directory where GWT (and the App Engine SDK if necessary) was unzipped.
3. Click the Finish button.

## Examining the project components

Let's examine some of the generated files and see how they fit together to form your GWT project.

### The module XML file

Open the module XML file, StockWatcher/src/com/google/gwt/sample/stockwatcher/StockWatcher.gwt.xml.

It contains the definition of the GWT module, the collection of resources that comprise a GWT application or a shared package. By default, StockWatcher inherits the core GWT functionality required for every project. Optionally, you can specify other GWT modules to inherit from.



In the module XML file, you specify your application's entry point class. In order to compile, a GWT module must specify an entry point. If a GWT module has no entry point, then it can only be inherited by other modules. It is possible to include other modules that have entry points specified in their module XML files. If so, then your module would have multiple entry points. Each entry point is executed in sequence.

By default, StockWatcher uses two style sheets: the default GWT style sheet, standard.css (which is referenced via the inherited theme), and the application style sheet, StockWatcher.css which was generated by webAppCreator. Later in this tutorial, you'll learn how to override the default GWT styles.

### The Host Page

Open the host page, StockWatcher/war/StockWatcher.html.

The code for a web application executes within an HTML document. In GWT, we call this the host page. For example, the host page for the StockWatcher project is StockWatcher.html.

The host page references the application style sheet, StockWatcher.css.

The host page references the path of JavaScript source code (generated by GWT) responsible for the dynamic elements on the page. The contents of the entire body element can be generated dynamically, for example, as it is with starter application. However, when you implement the StockWatcher application, you will use a mix of static and dynamic elements. You'll create an HTML <div> element to use as placeholder for the dynamically generated portions of the page.

#### Selecting Quirks Mode vs. Standards Mode

To provide better cross-browser compatibility, GWT sets the doctype declaration to HTML 4.01 Transitional. This, in turn, sets the browser's rendering engine to "Quirks Mode". If you instead want to render the application in "Standards Mode", there are a [number of other doctypes](http://hsivonen.iki.fi/doctype/) you can use to force the browser to this render mode. In general, GWT applications will work in "Standards Mode" just as well as "Quirks Mode", but in some cases using widgets like panels and such may not render correctly. This problem has been greatly improved since GWT 1.5, and more work is being done to solve this problem once and for all.

#### Preserving Browser History

GWT provides a mechanism for helping your application meet users' expectations of a web page, specifically in their ability to use the browser's back button in such situations as a multi-page wizard or a shopping cart/checkout scenario. The host page contains the iframe tag necessary for incorporating history support in your GWT application.

To learn more about managing browser history in a GWT application, see the Developer's Guide, [History](http://www.gwtproject.org/doc/latest/DevGuideCodingBasics.html#DevGuideHistory).

### The Application Style Sheet

Open the application style sheet, StockWatcher/war/StockWatcher.css.

A style sheet is associated with each project. By default, the application style sheet, StockWatcher.css, contains style rules for the starter application. In the[Applying Style](http://www.gwtproject.org/doc/latest/tutorial/style.html) section of this tutorial, you'll replace the style rules for the starter application, with the style rules for the StockWatcher application.

Just as for any web page, you can specify multiple style sheets. List multiple style sheets in their order of inheritance; that is, with the most specific style rules in the last style sheet listed.

### The Java source code

Open the source for the StockWatcher entry point class, StockWatcher/src/com/google/gwt/sample/stockwatcher/client/StockWatcher.java.

Currently, StockWatcher.java contains the Java source for the starter application. In this tutorial, you'll replace this code with the client-side code for StockWatcher.

The StockWatcher class implements the GWT interface [EntryPoint](http://www.gwtproject.org/javadoc/latest/com/google/gwt/core/client/EntryPoint.html). It contains the method onModuleLoad. Because the StockWatcher class is specified as the entry point class in StockWatcher's module definition, when you launch StockWatcher the onModuleLoad method is called.

The StockWatcher class inherits functionality via other GWT modules you included in StockWatcher's module definition (StockWatcher.gwt.xml). For example, when building the user interface, you'll be able to include types and resources from the package com.google.gwt.user.client.ui because it is part of the GWT core functionality included in the GWT module com.google.gwt.user.User.