



GWT IN ACTION

SECOND EDITION

Robert Hanson
Adam Tacy
Jason Essington
Christopher Ramsdale
Anna Tokke

MEAP



MANNING



**MEAP Edition
Manning Early Access Program
GWT in Action, Second Edition, version 6**

Copyright 2012 Manning Publications

For more information on this and other Manning titles go to
www.manning.com

Table of Contents

Part 1: Basics

1. Introducing GWT
2. Saying Hello World
3. Realizing a GWT application

Part 2: Next Steps

4. Creating your own widgets
5. Using client bundles
6. Interface design with UIBinder
7. Communicating with GWT-RPC
8. Using Request Factory
9. The Editor framework
10. Data presentation widgets
11. Using JSNI—the JavaScript Native Interface
12. Classic Ajax and HTML forms
13. Internationalization and localization

Part 3: Advanced

14. Advanced event handling and event bus
15. Building MVP-based applications
16. Dependency injection
17. Deferred binding
18. Automatically generating code
19. Metrics and code splitting

1

Introducing GWT

Google Web Toolkit (GWT) has filled a gap that most people hadn't even realized was there. Some people love JavaScript, but most web programmers aren't big fans of it - you need an in-depth knowledge of JavaScript, HTML, CSS and browser quirks to be sure that everything is safe and will work in all major browsers. One gets caught up using a dynamic language, and there isn't a lot in the way of development tools to help you with code completion, live syntax checking and so on.

Java, on the other hand, has everything you could wish for, an awful lot of development and testing tools. Even if you stick to just the ones that are free, there is a tremendous choice of comprehensive applications that have more features than you'll ever need.

But Java isn't JavaScript, so what is the point of discussing it?

Well, what if you took the Java source code and turned it into JavaScript? You would have all the advantages of the free development environments, debugging, testing tools and code coverage reports, and the end result would be JavaScript which can run in a browser.

And what if, while compiling, you created a different JavaScript file for each browser, each file tuned to that browser's quirks and foibles? And while we're at it, let's have an easy way to internationalize the site. And simplify AJAX. Oh, and I want a library of widgets with all the bells and whistles of rounded corners, animation, built in styles and so on. And I want it all to be free. And open-source. And I want browser history to work.

Meet GWT.

GWT is a toolkit. It is not a language, a "way", or a framework. It is a set of tools that are meant to provide an easy way to write sophisticated, reliable, AJAX applications using Java. It is not meant to take advantage of the Java runtime, but instead it makes use of the Java language and its existing tooling.

So what does this mean? Oversimplifying a bit, you write your code in Java and then compile it to JavaScript. The obvious question is why would you want to do that, and the answers are numerous. One possible answer might be that there are many more skilled

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=659>

Java developers in the world than there are seasoned JavaScript developers, but that isn't why GWT came to be. GWT was created in order to facilitate the development of large client-side browser applications with a focus on providing a great user experience, although it is applicable to a much wider range of usage as well.

NOTABLE QUOTE

Lars Rasmussen is the co-founder of two successful Google applications, namely Google Maps and Google Wave. When speaking about Google Wave Lars commented, "I think that we couldn't possibly have built something this rich without the Google Web Toolkit."

This focus on the user experience means the tools provided are geared to making the application download faster, start faster, and run faster. Now we can't claim that GWT is a golden solution. After all, everything you can do in GWT can be done with HTML, JavaScript, and CSS. But what GWT brings such a wide array of tools to the game it can conceivably replace every tool used by a traditional JavaScript developer, all within a single tool.

In this chapter we will provide you with an overview of GWT and help you set up your development environment. The overview will provide a glimpse into each tool in the toolkit, and provide references to where we cover the tool in detail.

When you have read this chapter you will have a better understanding of what GWT provides, and will have a development environment, leading us to the next chapter where we start writing code.

1.1 Exploring the toolkit

In this section we will provide an overview of the tools that make up the toolkit, explain a little about what they offer, and provide references to the chapters that cover the feature in detail, starting with the Java to JavaScript compiler.

1.1.1 Compiling and optimizing

The compiler is the core of the system. It takes your Java code and compiles it into JavaScript. During the process it will analyze your code in order to optimize it and remove any code that is not reachable by your application. It then generates JavaScript code that is condensed so that the file size is as small as possible. And this is only part of what the compiler does!

So the compiler is a busy bee, and throughout the book we will show you some of its secrets. One of these is code generation. The compiler may kick off one or more code generators based on what it finds in your code. This could be the generator that generates the code to serialize Java objects and send them to the server (Chapter x – GWT-RPC). Or it could be the generator that will allow you to define your interface in an XML file (chapter x - UIBinder). It could also be the generator that takes external files like CSS or images, and optimizes in a way that allows the browser to load them more quickly (chapter x -

ClientBundle). Or perhaps, you will create your own custom generator to fill a specific need (chapter x - Generators).

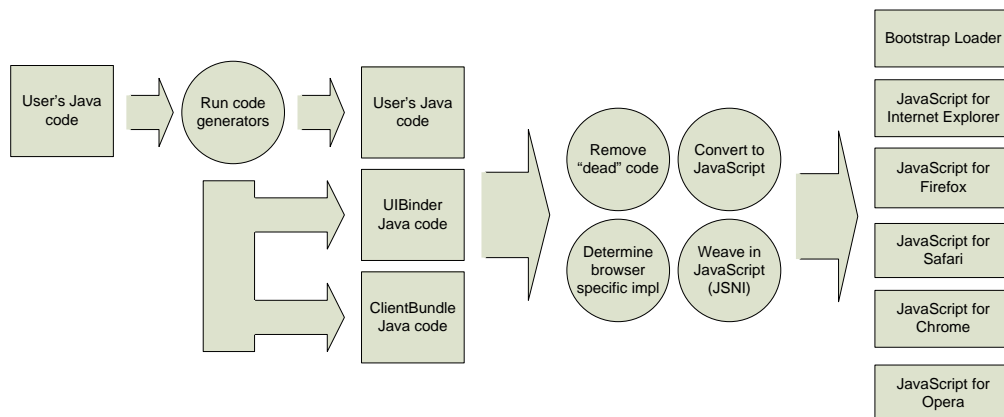


Figure x.x This can be considered an artist's rendering of what the GWT compiler is responsible for. This doesn't cover everything the compiler does, but it does provide a high-level overview of how Java code (on the left) is compiled into JavaScript code (on the right).

In addition to this the compiler has the ability to weave your Java code with existing JavaScript code. This is called the JavaScript Native Interface (JSNI), which allows you to embed JavaScript code within your Java classes in order to facilitate Java to JavaScript communication (chapter x - JSNI). Or course it all ends up as JavaScript in the end when the compiler weaves everything together.

But using JavaScript can be tricky because each browser has its own idiosyncrasies. That is where deferred binding comes in. Deferred binding allows you to create multiple implementations of the same functionality, perhaps one for Internet Explorer and one for all others, and the compiler will generate multiple JavaScript output files, one for each browser (chapter x – Deferred Binding).

So with multiple JavaScript files, for different browsers, the compilation process also generates a bootstrap loader. This loader is a JavaScript file that will run in your browser and load the correct application code file for that browser type.

As you can see, the compiler is the key to everything in GWT. But when someone comes to look at your application they don't really care about how you wrote the code, they care about what it looks like.

1.1.2 Powerful widgets and a layout binding engine

In GWT the fundamental building block of the user interface is the Widget, and they come in all shapes and sizes. And to group these Widgets you would use a specialized class of Widget called a Panel, which are Widgets that have the ability to contain other widgets. And

then there are Composites, which are Widgets encapsulate some other Widget, hiding the internal implementation from outside callers. All of which will be covered throughout this book.

Together Widgets, Panels, and Composites will allow you to piece together your user interface. GWT even provides a special class of widgets called Data Presentation Widgets that allow you to display a large amount of data to the user in a series of pages, like a search result listing (chapter x – Data Presentation Widgets). These are designed to be fast, to conserve memory in the browser, and to make it easy for you as the developer to provide access to extremely large data sets.

But not all Widgets are as full-featured as the Data Presentation Widgets. Some are just building blocks that you can use to build more complex widgets, like the Image, Button, and Hyperlink Widgets. So as you use GWT to build your interface you will naturally combine basic Widgets into complex Composites, then reuse those over and over (chapter x – Building your own widgets)

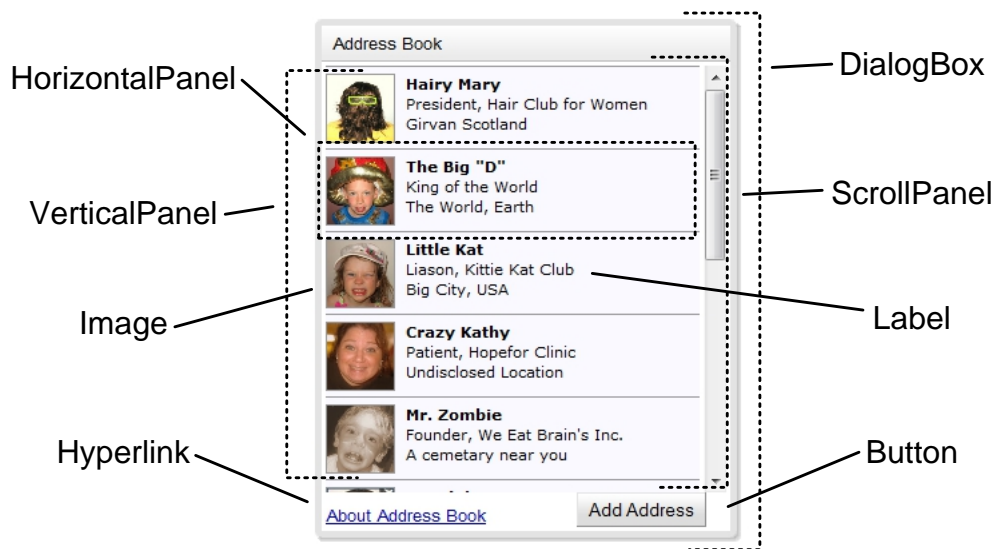


Figure x.x An example of a user-built Address Book widget. It is made up of at least eight different types of widgets and dozens of instances. Some of the widgets are used for layout like the VerticalPanel, and others will react based on user events like the Button.

If you find that your Composites are useful you may even bundle them in a JAR file and share them with other GWT developers. Or perhaps more likely you will want to take

advantage of the work of others and use third-party widget libraries in your projects. One of the most popular is the GWT Incubator Project¹, which is used by the GWT team to release beta Widgets and work out the kinks before permanently adding them to the GWT distribution. Then there are third-party libraries for drag-and-drop², Google APIs³ (maps, search, etc.), and more.

But one thing that initial versions of GWT taught the early adopters was that building large interfaces with dozens and dozens of Widgets is hard. The amount of code required made it time consuming to build an interface. This changed when a new tool was added to the toolkit called UIBinder (chapter x - UIBinder), which allows you to define your interface in XML code that looks a lot like HTML. This is more compact than Java code, allowing you to define complex interfaces with minimal effort.

Of course developing what your user sees is only part of the work involved in developing an interface, you need to handle user events too.

1.1.3 Event handling beyond JavaScript

At one end of the spectrum GWT allows you to register for plain old JavaScript events. This includes clicks on a button, focus on a checkbox, and mouse movements over a widget. In GWT these are called DOM⁴ events.

On the other end are events that are specific to a particular widget. For example when a user clicks on a date in the DatePicker widget it fires a ValueChangeEvent. These events are called GWT events. As you develop more advanced widgets you will create your own GWT events (chapter x – Advanced Event Handling).

In addition GWT provides a tool called the HandlerManager, which can be used as an event bus. An event bus is a messaging channel where an event producer sends an event, and any number of event listeners can receive the event and act on it. The important distinction between an event bus and basic event handling is that with an event bus the event producer and recipient don't know about each other, they only know about the message channel. This is a popular pattern that allows you to develop an architecture where components are loosely coupled. It works similar to IRC (Inter-Relay Chat), where you send a message to the channel, and everyone may respond if they wish to.

¹ The GWT Incubator project can be found at <http://code.google.com/p/google-web-toolkit-incubator/>.

² The GWT Drag-and-Drop project can be found at <http://code.google.com/p/gwt-dnd/>.

³ The GWT Google APIs project can be found at <http://code.google.com/p/gwt-google-apis/>.

⁴ DOM is short for Document Object Model.

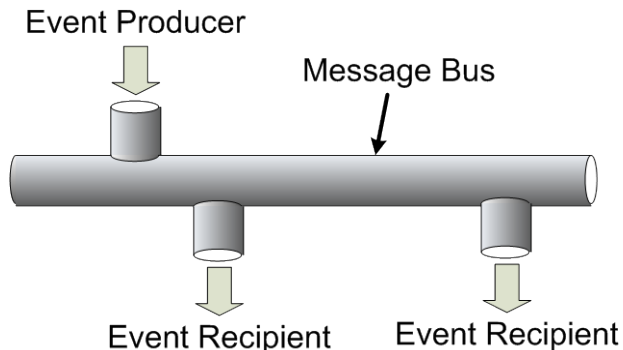


Figure x.x An event bus is like a messaging channel or pipe. A producer puts an event onto the bus, and any number of recipients can handle it. Producers and recipients are decoupled from each other because they connect to the message bus and not each other.

Speaking of communication, one type of communication that vital to most GWT applications is the ability to pass data to and from the web server.

1.1.4 Client-server communication

In a perfect Java-centric world you would be able to use Java on the server, Java to write your GWT-based application, and you would pass Java objects between the browser and the server. This is where GWT-RPC fits in (chapter – Getting a handle on GWT-RPC). It provides the tooling required to handle the serialization and transmission of Java objects to and from the server, even though the client-side code in this is really JavaScript.

This can be a relatively difficult problem given the mismatch in languages. On the server this isn't so hard because of Java's reflection capabilities. On the server the application can inspect the Java object to determine its field types and values. In the browser this just isn't possible, reflection won't work.

In order to solve this problem GWT-RPC makes use of a code generator like we spoke about in section x.x. This allows GWT to generate all of the serialization and communication code at compile time. But still, it isn't a magical process, and you will need to use some annotations to help the code generator determine what to do. We will cover all of that in great detail.

But for those of you interested in communicating with the server Ajax style, GWT can do that too. The tool for that job is RequestBuilder, and it allows you to send GET and POST requests to the server (chapter x – RequestBuilder and HTMLForm). But RequestBuilder alone isn't particularly useful, especially considering that most remote services send either JSON⁵, JSON with Padding (JSONP), or XML⁶ data. To allow usage of these formats GWT

⁵ The JavaScript Object Notation (JSON) specification can be found at <http://www.json.org/>.

⁶ The Extensible Markup Language (XML) introduction and specification can be found at <http://www.w3.org/XML/>.

provides an API for reading and writing JSON data, and an XML parser for reading XML. We will cover both of these API's in the RequestBuilder chapter.

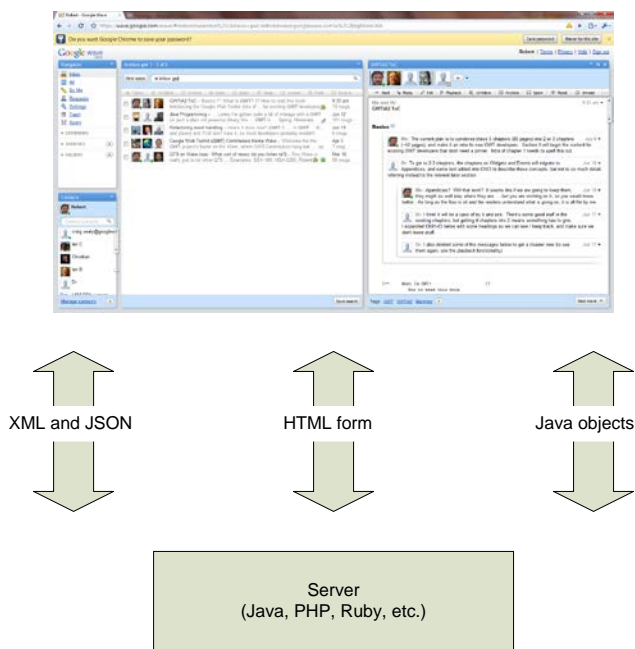


Figure x.x GWT provides a number of tools for passing data between the server and browser. This includes Ajax style communication for passing XML and JSON data, HTML forms for form data, and GWT-RPC for passing serialized Java objects.

And of course, GWT also has support for the original client-server communication tool used by the browser, the HTML form (chapter x – RequestBuilder and HTMLForm). There are some differences on how GWT handles form submissions though. For example, in a standard form submission the page in the browser will change. In GWT we don't usually want to load another page into the browser, and GWT provides a way to do this. We will cover all of this when we discuss forms.

So hopefully you will agree that GWT provides some serious tools for developing a rich user interface. But there is more to it than that. GWT also provides you, the developer, the tools you need to develop your application rapidly.

1.1.5 Simplified development and debugging

When doing traditional Java development there are some things that are less than optimal. When you modify the code for a servlet (Spring Controller, Struts Action, etc.) you need to

recompile your code, deploy it to a server, then start the server. That can take a lot of time depending on what you are doing.

Now think about how you might test a change in GWT. You need to compile your client-side code to JavaScript, compile your server-side Java code, deploy all of it to a server, and start the server.

Besides being a slow prospect, how might you handle debugging an application that lives as Java on the server and as JavaScript in the browser? How to you debug clicks to the user interface, and remote calls to the server.

To solve this problem GWT has a closely related project named the Google Eclipse Plugin. This tool provides support for GWT as well as support for Google App Engine⁷. For GWT the plug-in provides wizards, single click compiling, auto-completion support, and one click access to development mode, where you can test your application without deploying it to an external server.

If you are a fan of a competing IDE to Eclipse we completely understand that you might not be excited by this news. And although you can develop GWT code using any IDE (or no IDE at all), we strongly recommend that until you get your feet wet with GWT you should use Eclipse. In section x.x we will cover installing both Eclipse and the Google Eclipse Plugin and in chapter 2 we will show you how to use it.

Now we did mention that the Google Eclipse Plugin provides auto-completion support. One area where it provides that is when writing code to bridge the gap between Java and JavaScript.

1.1.6 Integration with JavaScript

For the most part you can write your GWT applications using Java only, but there are always going to be exceptions to the rule. Take the case where you are just moving to GWT and you have hundreds or thousands of lines of JavaScript code that you want to reuse. Or perhaps you need to access some new JavaScript API that GWT doesn't directly support yet. Or maybe there is a third-party JavaScript API that you need to use. GWT supports all of these use cases with the JavaScript Native Interface, also known as JSNI (chapter x - JSNI).

The way JSNI works is that it allows you to embed JavaScript code right inside your Java code by making use of a Java comment block along with the "native" Java keyword. The purpose of the "native" keyword in Java is to denote that a method is implemented in another language like C, C++, or even assembly. And although GWT isn't using the keyword in the way that its creators envisioned, it is a creative solution to our needs.

Even without any background knowledge of JSNI, this following example should be easy to follow. It simply prints a specified message a specified number of times in the browser.

⁷ Google App Engine can be found at <http://code.google.com/appengine/>.

Listing 1.1 An example of a JSNI method

```
public native String printMessageTimes (String msg, int times) |#A
/*- {
    for (x = 1; x <= times; x++) {
        var prefix = '#' + x + ' - ';
        $doc.write(prefix + msg + '<br />');
    }
} -*/;
```

#A Java signature
#B JavaScript body

If you look closely you can see that the code in the method is really inside of a Java comment block. And just to prove that it really is JavaScript the example code fails to declare the variable `x`, uses `var` to declare the variable `prefix`, and uses single-quotes around string constants. The only thing that isn't standard JavaScript is the `$doc` variable, which is simply an alias GWT uses to represent the JavaScript `document` object.

In the Java IDE this code is ignored because the code has been hidden in a Java comment, and by declaring the method as `native` the IDE doesn't expect a method body. But although this code may be hidden to the standard Java compiler, the GWT compiler will extract this JavaScript code and weave it into the final output.

But this is only part of what there is to know about JSNI. Besides being able to call JavaScript code from Java you can also call your Java code from JavaScript. In addition there is a GWT feature called JavaScript Overlay Types which allows you to wrap a JavaScript object in a Java class. There is a lot to know about JSNI, which is why we have dedicated a whole chapter to explain it all.

So at this point you can see that GWT isn't just about widgets, and provides a very full solution to building rich user interfaces. One such feature GWT provides is a solution to the broken back button.

1.1.7 History support

When Ajax was conceived in 2005⁸ one thing it did was to break the back button. What this means is that a typical browser user is used to the idea of using the back button in the browser to go back to the last page they were looking at. Ajax though has the notion of not changing the page you are looking at, and instead it just manipulates the page with JavaScript to alter its contents.

So when the typical browser user used one of these Ajax applications and hit the back button, as that became the natural thing to do, they were surprised that they were directed to not the last thing they were looking at within the Ajax application, but instead were directed to whatever they were looking at before they came to the Ajax application.

⁸ Ajax is an acronym for Asynchronous JavaScript and XML, coined by Jesse James Garret in his essay "Ajax: A New Approach to Web Applications", <http://www.adaptivepath.com/ideas/essays/archives/000385.php>.

This isn't a new problem, but it became very noticeable in 2005 when many developers rushed to deploy Ajax applications. So when GWT was released in 2006 it came with history support built in (chapter x - ???).

GWT's solution isn't new, and is pretty simple when you think about it. What GWT does is uses a hidden HTML iframe, and each time you go to another "page" in your application it changes the URL of the hidden frame. To your browser it considers this URL change as going to another page, and adds it to your browsing history. So when you click the back button in your browser all you are doing is changing the URL of the hidden frame to the last URL it had in the history. This in turn triggers an event in your application that you can handle.

It is far from automatic, but with proper usage you can allow your users to use the browser's forward and backward buttons to navigate through your GWT application. In chapter x we provide everything you need to know in order to use this feature.

Another feature that makes GWT a complete solution is its support for internationalization.

1.1.8 Internationalization - Sprechen Sie Deutsch?

GWT doesn't provide any translation capabilities, but it does provide a framework where you can deploy your application in any number of languages (chapter x - Internationalization). In Java the typical way this is handled in web application is to have the web framework you are using detect the language requested by the browser, and use a properties file with messages specific to that language.

This works great on the server, but has some failings when you try to do this for a browser application. For example, if you want to support 10 languages, your application now needs to hold 10 translations of every phrase used in the application. This translates to a larger download and slower application startup for the user.

GWT's approach is to handle this the same way it handles the multiple browser-specific implementations of the same Java code that we discussed in section x.x. And that is to generate a separate JavaScript file for each language. That way you can support one language or one hundred, with no output file size penalty.

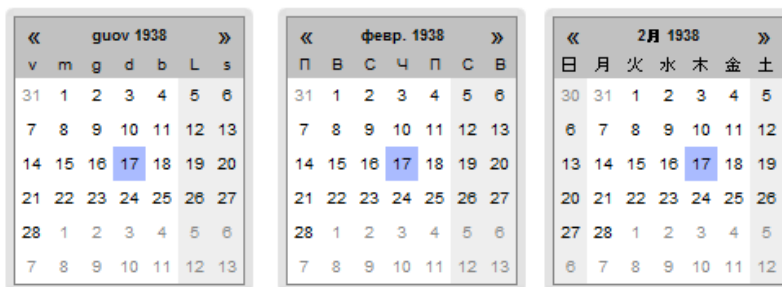


Figure x.x Pictured here is the GWT DatePicker widget using three different locale settings. From left to right calendars for Northern Sami, Russia, and Japan.

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=659>

In addition GWT provides support for formatting dates and currency in dozens of different locales. This support is used by the DatePicker widget, which allows it to not only use locale specific month names but also to make sure the week starts on the accustomed day for the region.

And with that we will wrap up the GWT feature tour. As you can see, GWT is a full-featured toolkit, and new features will be added as a need becomes apparent. So now let's switch gears and prepare to do some coding in the next chapter. In order to do that you will first need to setup your develop environment.

1.2 *Setting up your development environment*

We understand that everyone is different, with different needs and affinities, but unfortunately we can't cover every IDE or "way" to work with GWT. In this book we will make use of the tools that are at the time of this writing a best fit for use with GWT. If you are new to GWT we strongly suggest that you follow our lead and use the tools we recommend. As you gain proficiency in GWT you will find that it is much easier to use other tooling that we don't cover in this book.

For this book the "way" is to utilize the Eclipse IDE along with the Google Eclipse Plugin and GWT Designer, which is also an Eclipse plug-in. In addition to the many wizards provided by these plug-ins they also offer code completion for GWT specific files and a visual designer for developing the layout of your application. And best of all they are completely free to use.

Besides Eclipse and the mentioned plug-ins you will also need to have the Java JDK installed as well as the Development Mode Browser Plugin. In this section we will walk you through installing these tools, providing you with a good development environment for working with GWT.

Below is a summary of the tools we will install in this chapter.

Development tools for use with Eclipse and the Google Eclipse Plugin

Java JDK	Win/Linux: http://java.sun.com/javase/downloads Mac OSX: http://developer.apple.com/java
Eclipse IDE	http://www.eclipse.org/downloads Recommended version: Eclipse IDE for Java EE Developers
Google Eclipse Plugin	http://code.google.com/eclipse/docs/download.html
GWT Designer	http://code.google.com/webtoolkit/tools/download-gwt designer.html
GWT SDK	http://code.google.com/webtoolkit/download.html The SDK can also be installed via the Google Eclipse Plugin update site.
Dev Mode Browser Plugin	http://gwt.google.com/missing-plugin/MissingPlugin.html Allows you to run GWT in "development mode" with your browser.

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=659>

If for some reason you can't or are not willing to use Eclipse and the mentioned plug-ins, your journey will be a little more difficult, but not impossible. GWT provides a command-line tool for creating and compiling GWT projects. In addition the project creation tool generates and Ant build file that contains macros for compiling, testing, and packing your project. If these are of interest to you we cover using them in Appendix X. A modified summary of the tools you would need if you followed that path are listing below.

Development tools for use with non-Eclipse environments

Java JDK	Win/Linux: http://java.sun.com/javase/downloads Mac OSX: http://developer.apple.com/java
GWT SDK	http://code.google.com/webtoolkit/download.html
Apache Ant	http://ant.apache.org/bindownload.cgi Used for compiling and launching dev mode when not supported by the IDE.
An IDE	Optional.
Dev Mode Browser Plugin	http://gwt.google.com/missing-plugin/MissingPlugin.html Allows you to run GWT in "development mode" with your browser.

In addition to the core development tools that you will need to work with GWT there are several other useful tools that you might want to include in your environment, some of which we make reference to throughout this book. Perhaps these are best to be installed after you have a working GWT environment, so we present them merely as a reference.

Optional development tools

Speed Tracer	http://code.google.com/webtoolkit/download.html A diagnostic plug-in for Chrome
Firebug	http://getfirebug.com/ A diagnostic plug-in for Firefox.
Firebug Lite	http://getfirebug.com/firebuglite A diagnostic tool for non-Firefox browsers.
Gwt4nb	https://gwt4nb.dev.java.net/ A NetBeans GWT plug-in
Maven	http://maven.apache.org/ A build management tool, often used in place of Ant. When using Maven with GWT it is recommended to use the GWT Maven Plugin (http://mojo.codehaus.org/gwt-maven-plugin/).
Spring Roo	http://www.springsource.org/roo A code generation tool that includes support for GWT.
SpringSource Tool Suite	http://www.springsource.com/products/sts

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=659>

Hopefully a lot of these tools will look familiar, and if not we hope that you will explore what they have to offer. So without delay, let's run through the detailed installation of the recommended environment. We begin with the Java JDK.

1.2.1 Installing the Java JDK

The first thing you will need is the latest version of Java. We have made the assumption that our readers already know the basics of Java, so it is likely that you already have the Java JDK installed. For developing with GWT we suggest using the latest version of Java, although all versions starting with Java 5 should suffice. The Java JDK for Windows and Linux can be downloaded from <http://java.sun.com/javase/downloads>. For Mac OSX you can download the latest JDK from Apple's Java developer site located at the URL <http://developer.apple.com/java>.

If you already have Java installed you should verify that you have the JDK (Java Development Kit) installed. This differs from the JRE (Java Runtime Environment) in that it includes tools for compiling Java code. This isn't strictly needed for compiling GWT applications since the GWT compiler will handle that, but it will be required if you plan on using Java on the server-side of RPC calls, which we discuss in chapter x.

If you are unsure what you have, open a command prompt and run `javac -version`. If `javac` is not found then you don't have the JDK installed, or it is not on your path. If you do have `javac` it will let you know what version you have with output that looks like the following.

```
$ javac -version
javac 1.6.0_20
```

If you have the JDK installed but it isn't in your path, you may want to consider adding it. In particular, if you plan on using an IDE other than Eclipse you will need to run command line tools, and these tools will expect the `java` command will be found in your path.

If you are not sure how to add a directory to your path you can visit Appendix X where we provide instructions for Windows, Linux, and Apple operating systems.

Next we will need to install Eclipse.

1.2.2 Installing Eclipse

Eclipse is an open-source Integrated Development Environment (IDE) with support for Java, PHP, C++, and many other languages. In addition it has a rich plugin ecosystem, including a wide range of tools useful for web developers.

Eclipse can be downloaded from <http://www.eclipse.org/downloads/>. There are several available packages for Eclipse. Each comes with the core Eclipse workbench along with a set of plug-ins that are typically appropriate for a specific kind of developer. You can use any of

these, but we suggest installing the Eclipse IDE for Java EE Developers, which has a good assortment of tools for Java developers.

Installing Eclipse couldn't be easier, simply uncompress the distribution and place the directory somewhere on your hard drive. There is no installation, and if you are running Eclipse on Windows it won't alter your registry file. If you have problems downloading or uncompressing the distribution, consult the Eclipse installation guide for help, located at http://wiki.eclipse.org/FAQ_Where_do_I_get_and_install_Eclipse%3F.

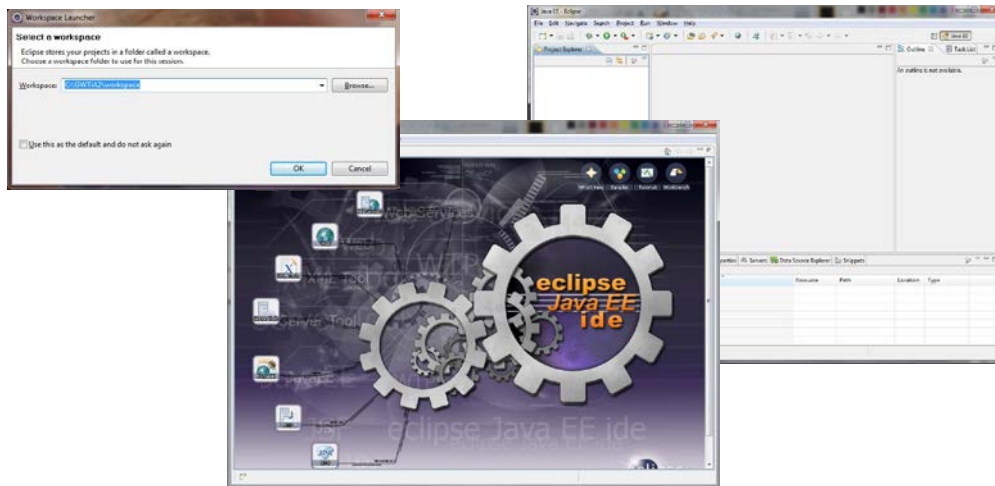


Figure x.x When you run Eclipse for the first time it will ask you for the location to store your projects (left), also known as your Workspace. After selecting a location you will see a Welcome screen that contains Eclipse documentation (center). Closing the Welcome tab will bring you to the editor (right).

Once downloaded and uncompressed, you should run the eclipse executable to verify that it is working properly. On startup it will ask you the location of your Workspace as shown in figure x.x. This is a directory where Eclipse will store configuration information and is usually where you will store your projects. Select a suitable empty directory when prompted. This will bring you to the Welcome screen.

Once at the Welcome screen you may want to explore some of the documentation provided, or close the Welcome tab to go to the editor.

When you get to the editor we can now work on getting the Google Eclipse Plugin installed.

1.2.3 Installing the Google Eclipse Plugin

When installing plug-ins for Eclipse you use what is referred to as the Install New Software tool. This wizard is located under the Help menu in Eclipse. In this section we will walk you through using this tool.

The first step is to get the location of the Update Site. An Update Site is a URL to a web server that is providing the Eclipse plug-in. For the Google Eclipse Plugin there are different Update Site URL's depending on which version of Eclipse you are running. You will need to visit the Google Eclipse Plugin website, located at <http://code.google.com/eclipse/docs/download.html>, to find the appropriate URL.

The download page will list different update URLs for different versions of Eclipse. The easiest way to know which version of Eclipse you are running is to look at the splash screen when you start Eclipse, which will display the version name on the splash screen.

If you are unable to determine the version from the splash screen, which could happen if you are using a distribution of Eclipse from another vendor (e.g. the SpringSource Tool Suite), you will need to find it in the Help menu. To find the version of Eclipse, or the version of any installed plug-in, you would go to Help > About on the menu bar. On the About screen there will be several buttons, as pictured in figure x.x. You will want to click on the button that has the tool tip "Eclipse.org". Clicking on this will bring up the versions of the individual features. The feature version you are interested in is the "Eclipse Platform".

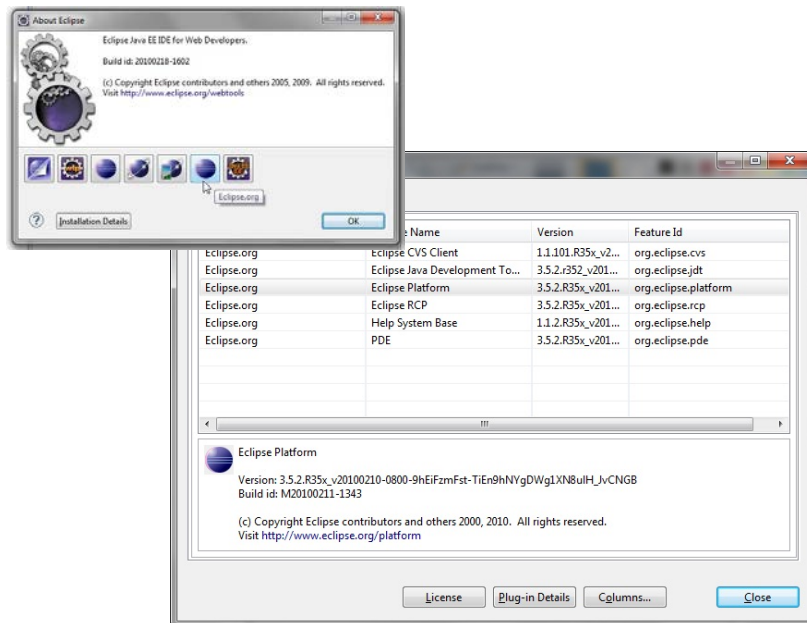


Figure x.x The Eclipse About dialog box (left) includes several icons representing the different organizations that provided the features and plug-ins you currently have installed. Clicking on the icon for Eclipse.org will present the feature versions (right), including that of the Eclipse Platform.

Once you have determined the version of Eclipse you are running and the URL of the update site for your version, you can install the Google plug-in. To do this, in Eclipse click Help > Install New Software. This will bring up the Install dialog as shown in figure x.x.

In the Install dialog you can click the Add button, which will bring up the Add Site dialog. In this dialog you should supply "Google Eclipse Plugin" for the name field, and the Update Site URL for the URL field. Then click OK when complete.

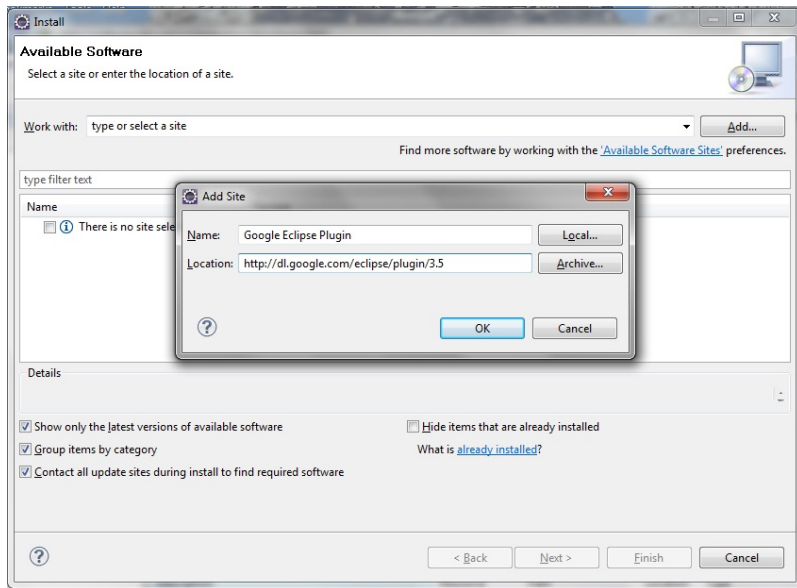


Figure x.x The Eclipse Add Site dialog is used to add a new URL to Eclipse where plug-ins can be downloaded. The Add Site dialog is displayed by clicking the Add button on the Install dialog, which is shown in the background.

This should bring you back to the Install dialog, which will now provide a list of plug-ins that can be installed, as seen in figure x.x.

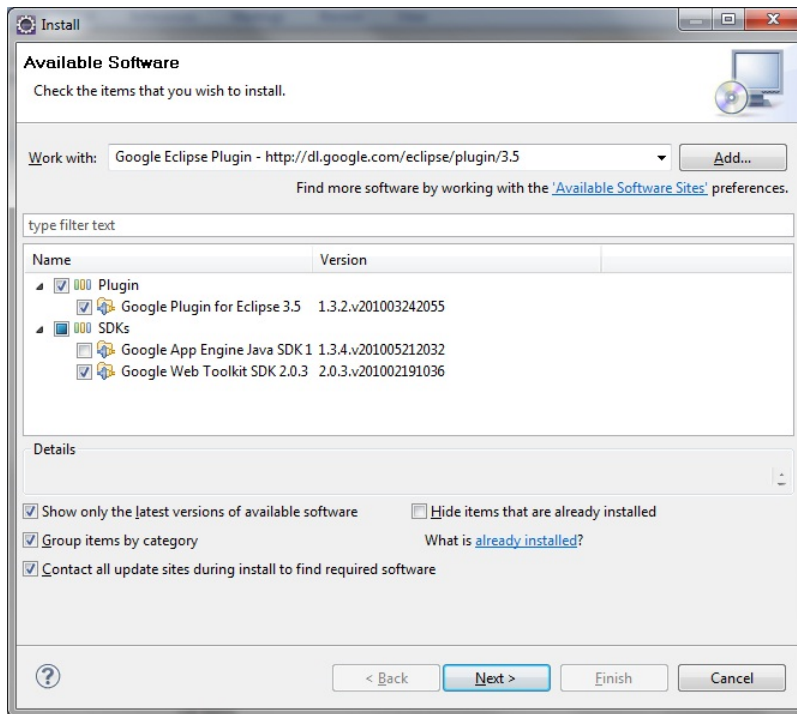


Figure x.x Once you add the update site for the Google Eclipse Plugin it will present you with a list of software available for installation.

You will want to select the options for the Google Plugin for Eclipse, and the Google Web Toolkit SDK, then click Next. Continue the process by following the instructions provided. Once complete it will ask you to restart Eclipse to finish the installation.

Next we want to install a second plug-in that will provide access to additional wizards and a visual designer tool.

1.2.4 Installing GWT Designer

In August of 2010 Google acquired Instantiations, a manufacturer of development tools, including GWT Designer. GWT Designer is a plug-in to aid with the development of GWT applications, and its most important feature is a visual designer named WindowBuilder Editor. Figure x.x shows Eclipse with a GWT application opened in the WindowBuilder editor so that you can get a sense of what this tool looks like.

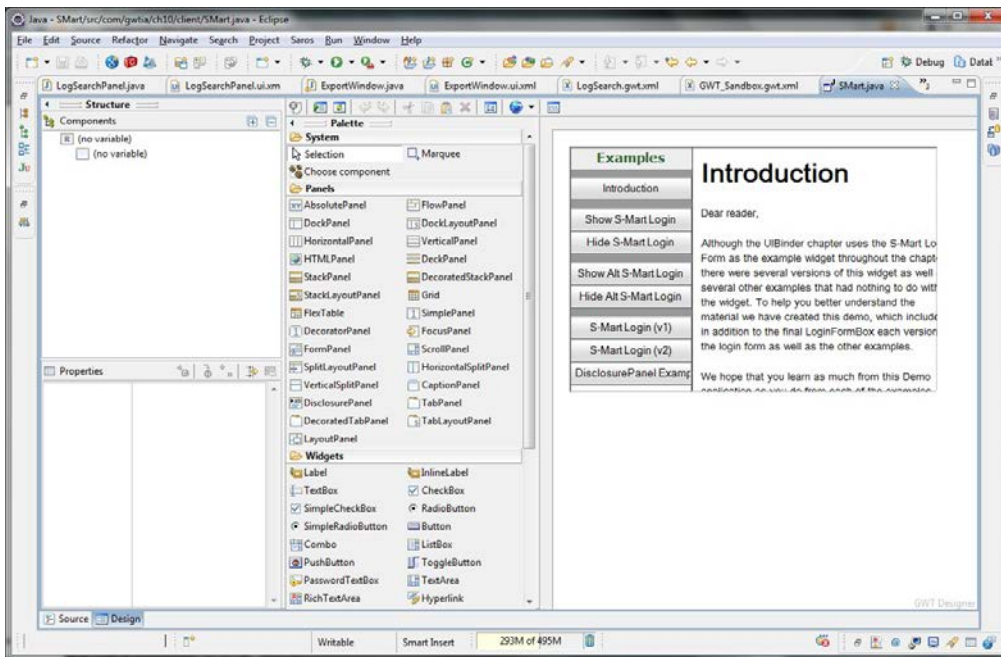


Figure x.x shows off the features of GWT Designer's visual editor. In this case we have opened the S-Mart examples that we discuss in the GWT-RPC chapter of the book (chapter X).

There is some overlap in functionality between GWT Designer and the Google Eclipse Plugin. In fact a lightweight version of GWT Designer is already included in the Google Eclipse Plugin. If you are new to GWT we suggest that you stick with only the Google Eclipse Plugin, but as your progress you may need some of the extra features that GWT Designer provides. Some of these features include integration with third-party widget frameworks like GXT⁹ and SmartGWT¹⁰.

Installing the GWT Designer plug-in is done in the same way that we installed the Google Eclipse Plugin, which is by using the Add New Software tool. In order to install the plug-in you need to know which version of Eclipse you are running. If you don't know, refer to the previous section on installing the Google Eclipse Plugin (section 1.2.3) for how to determine the version.

The URL for the update site can be found by going to the GWT downloads page located at <http://code.google.com/eclipse/docs/download.html>, and clicking the button for GWT

⁹ GXT (a.k.a. Ext-GWT) is a product of Sencha found at <http://www.sencha.com/products/extgwt/>.

¹⁰ SmartGWT is an open-source project found at <http://code.google.com/p/smargwt/>.

Designer. The page will provide a different update URL for each version of Eclipse. Find your version and make a note of the URL.

Once you have determined the version of Eclipse you are running and the URL of the update site for your version we follow the same steps as we did in the previous section, which is to Help > Install New Software. This will display the Install dialog.

In the Install dialog you will then click the Add button, which will bring up the Add Site dialog. In this dialog you should supply “GWT Designer” for the name field, and the Update Site URL for the URL field. Then click OK when complete.

This should bring you back to the Install dialog, which will now provide a list of plug-ins that can be installed, as seen in figure x.x.

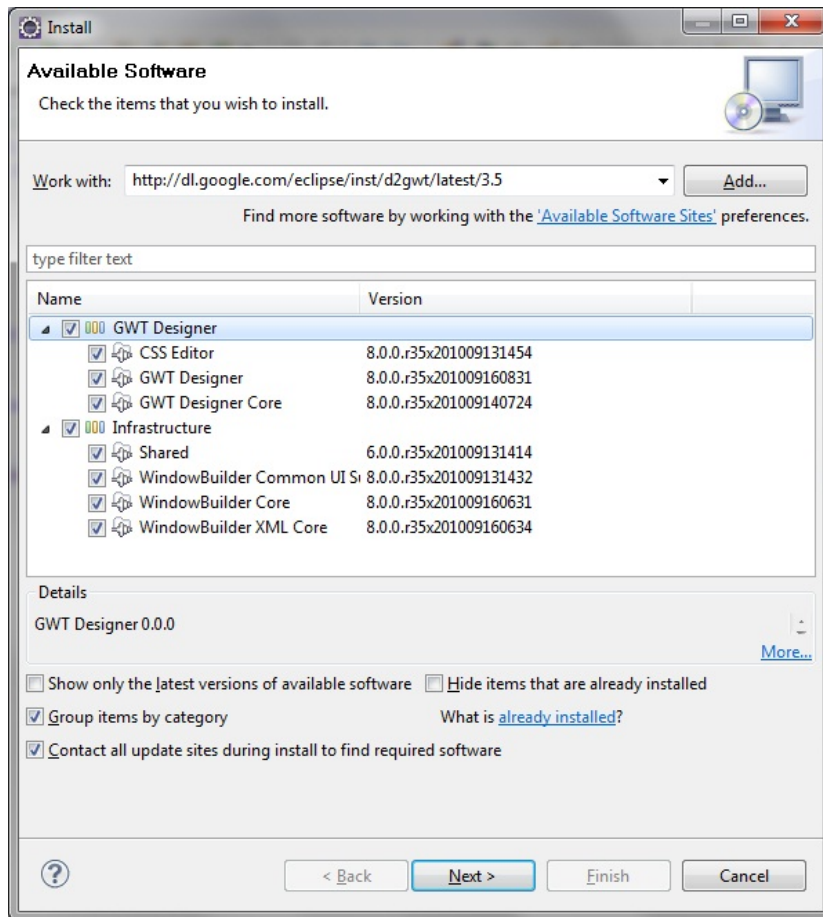


Figure x.x Once you add the update site for the GWT Designer it will present you with a list of software available for installation.

You will want to select the options for both the GWT Designer and Infrastructure, then click Next. Continue the process by following the instructions provided. Once complete it will ask you to restart Eclipse to finish the installation.

At this point you have installed everything required to write GWT applications, but there is one tool left that you will need in order to quickly test your application.

1.2.5 *Installing the Development Mode browser plug-in*

The Development Mode browser plug-in is tool that allows your web browser to communicate with a development mode server running on your workstation. We will show you how to launch development mode in chapter 2, so for now we only want to show you how to install the plug-in.

To install the plug-in for a target browser you can open up the following URL in your browser <http://gwt.google.com/missing-plugin/MissingPlugin.html>. This will look like figure x.x.

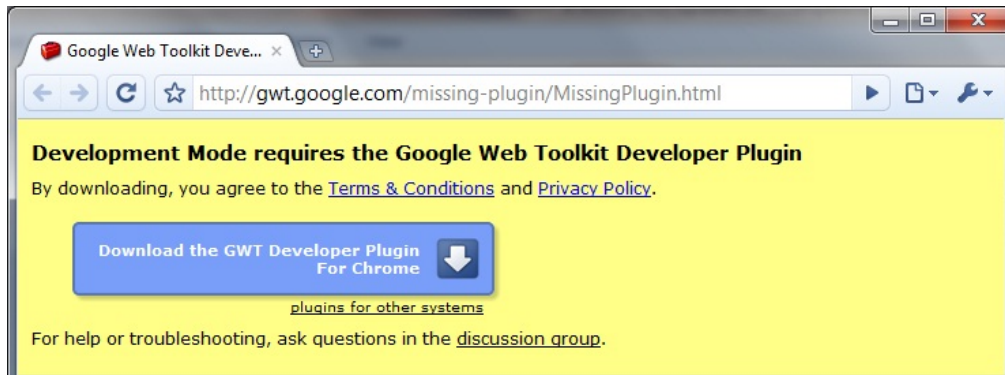


Figure x.x The dialog window for the development mode browser plug-in allows you to install the plug-in for your browser.

At the time of this writing this plug-in is available for most modern browsers. If the plug-in is not available for a specific browser you will be presented with a message indicating when you navigate to the plug-in installation URL.

Before moving on we should point out if you ever try to view your application that is running in development mode, you will be presented with this same installation dialog.

At this point your environment is complete. You have Eclipse, the GWT SDK, the Google Eclipse Plugin, and the Development Mode browser plug-in all installed. So let's recap everything we covered in this chapter and discuss where we go from here.

1.3 Summary

In this chapter we provided an overview of what GWT has to offer. From an optimizing compiler, to a rich widget set, to tools for dealing with popular data formats like JSON and XML, to communicating with the server and history management.

Tools are great, but the takeaway that we want you to get from this chapter is that GWT is more than just a tool for writing JavaScript application in Java. GWT is a platform with which you can build extraordinary complex applications that run in the browser without any proprietary plug-ins.

But this isn't anything new. The truth of the matter is that browsers have had support for Dynamic HTML and making remote calls for an entire decade, yet we see very few large browser-based applications outside of those coming from Microsoft, Google, or Yahoo!. The reason is that writing large applications in JavaScript is difficult and complex. In fact, this is the primary reason for the adoption of GWT.

Developers who write desktop applications have for a long time had great tool support. They have had the ability to design layout using a visual designer, support for working with databases, and structural tools like those that facilitate dependency injection. GWT might not provide all of this quite yet, but it is a major step forward for the web developer.

GWT is currently limited by what can be done by JavaScript in the browser, and if you haven't been watching, HTML 5 is adding loads of new tools to facilitate richer JavaScript applications. Many browsers now support the new canvas tag for bitmap drawing, some support built-in databases, some offline JavaScript application caching, and even dragging files right into the browser for file uploads. If you aren't a GMail user (or didn't notice), you can now drag files right into the browser in order to attach them to an email... as long as you are using a browser that supports that functionality.

At the time of this writing GWT provides very little support for these new HTML 5 features, but it will. And that is another feature of GWT, which is a dedicated development team that continually advances the product to match advances in the web. This makes GWT a good bet for the future. And given that GWT is also open source with the extremely liberal Apache 2 license, it an even better bet for the future.

We don't mean to sound like cheerleaders, but GWT excites us as web developers, and is why we wrote this book. RAH RAH GWT!

Now that we have a good understanding of what GWT provides and a development environment that we can use to develop GWT applications, we can now proceed to building GWT applications. In chapter 2 you will immediately put your development environment to work by creating a GWT application and walking through what a GWT project looks like.