CHAPTER 8

# Rich Web Clients

The term "rich web clients" is typically interspersed with the terms "Web 2.0" and "AJAX." It refers to web interfaces that provide a relatively high level of interactivity compared to traditional click-and-reload web interfaces. These rich web interfaces are thought of as the next generation of user experience on the Web (hence the association with Web 2.0), and the technology most often brought to bear to create them is JavaScript (hence the association with Asynchronous JavaScript and XML, or AJAX).

In this chapter, we will explore Seam's support for integrating rich web clients with your Seam components, using these components to provide the server-side data and functionality needed for the web interface. The key factor here is that the data and functionality is accessed directly from the web client (the browser) rather than through the standard JSF web request processing life cycle. This enables more immediate and interactive user experiences in the web interface, among other things.

Much of the discussion in this chapter involves JavaScript and dynamic DOM manipulation, which are typical parts of any AJAX interface. I am not going to provide a tutorial on JavaScript here. If you are not already familiar with the basics of JavaScript and accessing the page DOM at runtime, you might want to defer reading this chapter and instead get familiar with these foundations first.

## What Is a Rich Web Client?

Typical web interfaces are like the ones you've seen so far in this book. The user navigates through a series of pages to accomplish tasks, like creating entries in catalogs, searching for information, and so on. The interaction paradigm is very simple: a page is presented to the user, the user edits fields and/or clicks a link or a button on the page, and he or she is taken to another page. In between the two page views, the client browser sends an HTTP request to the site, and this request can cause information to be updated, data to be retrieved, and so forth. This sequence (view a page, perform an action on that page, and be redirected to another page) continues until the user is done with whatever he or she is doing.

This mode of interaction leaves a lot of room for improvement. It doesn't feel very interactive to users, for obvious reasons. User interactions are broken down into relatively

coarse transactions, delimited by pages and browser redirects. Users experience a "read/edit/wait" sequence, where they are presented with a page, they have to visually parse what's on the page and what's being asked of them, decide what action they want to take, take it, and then wait for the browser to load up the next page.

A rich web client implements a more interactive experience for the user. Transactions with the system are broken down into smaller units of work, and users see more immediate feedback from their actions (more immediate than a full browser page load, at least). A rich web client allows a user to push a button and see the impact immediately in the page that he or she is viewing. There's no disorienting context switch to a new page, and the user feels more effective and more efficient.

The most commonly used technology currently for implementing rich web clients is AJAX. AJAX is more of a technique than a specific API or tool. It involves the use of JavaScript to make "intrapage" requests back to a web server to perform actions on the user's behalf. The JavaScript is loaded in the page along with the HTML used to display the contents, and as the user interacts with the HTML elements (clicks buttons, types into text fields, etc.), JavaScript calls are made in response to the user's actions in the page. These calls can, among other things, exchange XML information with some server-side component, such as a CGI script, a Java servlet, or any active code. This exchange might bring additional information into the page, update server-side data, or just about anything else that's needed to accomplish what the user is trying to do. Again, the difference is that these server calls are made without a full page request being made by the web browser, so a user sees more immediate feedback to his or her actions in the page.

## Seam's Remoting Services

Seam's primary support for rich web clients is provided in its component remoting services. These services will dynamically (and automatically) generate JavaScript client stubs for your Seam components and use them from a web browser client to interact directly with your server-side Seam components.

Figure 8-1 shows the runtime model used by the Seam remoting services, and the interaction that they enable between your Seam components and client-side JavaScript code.

At the bottom of the figure is your Seam application running in an application server. At the top is the client browser. The Seam remoting services consist of a set of Java code running on the server and another set of JavaScript code running in the browser. The server-side code generates a JavaScript binding for any Seam components that you choose to export. This JavaScript code runs in the browser and can be used to make direct calls back to the Seam components. Seam's remoting services handle all of the communication between the browser and the Seam components. This support code converts JavaScript calls made in the browser to XML and back again. Similarly, the Seam remoting services running on the server convert the XML coming from the browser into