9.2 Iteration D2: An AJAX-Based Cart

AJAX lets us write code that runs in the browser that interacts with our server-based application. In our case, we'd like to make the `Add to Cart` buttons invoke the server add_to_cart action in the background. The server can then send down just the HTML for the cart, and we can replace the cart in the sidebar with the server's updates.

Now, normally you'd do this by writing JavaScript that runs in the browser and by writing server-side code that communicated with this JavaScript (possibly using a technology such as JSON). The good news is that, with Rails, all this is hidden from you. We can do everything we need to do using Ruby (and with a whole lot of support from some Rails helper methods).

The trick when adding AJAX to an application is to take small steps. So, let's start with the most basic one. Let's change the catalog page to send an AJAX request to our server application, and have the application respond with the HTML fragment containing the updated cart.

On the index page, we're using button_to to create the link to the add_to_cart action. Underneath the covers, button_to generates an HTML form. The helper

```
<%= button_to "Add to Cart", :action => :add_to_cart, :id => product %>
```

generates HTML that looks something like

```
<form method="post" action="/store/add_to_cart/1" class="button-to">
  <input type="submit" value="Add to Cart" />
</form>
```

This is a standard HTML form, so a POST request will be generated when the user clicks the submit button. We want to change this to send an AJAX request instead. To do this, we'll have to code the form explicitly, using a Rails helper called form_remote_tag. The form_..._tag parts of the name tell you it's generating an HTML form, and the remote part tells you it will use AJAX to create a remote procedure call to your application. So, edit index.rhtml in the app/views/store directory, replacing the button_to call with something like this.

Download depot_l/app/views/store/index.rhtml

```
<% form_remote_tag :url => { :action => :add_to_cart, :id => product } do %>
  <%= submit_tag "Add to Cart" %>
<% end %>
```

You tell form_remote_tag how to invoke your server application using the :url parameter. This takes a hash of values that are the same as the trailing parameters we passed to button_to. The code inside the Ruby block (between the do and end keywords) is the body of the form. In this case, we have a simple submit button. From the user's perspective, this page looks identical to the previous one.