
3.2. Updating the HTML

Every GWT application has at least, and 99% of the time only, one HTML page. It can contain anything you could normally have in an HTML file: images, hyperlinks, `div` objects, other JavaScript code, plain text, and so on. This is great for flexibility because it means you can incrementally GWT-ify an existing web page/application, adding GWT where needed. You can gradually swap out functionality for GWT implementations or, as you'll do here, put content in the page that the GWT application can use. Equally, the HTML file could contain no content; that is, it could be a blank page on which your GWT application will create everything.

As we noted in chapter 2, the file must contain a link to your GWT application's bootstrap JavaScript code. Bootstrap code? This is the nocache.js script produced by the compiler that will determine which permutation of your code to request and then start the application when loaded. For our application this script is called basicproject.nocache.js, and you can find it in the basicproject folder of the web archive once the application is compiled or you've run development mode at least once.

If you've used any of the GWT-creation tools, including GPE, then this link has already been created for you in the HTML file. If you're incrementally adding GWT to an existing web application, then you'll need to add this link yourself.

Be careful

If you change or add a `rename-to` tag in your main module definition, you'll need to make sure the bootstrap code in your HTML points to the right place. For example, if you have the following for your `MyApp` module

```
<module rename-to="look_here" />
```

it would mean that your HTML needs to link to the bootstrap code, as follows:

```
<script language="javascript"

       src="look_here/look_here.nocache.js">
```

Listing 3.1 shows the HTML file of `BasicProject` with the mandatory bootstrap code, a link to some optional styling, an iFrame to support history management (needed for Internet Explorer), and some content.

Listing 3.1. The HTML file


```
<!doctype html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <link type="text/css" rel="stylesheet" href="BasicProject.css">
    <title></title>
    <script type="text/javascript" language="javascript"
            src="basicproject/basicproject.nocache.js"></script>
  </head>
  <body>
    <iframe src="javascript:''" id="__gwt_historyFrame" tabIndex='-1'
         style="position:absolute;width:0;height:0;border:0">
    </iframe>

    <div id="logo"></div>
    <div id="home">Welcome to the application!</div>
    <div id="products">Here are some of our top products...</div>
    <div id="contact">
       You can contact us at
           <a href="mailto:a@a.com">1 Rudolf Street, The North Pole</a>
    </div>
    <button type="button" id="search">Search</button>
  </body>
</html>
```

The application's HTML file looks like (and is) a standard HTML file: 1, 2, 3, and 4 are the result of the creation tools; 5, 6, and 7 are standard HTML content we've added ourselves for this specific application.

You indicate the application will run in the browser's so-called standards mode[1] by declaring a `doctype` 1. This isn't necessary, because most of GWT will currently work in "quirks" mode (except the layout panels we'll discuss shortly), but because we're looking to the future, let's start off in the right direction (GWT might drop support for "quirks" mode in the future as browsers converge on standards). Putting the browser in standards mode also has some impact on the way you use panels, as you'll see in section 3.4.2.

[1] Information about various browser modes is available at www.quirksmode.org/css/quirksmode.html.

The application bootstrapping code is at 3, followed by two optional parts. First, there's a link to a stylesheet 2. You don't need styling, but your application will look much better for it. Section 3.8 looks at styling in more detail. Second, there's an iFrame 4, needed to support GWT's approach to history management in Internet Explorer (you'll see more about this in section 3.7). Like the link to bootstrap code, this item is added automatically by the creation tools. If you aren't using history support in your application, you could delete this entry.

This is all you need in an HTML file for a GWT application. But you'll add some additional content because you're expecting this application to do the following:

- Insert things in defined areas of the existing page. For example, you'll add an image to the `logo div` 5.
- Use existing components of the page. You'll wrap the `search` button 7 and use it in the application.
- Pull in content. The `home`, `products`, and `contact div`s 6 all hold content that you wish to delete from the HTML page and use directly within the tabs we'll show.

If you're sharp eyed, you may have noticed you give the `div`s and `Button` objects in the list an `id` as you declare them in the HTML page. It's that `id` that you'll use later in the enhanced Java code to get access to them.