**Username:** David Cao **Book:** GWT in Action, Second Edition. No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

## 3.3. Enhancing the code

Here's BasicProject 's EntryPoint class (the class that's identified to have the onLoad method that's called when the application loads in the user's web browser):

```
public class BasicProject implements EntryPoint {
   public void onModuleLoad() {
     setUpGui();
     startHistoryHandling();
     setUpEventHandling();
}

Implementing
     EntryPoint

Providing the
     entry point
```

It implements the <code>EntryPoint</code> interface and the <code>onModuleLoad</code> method as you'd expect. Unlike <code>HelloWorld</code> 's entry point, where you added a <code>Label</code> to the <code>RootPanel</code>, <code>BasicProject</code> 's method calls two helper methods defined in the same class to set up the user interface and start GWTs history handling. You can put any valid Java code in the <code>onModuleLoad</code> method; our preference is to keep it clean and simple and use helper methods.

Setting up the GUI is achieved by the helper method shown in the following listing.

## Listing 3.2. Setting up BasicProject 's GUI

```
private void setUpGui() {
  buildTabContent();
  wrapExistingSearchButton();
                                                                     Creating content
  insertLogo();
  createFeedbackTab():
  styleTabPanelUsingUIObject();
                                                                    Programmatically
                                                                    styling content
  styleButtonUsingDOM();
  RootPanel.get().add(feedback);
  RootPanel logoSlot = RootPanel.get("logo");
                                                                      Adding content
  if (logoSlot!=null)logoSlot.add(logo);
                                                                      to page
  RootPanel contentSlot = RootPanel.get("content");
  if (contentSlot!=null) contentSlot.add(content);
)
```

In <u>listing 3.2</u>'s method you call several methods that help you create content either by inserting the logo image or wrapping the Search button in the HTML page, as you'll see in <u>section 3.4</u>, or scraping out the content for the tabs that we'll look at in <u>section 3.5</u>.

A lot of the content is styled via plain CSS or a GWT theme, but you can also choose to apply styles using methods in the widget's underlying UIObject class or through GWTs DOM class 2. You'll see how all this is done in section 3.8.

The rest of the method is concerned with manipulating the browser DOM (web page), inserting GUI components onto the page . For now you'll build the user interface directly in your code because it's the simplest way to get across the points in this chapter. But from <a href="https://chapter.6">chapter.6</a> you'll use the UiBinder approach—GWT's approach to declarative user interface definition. There's no right or wrong way here, but after using UiBinder in <a href="https://chapter.6">chapter.6</a>, we hope you'll find it the best way.

Let's take a look at how to create user interfaces with widgets, panels, and events.