A better way to handle the problem is to create the cart HTML but set the CSS style to display: none if the cart is empty. To do that, we need to change the store.rhtml layout in app/views/layouts. Our first attempt is something like this.

```
<div id="cart"
    <% if @cart.items.empty? %>
        style="display: none"
    <% end %>
>
  <%= render(:partial => "cart", :object => @cart) %>
</div>
```

This code adds the CSS style= attribute to the <*div*> tag, but only if the cart is empty. It works fine, but it's really, really ugly. That dangling > character looks misplaced (even though it isn't), and the way logic is interjected into the middle of a tag is the kind of thing that gives templating languages a bad name. Let's not let that kind of ugliness litter our code. Instead, let's create an abstraction that hides it—we'll write a helper method.

Helper Methods

Whenever we want to abstract some processing out of a view (any kind of view), we want to write a helper method.

If you look in the app directory, you'll find four subdirectories.

```
depot> ls -p app
controllers/    helpers/        models/         views/
```

Not surprisingly, our helper methods go in the helpers directory. If you look in there, you'll find it already contains some files.

```
depot> ls -p app/helpers
admin_helper.rb         application_helper.rb   store_helper.rb
```

The Rails generators automatically created a helper file for each of our controllers (admin and store). The Rails command itself (the one that created the application initially) created the file application_helper.rb. The methods we define in a controller-specific helper are available to views referenced by that controller. Methods in the overall application_helper file are available in all the application's views. This gives us a choice for our new helper. Right now, we need it just in the store view, so let's start by putting it there.

Let's have a look at the file store_helper.rb in the helpers directory.

```
module StoreHelper
end
```

Let's write a helper method called hidden_div_if. It takes a condition, an optional set of attributes, and a block. It wraps the output generated by the block in a <*div*> tag, adding the display: none style if the condition is true. We'd use it in the store layout like this.