****

Philip Pannenko

Zaharadeen Garuba

James Murnane

**Introduction**

There are a handful of common-sense attributes a good website should have. Certainly it should be intuitive for the visitor. This not only means it is wise to use a standard layout, but it also means meeting the users expectations with respect to what features are offered, how one navigates, how many clicks it takes to complete a task, etc. Also important is the quality of the code behind any glossy interface. The code should be clean, readable, stable, and created with a realistic understanding of how it’s likely to evolve. A solid foundation, along with a healthy dose of good management and an integrated QA processes, will help fend off bugs and dead-end functionalities that plague even the largest of websites - especially the largest of websites.

The Java Bean Shop follows these guiding principles. Below is a description that shows an intuitive interface that offers all the expected features: login control, product lists and views, search, a shopping cart and shipment reviews. In addition, the code is streamlined, has a logical division of components (JSP, Java Beans and servlets), and leverages AJAX for simple communication between the front and back ends.

**The Front of the Store (interface)**

There were a handful of interface requirements, all of which were implemented:

1. Standard layout of a header, footer, left navigation panel and one or two display panels.
2. AJAX-based search box
3. Login controls with user identification
4. Product Views in which a user can select “Add to Cart”
5. A shopping cart menu bar icon/link
6. A shopping cart screen

**The Back of the Store (code)**

Numerous elements come together to provide a good user experience and also form a strong code foundation:

1. *Ajax-based pagination, sorting and search* – These operations, though different in purpose, share the exact same flow: an interface event forces a Javascript function to send a data request to the server and inserts the returned html code upon its return. The JS code is generic enough to handle these different uses (file: ajax-utils.js).
2. *Categories* – With an eye toward flexibility and extendibility, product categories are not hardcoded. They are data as much as any product.
3. *Header Menu updates based off of current user and cart content count* – Although small, these are necessities in today’s online shops. Providing basic info such as login state, username and a shopping cart count gives the customer a sense of connection, status and state.
4. *Form validation on order confirmation page* – Such validations prior to payment processing are realistic. Without this, orders without destinations could be paid for, shipped, but never delivered.
5. *Info/Error message per action. (blue bars and red bars w/ messages)* – There are few standards regarding the format of messages to the user (confirmations, errors, etc.). That said, it is expected that the messages (a) exist, (b) communicate effectively, (c) are easily viewable. And this is what we have.
6. *User account view/edit/create/delete ability* – Very basic, completely realistic, and absolutely necessary functions. They exist in full working order.
7. *Ability to make order and view the order confirmation without having an account* – It is undesirable to let the user think the system is crippled in any way. For example, if a new or existing user is not logged in, one doesn’t want to forbid that user from filling a cart and viewing the order prior to shipment. In fact, you want them to do this. Get them as close as possible to the transaction.
8. *Feature where if the current user is performing a confirm cart contents, the user's details are automatically placed in the confirm form*. – This is an older standard. It allows yet one more chance for the user to confirm what was ordered and, with the display of such information, make a merchant seem that much more connected to the user.
9. *BeanUtils usage to populate bean from JSP form* – Basic good design pattern. For example, a user bean is populated with account information from the screen. With this example, there would be an intermediate servlet to which the account information is ‘posted’ (always desirable when credit card info is involved).
10. *MVC layout. JSP -- calls --> Servlet -- renders --> JSP* – The MVC layout is desirable with a large scale commercial site. While one could use a more basic configuration with a small store, we are following the principal of coding with an eye toward the future.
11. *Template framework for beans/pages. (code reuse)* – Reuse is supported through generic coding.
12. *Serialization of data to disk (persist data)* – Two comments: (1) This is necessary for continuity of session data, and (2) it is important to have all personal data, serialized or otherwise, within the WEB-INF folder, which is indeed the case.