StrutsWebIslands

First some well-known facts: a browser builds page history for visited resources. Page history can be navigated with Back and Forward buttons. If page history contains less than two locations, Back button is not enabled.

A Simple Data Entry Form

Let us consider a login action:

- Browser submits username/password to login action using POST request.
- Login action validates the input information; if account is not found, login action generates error message and redirects to the same address where the user's credentials were submitted at the first place, that is, to itself.
- After redirect, browser loads login action using GET request, and action redisplays login form along with error messages.

This cycle repeats, until a user makes it right.

Influencing Client-Side Resource History

Browser has to retain the chronological order of visited resources in the page history. If an action uses Redirect-After-Post pattern, it produces a pair of POST/GET requests for each data submit. Therefore, browser should interlace POST entries with GET entries in the resource history list. After three login attempts the history list will look like this:

POST » GET(view) » POST » GET(view) » POST » GET(view) ...

Well, not exactly. It would be so if there were no redirection from POST to GET. With redirection, browser "forgets" the location that preceds redirection, and stores only redirected requests:

GET(view) » GET(view) » GET(view) ...

What if all GET requests were identical? Input data can be submitted with POST request, and can be stored in the session, no matter is it correct or not. Therefore, all GET requests can be "cleaned" from input data. A smart browser does not see any reason to include exactly the same GET requests in the history list several times. This is how Microsoft Internet Explorer and Mozilla/Firefox work, they keep only one GET request out of several identical requests:

GET(view) ...

Returning to login action example, we can see that MSIE and Mozilla do not add new entries in the history list when a user tries to log in several times. This, in turn, means that if a user wants to bail out the login process, he needs to click Back button only once to return from login dialog to a preceding page. If GET requests were not identical, a user would have to click Back button as many times as many input attempts were made.

Struts Is About Actions, Not Pages

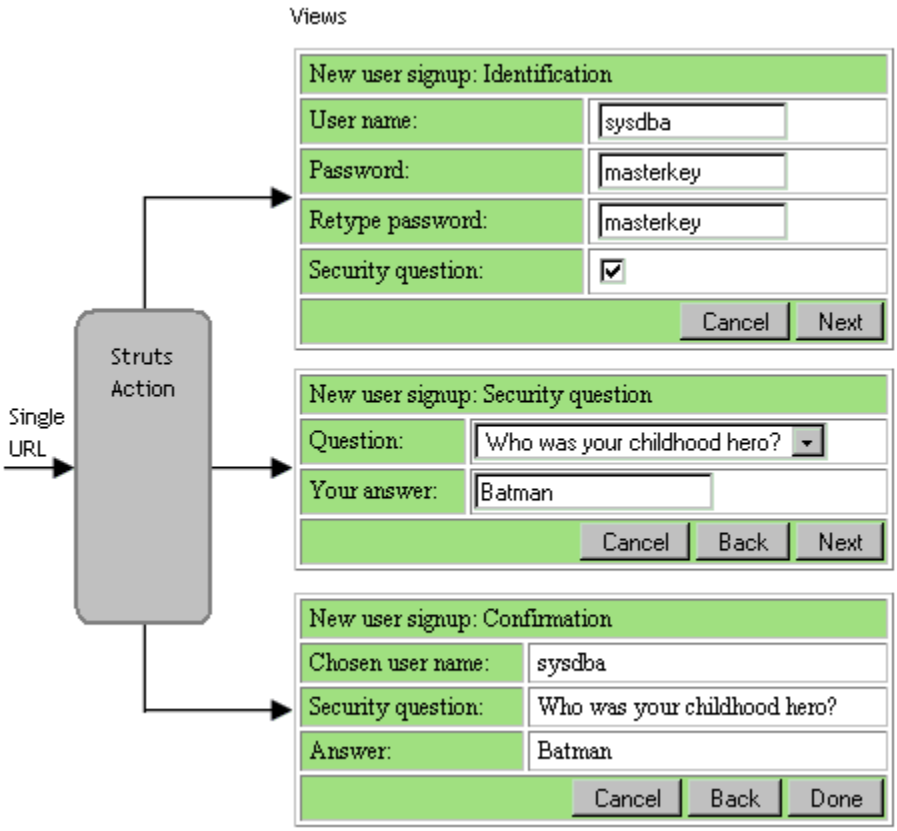
But wait, there's more! Consider a login action that has two pages, each corresponding to different login state. These pages are rendered using one GET location - the location of login action. This means, that despite different content of the pages, they are seen by browser as one resource. When login action is invoked for the first time, it presents a login page.

After successful login the same component displays log out page along with user information and "Log Out" button. Thus, after user logs in, it is impossible to navigate back to login page, because there is no address that identifies a particular **page**. The GET request identifies the **action as a whole**.

Single-Page Resources In Struts Without Ajax

This trick allows to create single-page resources which can be emphatically called *web islands*. Essentially, a web island is a multi-state multi-page resource. This is where Struts as Front Controller framework, shines.

As an example, consider a wizard shown below. Its pages correspond to one action and have the same web address. There is no way to go back or forward to another wizard page using browser Back or Forward buttons, because pages are not addressed individually. This approach protects a user and the application from resubmitting data from the stale page, and makes user interface more reliable.



The concept of a web island changes traditional navigation paradigm a little. Back and Forward buttons are used to navigate between web islands, while navigation within a web island is completely controlled by a corresponding web component. A web island can be bookmarked as a whole, but not a particular page of it.

The single-page trick does not work in every browser, for example, it does not work in Opera. But it is a very sweet icing on the cake for users of MSIE and Mozilla, and possibly other browsers. MSIE and Mozilla together cover about 98% of web browser market, so I think that web island approach is worth the efforts.