

Accelerated GWT

Building Enterprise Google
Web Toolkit Applications



Vipul Gupta

Accelerated GWT: Building Enterprise Google Web Toolkit Applications

Copyright © 2008 by Vipul Gupta

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-59059-975-4

ISBN-10 (pbk): 1-59059-975-6

ISBN-13 (electronic): 978-1-4302-0616-3

ISBN-10 (electronic): 1-4302-0616-0

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editor: Clay Andres

Technical Reviewer: Eric Briley

Editorial Board: Clay Andres, Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell,

Jonathan Gennick, Matthew Moodie, Joseph Ottinger, Jeffrey Pepper, Frank Pohlmann,

Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh

Senior Project Manager: Tracy Brown Collins

Copy Editor: Kim Wimpsett

Associate Production Director: Kari Brooks-Copony

Production Editor: Ellie Fountain

Compositor: Molly Sharp

Proofreader: Liz Welch

Indexer: Beth Palmer

Artist: Kinetic Publishing Services, LLC

Cover Designer: Kurt Krames

Manufacturing Director: Tom Deboliski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit <http://www.springeronline.com>.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit <http://www.apress.com>.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales—eBook Licensing web page at <http://www.apress.com/info/bulksales>.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at <http://www.apress.com>.

To my parents, for always being supportive and loving

Contents at a Glance

About the Author	xv
Acknowledgments	xvii
Introduction	xix

PART 1 ■ ■ ■ Getting Started with GWT

■ CHAPTER 1	GWT Basics and a First Application	3
■ CHAPTER 2	GWT Architecture and Internal Features	27

PART 2 ■ ■ ■ UI Programming and Client-Server Communication

■ CHAPTER 3	UI Programming: Basic Widgets	59
■ CHAPTER 4	Communication: RPC	89
■ CHAPTER 5	UI Programming: Handling Events and Using Advanced Widgets ...	105
■ CHAPTER 6	Communication: Advanced Techniques	135

PART 3 ■ ■ ■ Making Applications Ready for the Real World

■ CHAPTER 7	Testing GWT Applications	171
■ CHAPTER 8	Internationalizing Your Applications: A Modern-Day Reality ...	201
■ CHAPTER 9	Some Important, Not-to-Be-Missed Techniques	233
■ CHAPTER 10	Peeking Into the Upcoming GWT 1.5 Release	265
■ INDEX		283

Contents

About the Author	xv
Acknowledgments	xvii
Introduction	xix

PART 1 ■ ■ ■ Getting Started with GWT

■ CHAPTER 1	GWT Basics and a First Application	3
	Setting Up Your GWT Environment	4
	Hosted Mode vs. Web Mode	4
	Web Mode	4
	Hosted Mode	5
	What Are All Those GWT Files For?	6
	Creating Your First GWT Application	6
	Tools for Creating a Project	6
	Running the Application Using Generated Scripts	9
	Working with Modules in GWT	11
	Structure of a Module File	11
	Creating the Host HTML File	15
	Steps to Create a GWT Application	16
	Creating Another Application Step-by-Step	17
	Creating the Basic Project Structure	17
	Adding the Module File	17
	Creating the Entry-Point Class	18
	Creating the Host HTML File	21
	Running the Application in Hosted Mode	22
	Summary	25
■ CHAPTER 2	GWT Architecture and Internal Features	27
	Understanding the Components That Make Up the GWT Framework ...	27
	Development Tools Explained	27
	Class Libraries Explained	28
	What Version of the Java Language Does the GWT Support?	31

The Same Origin Policy and Its Implications on GWT	33
Same Origin Policy Explained	33
What Are the Implications of the Same Origin Policy on GWT? ...	34
Deferred Binding	34
Understanding Generator, Related Classes, and Code Generation	
Using Generators	36
Example of Using Generator to Autogenerate Code for Your	
Applications	39
Building the Generator-Based Application	40
GWT: Startup/Bootstrap Process	52
Summary	55

PART 2 ■ ■ ■ UI Programming and Client-Server Communication

■ CHAPTER 3	UI Programming: Basic Widgets	59
	GUI Building with Fundamental Widgets	59
	Hierarchy of Base Classes Explained	60
	How Do You Use the Widgets Provided by GWT?	65
	Understanding Layouts Using Panels	70
	Starting with a RootPanel	72
	Aligning Widgets Using a CellPanel	72
	What Is an HTMLTable Panel?	76
	What Is a FlowPanel?	78
	Creating Complex Widgets Using Composites	79
	Developing a Sample Application Using Composites	80
	Summary	86
■ CHAPTER 4	Communication: RPC	89
	Understanding RPC	89
	How to Use RPC in GWT	90
	Creating Service Interface (Also Called the Synchronous Interface) ...	90
	Creating the Asynchronous Interface	91
	Understanding the AsyncCallback Interface	91
	Making an Actual Remote Procedure Call	92
	Your First Complete RPC Example	93
	RPC in GWT: Behind the Scenes	100
	Summary	103

CHAPTER 5	UI Programming: Handling Events and Using Advanced Widgets	105
	Handling Events Generated by Widgets	105
	Handling Events Using Listeners	107
	Handling Events Using Adapter Classes	108
	Styling Applications Using CSS	110
	How Do Nested CSS Classes Apply to GWT Widgets?	111
	Including Style Sheets in Your GWT Application	112
	Using the TabBar Widget	114
	Using the TabPanel Widget	116
	Optimizing Applications Using ImageBundle	119
	Understanding AbstractImagePrototype	120
	Sample Application Demonstrating the Use of ImageBundle	121
	How Does an ImageBundle Work?	129
	Building Classic HTML Forms Using FormPanel	130
	The HasName Interface (in the com.google.gwt.user.client.ui Package)	131
	Sample Application Demonstrating the Use of FormPanel	131
	Summary	134
CHAPTER 6	Communication: Advanced Techniques	135
	What Is Serialization?	135
	Examining the Different Type of Objects That Can Be Used in Communication	136
	Making User-Defined Serializable Classes	136
	Designing an RPC Application	136
	Creating the Domain Objects Used for Communication	138
	Handling UI Events	139
	Defining the Service and Asynchronous Interfaces	140
	Creating the Callback Classes	141
	Creating the Controller Class	142
	Writing the Server-Side Implementation	144
	Mapping the Server-Side Path in the Module's XML File	144
	Running the Application	145
	Serializing Collection Classes	146
	Understanding Collection Classes Using a Comprehensive Example	147
	Creating the Entry-Point Class	151
	Example of Using HashMap	161

Creating Custom Field Serializers	162
Understanding the <code>SerializationStreamWriter</code> Interface	162
Understanding the <code>SerializationStreamReader</code> Interface	163
Communicating with Server Using HTTP Requests	166
Summary	168

PART 3 ■ ■ ■ Making Applications Ready for the Real World

■ CHAPTER 7 Testing GWT Applications	171
Understanding the <code>junitCreator</code> Utility	172
Creating Sample Tests Using the <code>junitCreator</code> Utility	172
Writing GWT-Based Unit Tests	174
Examining the <code>GWTTestCase</code> Class	175
Creating Tests Without Using the <code>junitCreator</code> Utility	178
Running the Tests	179
Points to Remember While Creating a JUnit-Based GWT Test Case	184
Testing Asynchronous Calls	185
Testing by Using a Timer Object	185
Testing by Using a Mock Callback Object	187
Using the Benchmark Utility and Writing Tests for Gathering Benchmark Results	189
Examining the Benchmark Class	190
Examining the <code>IntRange</code> Class	191
Sample Application for Benchmarking	192
Summary	199
■ CHAPTER 8 Internationalizing Your Applications: A Modern-Day Reality	201
What Is Internationalization?	201
Localization	202
Things to Remember While Developing an Internationalized Application	202
Understanding Character Encodings	203
Important Character Sets	203
Character Encoding and Web Applications	204

GWT's Internationalization Support	204
The I18N Module	204
Internationalization Techniques Available in GWT	205
Static String Internationalization	205
Dynamic String Internationalization	206
Exploring the Localizable Interface	207
Understanding Locale-Specific Substitution of Values	207
Setting and Using the Locale Value in Your Application	208
Tool for Internationalizing Your GWT Applications	209
Understanding the I18nCreator Tool	209
Creating Your First Internationalized Application	209
Creating the Project Structure	210
Working with the Properties Files	211
Generating the Interface Corresponding to Property Files	212
Adding Locale Values to a Module's XML File	213
Setting Up the Host HTML File	213
Setting the Module's Entry-Point Class	214
Running the Application	215
Exploring the Constants Interface	218
Example of Using the Constants Interface	218
Format of Methods in the Interface Extending the Constants Interface	220
Creating an Entry-Point Class to Access the Interface	221
Exploring the Messages Interface	222
Format of Methods in the Interface Extending the Messages Interface	222
Creating a Custom Interface	222
Working with the Properties Files	223
Creating an Entry-Point Class to Access the Interface	224
Running the Application	225
Creating the Messages Interface Using i18nCreator	225
Exploring the ConstantsWithLookup Interface	226
Embedding Data into Your Host HTML Pages	228
Examining and Using the Dictionary Class	229
Localizing Dates, Times, Numbers, and Currencies	230
Examining the NumberFormat Class	230
Examining the DateTimeFormat Class	231
Summary	232

CHAPTER 9	Some Important, Not-to-Be-Missed Techniques	233
	Understanding the History Mechanism	234
	History Class	235
	Steps to Add History Support	235
	Adding History Support to an Application	236
	The Hyperlink Widget and Its Integration with the History Mechanism	244
	Deploying a GWT-Based Application	245
	Default Directory Structure of a Web Application	245
	Steps for Deploying a GWT Application on a Web Server	246
	Maintaining Server Sessions with a GWT Application	249
	Modifying the RemoteService and Its Asynchronous Version	250
	Introducing the Callback Class Corresponding to the New Service Method	251
	Adding the Server-Side Implementation for the New Service Method	252
	Modifying the Util Class to Support the New Method	253
	Tweaking the Domain Object for Use	254
	Running the Application	254
	Creating an ImageBundle of Bundles	256
	Understanding and Using the <super-source> Tag	258
	Packaging a GWT Module for Reuse	259
	Steps to Package an Application as a Reusable Module	260
	Using the New Module in a Sample Application	261
	Summary	262
CHAPTER 10	Peeking Into the Upcoming GWT 1.5 Release	265
	Understanding the Major Changes in Version 1.5	265
	Setting Up Your Environment for Using Version 1.5	266
	Testing the New Release	267
	Using Version 1.5 of the GWT Framework	269
	Setting Up and Running the BookStore Example on the New Version of the Library	269
	Type-Safe Collections by Using Generics	271
	The Type-Safe AsyncCallback Object	272
	Setting Up the LoanServicingSystem Example on the New Library	274
	New Package for the Benchmark Classes	274

The New Annotations for Benchmark Tests	274
Setting Up the AdvancedWidgets (ImageGallery) Example on the New Library	278
The New Resource Annotation for ImageBundle	278
Exploring the Output Structure of Compilation with the New Release	280
Summary	282
 ■ INDEX	 283

About the Author



■ **VIPUL GUPTA** is a software engineer who designs and develops complex web-based applications and distributed software systems. His professional experience includes implementing a virtual file system for a web-based application and developing highly scalable back ends and extremely responsive web-based UIs for high-traffic websites. He has also worked on the prototype of a complex workflow-based solution that is used to handle various business processes. He is an expert in developing enterprise-level applications in the financial domain and has expertise in using a wide range of open source and commercial tools and technologies. Apart from his passion for computer science, he has numerous other interests including Formula 1, table tennis, and water sports.

Acknowledgments

This book would not have been possible without the help of a large number of people, so I would like to sincerely thank everyone involved. Thanks in particular to the team at Apress, especially Tracy Brown Collins and Clay Andres, for helping me keep the book on track and taking care of all the details that go into getting a book printed and on the shelves. I would also like to thank Jason Gilmore for his support during the initial part of the book and Kim Wimpsett for finding and correcting the many mistakes I made during the writing process. I would also like to thank Ellie Fountain and Tina Nielsen for their help in the production and administrative processes during the writing of this book.

Special thanks goes to Chris Mills for getting me excited about the prospect of writing this book and to Eric Briley, my technical reviewer, whose comments helped me polish some of the rough edges of the book.

I would also like to thank my family for their love and support throughout, especially my wonderful wife, Ria, for her patience and enthusiasm during the long hours of writing.

Introduction

Among other things, one of the biggest problems faced by a web application developer is the task of making an application compatible with different browsers. So, how does the idea of writing web applications in Java and testing and debugging them right in your favorite Java IDE sound? Exciting, doesn't it? This is what the Google Web Toolkit (GWT) framework lets you achieve. The GWT framework allows you to write and test your web applications in Java and compile the Java code into JavaScript for deployment purposes. Developing in a mature object-oriented language like Java brings with it all the benefits of object-oriented programming like modular design, type safety, and so on, which are essential for any project of even a moderate size.

I was originally skeptical about the idea of a framework supporting various browsers by converting the Java code into JavaScript and also about the quality of JavaScript code created by it. But once I started using the GWT framework and observed the quality of the generated JavaScript, I became convinced that GWT is going to become the de facto standard to write web-based applications in the future.

During the course of this book, you will learn how to use GWT to build high-quality web-based applications that will run across multiple browsers without the tweaks needed to achieve the same while using JavaScript directly. My aim is to provide you with all the knowledge you need to use GWT effectively in your own applications and to give you insight into what is happening behind the scenes in GWT.

Specifically, you will do the following in this book:

- Learn the fundamentals of using GWT to build UIs for your web applications.
- Learn how the framework works internally so you can solve most common programming problems in web-application development.
- Become aware of remote procedure calls and the asynchronous callback mechanism.
- Effectively write test cases for testing the different parts of your applications, including the asynchronous part of server-side communication.
- Learn how to write benchmark tests for your applications.
- Optimize your web applications by using techniques to bundle multiple images into a single image.
- Learn how to write applications with internationalization in mind.
- Design and implement reusable modules.

- Speed up your web application development by testing your applications right from your favorite Java IDE or from the command line, without the need for deploying them on a web server. You will also learn how to compile and convert the Java code into JavaScript and deploy your applications on a web server.
- Learn how GWT solves the problem of making an application compatible with different browsers without additional coding or development effort.

After reading this book, you will be equipped with all the knowledge you need to build applications using GWT.

Who This Book Is For

This book is for Java-minded web developers seeking to incorporate Ajax capabilities into their web applications without sacrificing sound development principles.

Downloading the Code

The code used in the book's examples will be available in Zip file format in the Downloads section of the Apress website (<http://www.apress.com>). The instructions for setting up and running the examples will be available in a Readme.txt file, which is bundled along with the source code. The software programs that are used in the book include Eclipse IDE (<http://www.eclipse.org>), the Tomcat web server (<http://tomcat.apache.org>) and of course the GWT framework library (<http://code.google.com/webtoolkit/download.html>).

Contacting the Author

The author can be contacted at vipulgupta.vg@gmail.com.

PART 1



Getting Started with GWT

The Google Web Toolkit, better known as GWT, is a Java-based open source framework that helps you develop Ajax-based web applications without having to worry about the quirky cross-browser details. Released in 2006 and with millions of downloads so far, GWT is changing the face of Ajax-based web application development with faster turnaround times in development and testing; it also offers application debugging right in your favorite Java-based IDE.

Chapter 1 will help get you started with GWT. You will learn some basic GWT terminology and understand how to download and set up GWT on your computer. The chapter will also explain details about various libraries in the framework along with the different modes in which the application can be run. This chapter will also guide you through the development and operation of some basic GWT applications with the tools provided in the framework as well as without them.

Chapter 2 will go into details about how the GWT framework works, providing you with a complete picture of GWT's capabilities and its libraries. It will discuss the concept of deferred binding and give you details about autogenerating code for your applications. It will also explain the bootstrap/startup process followed by a GWT application and discuss the various files created by the GWT compiler.



GWT Basics and a First Application

GWT is an open source framework that allows you to develop Ajax-based applications in the Java language and provides tools to convert the Java code into JavaScript and HTML. This frees you, the developer, from the burden of rewriting your JavaScript to suit the peculiarities and lack of standards support in all the various browsers in use on people's computers.

Since its June 2006 release, the GWT framework has made a tremendous impact on the developer community engaged in developing Ajax-based web applications. This chapter will help you get started with GWT and help you understand how GWT fits into the next Ajax-based application you develop.

Note GWT currently supports Internet Explorer 6 and 7; Firefox 1.0, 1.5, and 2.0; Mozilla; Safari 2.0; and Opera 9.0. The GWT compiler converts the Java classes into separate script files in compliance with the JavaScript understood by the various JavaScript engines found in the underlying web browsers. What this means for you as a developer is that Java's original promise of "write once, run anywhere" now applies to the world of Ajax-based applications. This allows you to focus more on the internal domain and application logic, rather than spending precious time making the application work across multiple browsers.

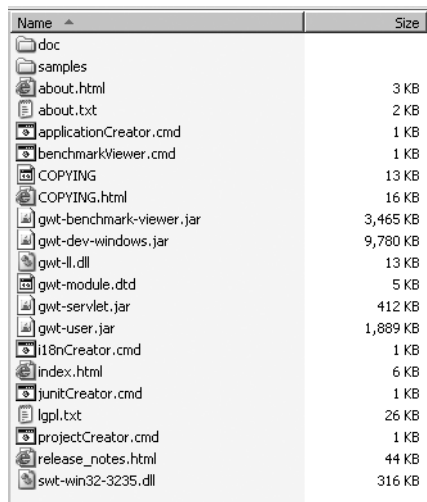
Debugging has long been a major problem for developers tasked with writing JavaScript. Although over the past few years a few very good JavaScript debugging tools have become available, most of them, such as Firebug, are designed to integrate closely with web browsers rather than with the modern IDEs used for Java development, such as Eclipse. GWT solves this problem by providing a mechanism to directly run and test the application from within the IDE as you'd do with typical Java code.

This chapter will start with the details of downloading and setting up GWT on your machine. Then it will go step by step through developing and running a sample application, using the tools and utilities provided by the GWT framework. The chapter will then dissect an entire sample application and explore its various parts to solidify your understanding of the different components of a GWT application. Finally, the chapter will close by showing how to write another application on your own without using the tools provided by GWT so that you have a clear understanding of the process involved in developing an application using GWT.

Setting Up Your GWT Environment

As of this writing, the current version of GWT is 1.4.61. You can download it from <http://code.google.com/webtoolkit/download.html>.

I downloaded the file named `gwt-windows-1.4.61.zip`. Then I extracted the `.zip` file to the root directory (`C:\`) of my system. In my case, I unzipped GWT at the location `C:\gwt-windows-1.4.61`. I renamed the folder to `C:\gwt` and will reference this name across all the code in this book and in the source code for the samples of this book. Figure 1-1 shows the contents of this directory.



Name	Size
doc	
samples	
about.html	3 KB
about.txt	2 KB
applicationCreator.cmd	1 KB
benchmarkViewer.cmd	1 KB
COPYING	13 KB
COPYING.html	16 KB
gwt-benchmark-viewer.jar	3,465 KB
gwt-dev-windows.jar	9,780 KB
gwt-ll.dll	13 KB
gwt-module.dtd	5 KB
gwt-servlet.jar	412 KB
gwt-user.jar	1,889 KB
i18nCreator.cmd	1 KB
index.html	6 KB
junitCreator.cmd	1 KB
lgpl.txt	26 KB
projectCreator.cmd	1 KB
release_notes.html	44 KB
swt-win32-3235.dll	316 KB

Figure 1-1. List of files after extracting the GWT package

Hosted Mode vs. Web Mode

Before discussing the files shown in Figure 1-1, it's important to discuss the various modes, web and hosted, in which an application can be run using GWT.

Web Mode

Traditionally, developers of web-based applications had to go through the complete cycle of building, packaging, and deploying the application on a web server to test each new feature that was implemented. This slow and time-consuming process led to overly long development times and project delays.

In *web mode*, the application is run as pure JavaScript and HTML. These files are the result of compiling Java source code of your GWT-based modules by using the Java-to-JavaScript compiler. The JavaScript and HTML files obtained by the compilation step are used for the actual deployment to production environments.

Hosted Mode

In addition to traditional web mode, GWT provides another approach for testing and running applications known as *hosted mode*. In this mode, the actual Java byte code of the classes is run within the Java Virtual Machine (JVM). While web mode requires you to deploy your application to a web server, hosted mode allows a developer to run the application right off their favorite IDE in a web browser embedded in the framework. This greatly reduces the amount of time required to verify, test, and debug changes in an application.

A major advantage of Java byte code being run in hosted mode is the ability to debug the application, in the Java language, by using powerful IDEs available for the Java language. The debugging capabilities of a modern Java IDE are far more mature than the evolving JavaScript-related tools.

Figure 1-2 shows a GWT application being debugged in the Eclipse IDE.

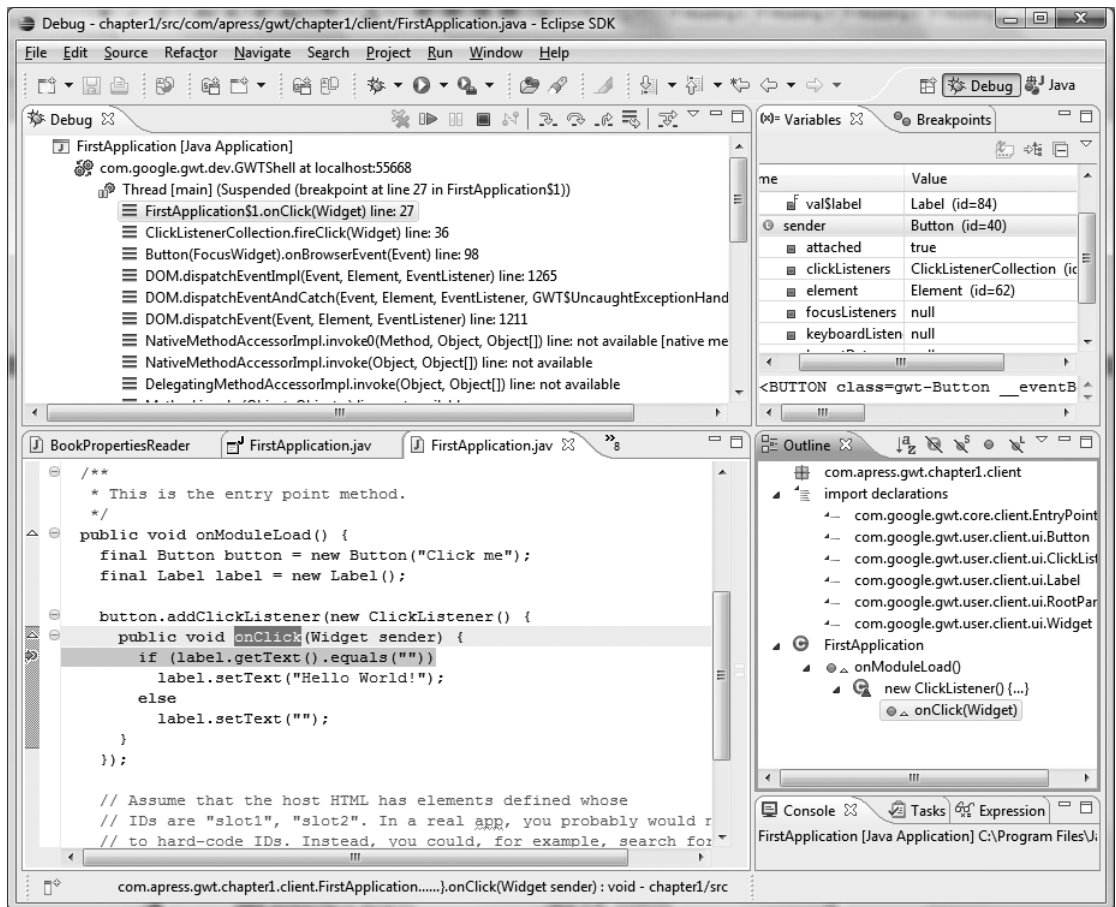


Figure 1-2. Debugging a GWT application directly in Eclipse while running the application in hosted mode

What Are All Those GWT Files For?

Here are details about some of the particularly important files shown in Figure 1-1:

`gwt-servlet.jar`: This provides all the GWT classes (including all the required RPC-related classes) that your application will need. This JAR file should be included in the application when running the application in web mode.

`gwt-user.jar`: This JAR file is needed when the application is run in hosted mode, that is, inside the web server bundled inside the GWT libraries. (This file should never be included as part of the application when deploying the application to an actual production environment.)

`gwt-dev-windows.jar`: This JAR file is needed if you want to run your application in hosted mode. Rather than a traditional web application development cycle of building, deploying, and testing the application in a web browser, GWT provides an internal hosted mode that you can use to test and debug the application by running the application in an embedded browser window. When the application is run in hosted mode, the startup class of your application should be `com.google.gwt.dev.GWTShell`.

`gwt-benchmark-viewer.jar`: This JAR file contains the benchmarking classes that you can use to create benchmarking tests for your applications (GWT has a `Benchmark` class that can be subclassed and used for common benchmarking needs). You can use the `benchmarkViewer` tool (`benchmarkViewer.cmd`, located in the root of the GWT directory) to derive charts/reports from the XML file that is generated when running the benchmarking tests. You'll learn more about benchmark testing in Chapter 7.

Table 1-1 lists the various JAR files, which should be included in your application when running it in the different modes supported by GWT.

Table 1-1. *JAR Files to Be Included for Web/Hosted Mode*

Application Mode	JARs to Be Included
Web mode	<code>gwt-servlet.jar</code>
Hosted mode	<code>gwt-user.jar</code> and <code>gwt-dev-windows.jar</code> (or <code>gwt-dev-linux.jar</code>)

Creating Your First GWT Application

The following sections will go over the steps needed to create an application using GWT. You will also develop your first application as you go through these steps.

Tools for Creating a Project

The developers of the GWT framework, by providing a number of easy-to-use utilities, have done their part in making starting a project with GWT as easy as possible. The following sections will cover the utilities you'll use to build your first application using GWT in this chapter.

projectCreator

This utility creates the project structure and an empty Eclipse project or Ant build file (or both) for your application. The command takes the following flags/parameters:

```
projectCreator [-ant projectName] [-eclipse projectName] [-out dir]
               [-overwrite] [-ignore]
```

Specifically, the flags/parameters are as follows:

- -ant generates a project build file to compile the source of the application. (The build file is named <projectName>.ant.xml.)
- -eclipse generates an Eclipse project (.project and .classpath files).
- -out is the directory where the output files will be written (default is the current directory).
- -overwrite overwrites any existing files.
- -ignore ignores any existing files (does not overwrite them).

Note You have to specify either the -ant or -eclipse flag to use this command.

For example, if you want to create a new project named chapter1 in a directory named chapter1, then you should run `projectCreator -eclipse chapter1` while inside the chapter1 directory.

Listing 1-1 shows the result of executing this command.

Listing 1-1. Project Structure and Corresponding Files Created by Running projectCreator

```
C:\gwt\chapter1>projectCreator -eclipse chapter1
Created directory C:\gwt\chapter1\src
Created directory C:\gwt\chapter1\test
Created file C:\gwt\chapter1\.project
Created file C:\gwt\chapter1\.classpath
```

Executing the command shown previously creates the standard project structure in the chapter1 directory (namely, src and test directories for Java source and tests, respectively) and the .project and .classpath files for the Eclipse project for the application.

As another example, say instead that you want to create the project under a subdirectory mysub of your current directory. Then you should use the -out flag as mentioned earlier and pass the name of the subdirectory where the project should be created. Listing 1-2 shows the output of using this flag in the example.

Listing 1-2. *Project Structure and Ant Build File Created in a Specified Folder by Running projectCreator with the -ant and -out Flags*

```
C:\gwt>projectCreator.cmd -ant MySubProject -out mysub
Created directory mysub\src
Created directory mysub\test
Created file mysub\MySubProject.ant.xml
```

Note You should enter the gwt directory to the path of your system so the shell can recognize the utilities when they are called from outside the main gwt directory. In Windows, this would mean adding C:\gwt to your system's path variable.

applicationCreator

This utility helps create a basic starter application for a GWT-based Ajax application. By using the -eclipse flag with the utility, you can also create a launch configuration for debugging the application in Eclipse. The command takes the following flags/parameters:

```
applicationCreator [-eclipse projectName] [-out dir] [-overwrite]
                  [-ignore] className
```

Specifically, the flags/parameters are as follows:

- -eclipse creates a launch configuration for debugging the application in Eclipse.
- -out is the directory where the output files will be written (default is the current directory).
- -overwrite overwrites any existing files.
- -ignore ignores any existing files (does not overwrite them).
- className is the fully qualified name of the entry-point class in the application.

If you want to create the sample project in the chapter1 directory with the base GWT package location as com.apress.gwt.chapter1, then you should run applicationCreator -eclipse chapter1 com.apress.gwt.chapter1.client.FirstApplication from the shell while in the chapter1 directory. Listing 1-3 demonstrates this scenario.

Listing 1-3. *Sample Project Including Launch Configuration Files Created by Running the applicationCreator Command with the -eclipse Flag*

```
C:\gwt\chapter1>applicationCreator -eclipse chapter1 ➡
                  com.apress.gwt.chapter1.client.FirstApplication
Created directory C:\gwt\chapter1\src\com\apress\gwt\chapter1
Created directory C:\gwt\chapter1\src\com\apress\gwt\chapter1\client
```

```
Created directory C:\gwt\chapter1\src\com\apress\gwt\chapter1\public
Created file C:\gwt\chapter1\src\com\apress\gwt\chapter1\FirstApplication.gwt.xml
Created file ➡
    C:\gwt\chapter1\src\com\apress\gwt\chapter1\public\FirstApplication.html
Created file ➡
    C:\gwt\chapter1\src\com\apress\gwt\chapter1\client\FirstApplication.java
Created file C:\gwt\chapter1\FirstApplication.launch
Created file C:\gwt\chapter1\FirstApplication-shell.cmd
Created file C:\gwt\chapter1\FirstApplication-compile.cmd
```

junitCreator

This utility creates a JUnit test and scripts that you can use for testing the various components of the application in both hosted and web modes. You will learn more about this utility and the details of testing GWT-based applications in Chapter 7.

i18nCreator

This utility creates scripts and a property file for internationalizing your applications. (You'll learn more about internationalization and this utility in Chapter 8.)

Running the Application Using Generated Scripts

You can execute the application in hosted mode by using `FirstApplication-shell.cmd`. With your current directory being your application's home directory, execute the command `<applicationName>-shell.cmd` on the command prompt to execute the application. You will see the hosted browser being loaded and the application running. Listing 1-4 shows the snippet with the script being used to run the application.

Listing 1-4. Command to Run Your First Application in Hosted Mode

```
C:\gwt\chapter1> FirstApplication-shell.cmd
```

Note You should ensure that all the Java files are compiled before running the application using generated scripts such as `FirstApplication-shell.cmd` mentioned previously. You can enable autocompile in the Eclipse IDE by select the Project ► Build Automatically option if it is not enabled already. This entire book will assume that this option is enabled and the project is fully built before running the application.

Figure 1-3 and Figure 1-4 show the result of running the application in hosted mode by running the script in Listing 1-4.

Specifically, Figure 1-3 shows the web server window. This window displays the application log and stack trace, if any, for the application.

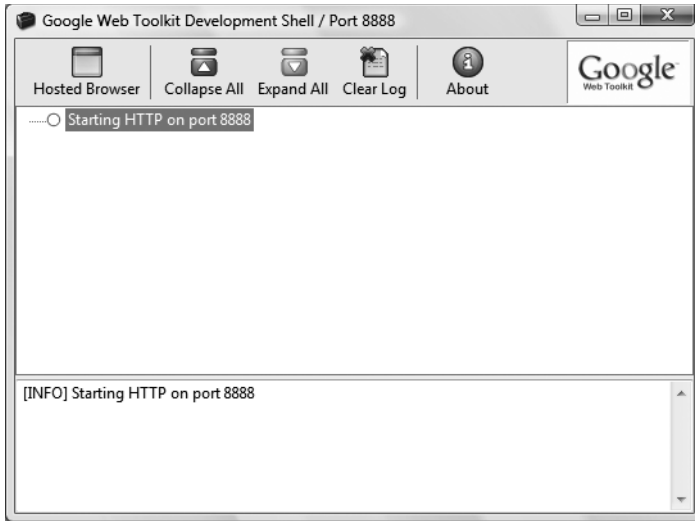


Figure 1-3. Running your sample application in hosted mode. The embedded web server window shows the application log and stack trace in case of any errors.

Figure 1-4 shows the actual web browser window with the application running in it.

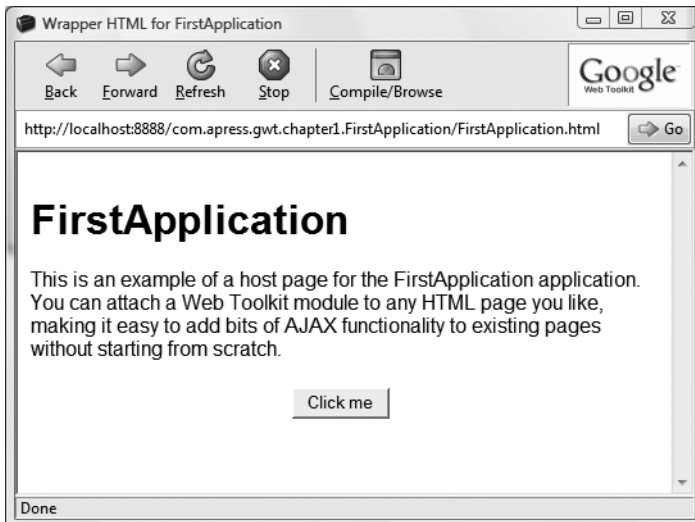


Figure 1-4. Running your sample application in hosted mode. The embedded web browser window shows the application running in it.

Understanding the Generated Scripts to Compile and Run the Application

The `applicationCreator` utility creates two script files for helping the development process. These files contain the Java commands to compile the application and to run the application in hosted mode (running the entry-point class). The file contents for the sample application are as follows:

`FirstApplication-shell.cmd`: This file loads the start page of the GWT module in the hosted browser and contains the following Java command:

```
@java -cp "%~dp0\src;%~dp0\bin;C:/gwt/gwt-user.jar;C:/gwt/gwt-dev-windows.jar"
    com.google.gwt.dev.GWTShell -out "%~dp0\www" %*
    com.apress.gwt.chapter1.FirstApplication/FirstApplication.html
```

`FirstApplication-compile.cmd`: This file is responsible for compiling the GWT module and outputs the resultant files in the folder named `www`. It contains the following Java command:

```
@java -cp "%~dp0\src;%~dp0\bin;C:/gwt/gwt-user.jar;C:/gwt/gwt-dev-windows.jar"
    com.google.gwt.dev.GWTCompiler -out "%~dp0\www" %*
    com.apress.gwt.chapter1.FirstApplication
```

Working with Modules in GWT

GWT applications are basically structured in terms of modules. Each module defines a well-defined functionality or component, and these modules are combined to make a full-fledged application.

Structure of a Module File

The recommended (default) structure of a GWT-based module (with the corresponding structure as an example from your first application) is as follows:

- `moduleName.gwt.xml`: This file includes the configuration of the module. For example:
`com/apress/gwt/chapter1/ FirstApplication.gwt.xml`
- `client package`: A client subpackage (and corresponding subpackages) with client-side code. For example:
`com/apress/gwt/chapter1/client`
`com/apress/gwt/chapter1/client/data ... and so on`
- `public package`: A public subpackage (and corresponding subpackages) with static web content. For example:
`com/apress/gwt/chapter1/public`
- `server package`: A server subpackage (and corresponding subpackages) with server-side code. For example:
`com/apress/gwt/chapter1/server`
`com/apress/gwt/chapter1/server/data ... and so on`

Note Even though using the listed module structure is just a recommendation, it's advisable that you stick to the previous structure because the entire community in general is sticking to this, and your modules will therefore be more easily understandable by others.

A *module* in GWT is simply an XML file and combines the entire configuration that your application would need in a single file. GWT modules define the entry point into the application. In the example discussed in this chapter, the module file is located at `chapter1/src/com/apress/gwt/chapter1` and is named `FirstApplication.gwt.xml`. Listing 1-5 shows the contents of the module file in the sample application.

Listing 1-5. *Contents of the Module File (FirstApplication.gwt.xml) for the Sample Application*

```
<module>
  <!-- Inherit the core Web Toolkit stuff. -->
  <inherits name='com.google.gwt.user.User' />

  <!--Specify the app entry-point class. -->
  <entry-point class='com.apress.gwt.chapter1.client.FirstApplication' />
</module>
```

Understanding Different Tags in a Module File

You can use the module file to define a number of properties related to the application:

- The entry-point class property
- The inherits property
- The source and public properties
- The servlet property
- The stylesheet property
- The script property
- The extend-property property

These are described in more detail in the following sections.

The entry-point class Property

When a module is loaded, the entry-point classes defined in the module's configuration file are instantiated, and the corresponding `onModuleLoad()` methods of these entry-point classes are called. This includes the entry-point classes of the inherited modules as well.

In the GWT library, `EntryPoint` is basically defined as an interface. This interface declares the `onModuleLoad()` method, which, as defined earlier, is automatically called when the module is loaded. (Adding an entry-point class is optional. However, you can add any number of entry-point classes in your application by including them in the configuration file of your application's module.)

Here's an example:

```
<entry-point class='com.apress.gwt.chapter1.client.FirstApplication' />
```

As you can see in the application's module file (`FirstApplication.gwt.xml`), the entry-point class for the application is defined as `com.apress.gwt.chapter1.client.FirstApplication`. Listing 1-6 shows the code for the entry-point class.

Listing 1-6. *Contents of the Entry-Point Class for the Sample Application*

```
package com.apress.gwt.chapter1.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.ClickListener;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.Widget;

/**
 * Entry-point classes define <code>onModuleLoad()</code>.
 */
public class FirstApplication implements EntryPoint {

    /**
     * This is the entry-point method.
     */
    public void onModuleLoad() {
        final Button button = new Button("Click me");
        final Label label = new Label();

        button.addClickListener(new ClickListener() {
            public void onClick(Widget sender) {
                if (label.getText().equals(""))
                    label.setText("Hello World!");
                else
                    label.setText("");
            }
        });

        // Assume that the host HTML has elements defined whose
        // IDs are "slot1" and "slot2". In a real app, you probably would
        // not want to hard-code IDs. Instead, you could, for example,
        // search for all elements with a particular CSS class and
        // replace them with widgets.
        RootPanel.get("slot1").add(button);
        RootPanel.get("slot2").add(label);
    }
}
```

The inherits Property

Inheriting a module is like including the entire configuration and code of that module in your application. This helps to reuse the existing functionality of other modules in your application. The configuration file has an `inherits` element that is used to inherit other modules in your application's module. You can inherit any number of modules by using this element.

Here's an example:

```
<inherits name='com.google.gwt.user.User' />
```

The source and public Properties

These two properties help declare the location of the source folder for the client code and for the public path. These two locations are relative to the path where the module XML file containing these attributes is found. If either of these attributes is missing, then default values for these properties are assumed.

For example, `<source path="myModuleClient"/>` defines that the `myModuleClient` directory and all its subpackages contain the client code, and all Java source files found in these locations adhere to the rules of GWT related to client-side code.

If the source element is missing, then `<source path="client">` is assumed as the default definition.

`<public path="myPublic"/>` defines that the `myPublic` directory acts as the root to the public path, and all files found in this folder and all its subfolders are directly accessible by external clients. The files found here are copied to the output directory during the translation steps followed by the GWT compiler.

If the public element is missing, then `<public path="public">` is assumed as the default definition.

If the module containing the previous property is located in the `com/apress/gwt/chapter1` folder and the module defines the public property as `<public path="myPublic"/>`, then the public folder for the application is located in the `com/apress/gwt/chapter1/myPublic` folder.

Note The `public` element allows filtering using patterns. This allows you to decide which resources to copy to the output directory while compiling the GWT module.

The servlet Property

This property allows you to test remote procedure calls (RPCs) by loading the servlets mapped to a specific path. (You are allowed to map all the servlets needed in your application by using this property.)

Here's an example:

```
<servlet path="url-path" class="classname"/>
```

The `url-path` value of the `path` attribute defines the location where the corresponding servlet for the application is mapped. The client should use the same mapping when registering

the call using the `ServiceDefTarget.setServiceEntryPoint(String)` method. (This will be explained in detail in Chapter 4.)

The stylesheet Property

This property allows you to add a CSS file to the module.

Here's an example:

```
<stylesheet src="css/Mycss.css"/>
```

The script Property

This property allows you to add external JavaScript code/libraries to the module.

Here's an example:

```
<script src="js/Mylibrary.js"/>
```

The extend-property Property

This property allows you to add values to a client property. Any number of such values may be introduced by adding them to the comma-separated list of values.

Here's an example:

```
<extend-property name="client-property-name" values="comma-separated-values"/>
```

This feature is most used in specifying locales for your application during internationalization. You will learn more about internationalization and using this property in Chapter 8.

Creating the Host HTML File

Listing 1-7 shows a sample startup HTML file that demonstrates how to add the GWT module to your application so that it is loaded when your application is started. It also shows how to add history support to the application. (Chapter 9 discusses how to handle history support in your applications.)

Listing 1-7. *Sample Host HTML File to Load the Module and Add History Support in the Application*

```
<html>
  <head>
    ...
    <!-- This script loads your compiled module. If you add any GWT meta tags,
         they must be added before this line -->
    <script language='javascript'
      src='com.apress.gwt.chapter1.FirstApplication.nocache.js'>
    </script>
  </head>
```

```

<body>
  <!-- OPTIONAL: include this if you want history support -->
  <iframe src="javascript:''" id="__gwt_historyFrame"
    style="width:0;height:0;border:0"></iframe>

    <!--Rest of HTML body goes here, for example, we have a table with two
    columns -- >
    <table align=center>
      <tr>
        <td id="slot1"/>
        <td id="slot2"/>
      </tr>
    </table>
</body>
</html>

```

Here I'll briefly explain some of the important parts of the previous HTML page:

- `<script ... src='com.apress.gwt.chapter1.FirstApplication.nocache.js'>`: After compilation, the application module is stored as a JavaScript file (with a `nocache.js` extension). The path for this file is added to the HTML page so that it can be downloaded and loaded when the page is referenced.
- `<iframe src="javascript:''" id="__gwt_historyFrame" ...>`: This adds history support to the application. This is an interesting functionality provided by the GWT framework that makes adding history management support to your application a simple task. (Chapter 9 discusses how to add history support to your applications.)

Steps to Create a GWT Application

The process of creating a GWT application can be broken down into the following four steps:

1. Create the basic project structure with the corresponding package. Add client, server, and public subpackages.
2. Add the Application module file, named `<appName>.gwt.xml`, to the basic package. Add the corresponding configuration details to the file.
3. Create an entry-point class (by implementing the `EntryPoint` interface) named `<appName>.java` in the client subpackage, and override the `onModuleLoad()` method with the required functionality.
4. Create an HTML file named `<appName>.html` in your application's public folder, with a script `src` reference to your application's package and with `<appName>.nocache.js` appended.