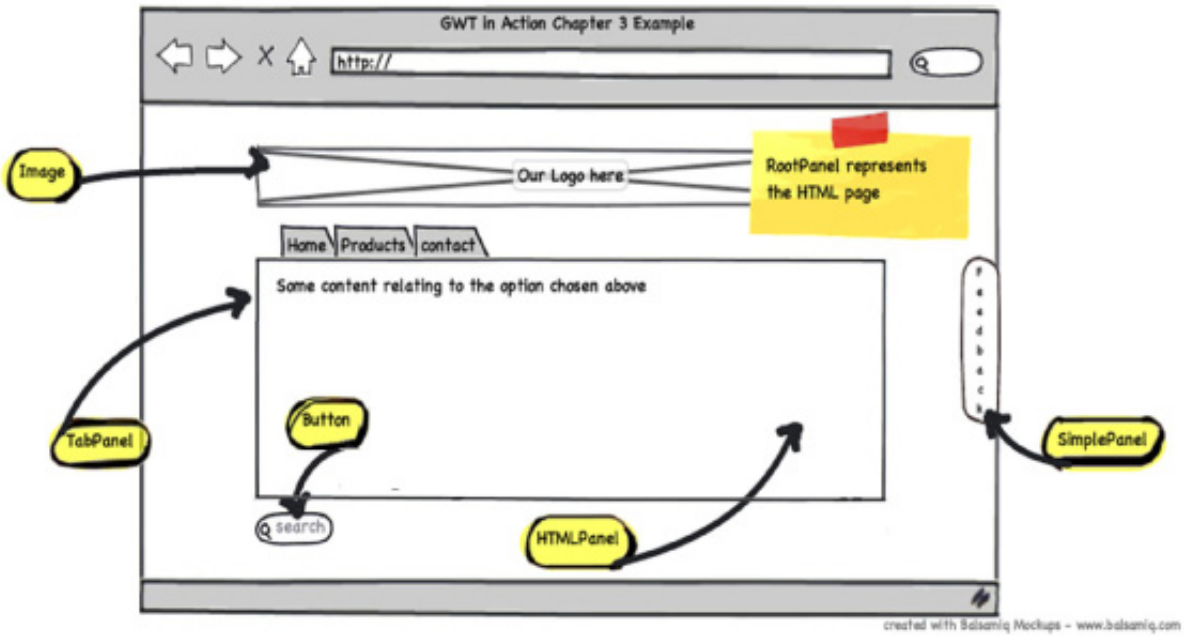

3.1. Reexamining the example application

Figure 3.1 highlights how you'll be implementing the mock-up. You'll use GWT widgets and panels to insert a logo as an `Image` widget into a predefined location in the application's HTML file, via a panel called `RootPanel`. That logo could be put in the HTML file directly, but we want to show a principle of GWT, and this is a simple way to do that.

Figure 3.1. Mock-up of chapter 3's main example application showing the layout in the browser window and which widgets and panels you'll use



You'll construct the tab area using a couple of GWT panels. The GWT `TabLayout-Panel` lets you look at three GWT concepts. You'll see GWT manipulating the Document Object Model (DOM) by extracting content from an HTML page and placing it in `HTMLPanel`s, which are in turn used in the tab panel. Building this component, you'll see how widgets and panels are added to each other.

In addition, we'll explore GWT history with this tab panel. Each click on a tab will change the content. If you then click the browser's Back or Forward button you'll change to the expected tab content automatically. As well as seeing history in action in the `gwtia-ch03-basicproject` project, you'll see a more isolated use of GWT history in the `gwtia-ch03-historyhelper` project.

On the right side of the application is a *feedback* tab, something that's becoming as common on websites nowadays as the "How is my driving?" stickers on commercial vehicles. Our application will explore how you can use some simple event handling and CSS styling to pop this panel in and out as the mouse is moved over it.

GWT can also wrap existing components in your HTML page—useful if you want to introduce GWT into a legacy application. Down on the bottom of the application is a Search button, which you'll define in the HTML page and wrap into the application to pop up a text box where a search term can be entered.

The last thing you'll do in the application is apply some Cascading Style Sheet styles to the panels and widgets. You'll do that in three ways: programmatically, by giving a widget a style name, and by using one of the provided GWT themes. These styles will give the application the final touches so it resembles our mock-up.

Out in the real world your GWT applications will be applying a more complicated version of some or all of these techniques together with some client/server interaction. You might bolster your design and development and support your maintenance team by applying the industrial-strength

techniques we cover later in this book, but it will still be an application that applies the previously mentioned techniques.

In this chapter we'll look into the `BasicProject` code and see how the items we've discussed are realized. Our aim is for you to understand how to get to figure 3.11 later in this chapter. As you read through this chapter, we ask that you take the example as something that allows us to show off various GWT techniques rather than wonder why we haven't done it in an easier and more sensible way.

We don't want it to look like we're pulling a rabbit out of a hat and present chapter 3's code and expect you to be impressed, so let's quickly discuss the enhancements we made to go from the framework of step 1 to the implementation of figure 3.11.

3.1.1. Enhancements

Table 3.1 lists the key enhancements we've made to the output of step 1 in chapter 2 to get to the final implementation of this chapter's application.

Table 3.1. Changes made to the similar output of chapter 2 to get to this chapter's project

Chapter 2 object	Chapter 3 changes
Overall	We created a project in the same way we did in chapter 2, except we called it BasicProject instead of HelloWorld so it wouldn't cause confusion.
HelloWorld.html versus BasicProject.html	We added content to the HTML file inside named div sections that will be used as the content for the Home, Products, and Contact tabs in the application.
HelloWorld.java versus BasicProject.java	We updated the onModuleLoad method so that it first calls a method to set up the UI for the application (setUpGUI) and then starts the GWT history-handling functionality. We added methods to support setting up the UI and handling events, as well as how to handle history. The history handling uses an enumeration to keep things consistent, but it might be a little confusing if you aren't familiar with Java enumerations—don't worry, when we get to the section describing history, we have a simplified example you can download to see how it works using only constants.
HelloWorld.css versus BasicProject.css	We added some style definitions so that the application looks prettier than having text on a screen and positions our feedback tab.
HelloWorld.gwt.xml versus BasicProject.gwt.xml	We added an inherit directive that directs the standard theme for GWT UI components to be included in the application—this makes the application look prettier than only text on the screen and saves us from having to add a lot of style definitions to the BasicProject.css. We added two other, commented-out theme inherits (they're there in case you want to play with them).

All of these changes will be covered in this chapter, sometimes in an obviously named section, sometimes spread out a little more where it makes the flow easier (unless explicitly mentioned otherwise, our future discussion is going to be about the `gwtiach03-basicproject` project).