

HW4 Amazon.edu

For HW4, we'll use servlets to build a little bookstore called Amazon.edu. The conceptual structure of Amazon.edu is similar to the CGI homework, but using servlets makes things a little easier. Amazon.edu is due midnight ending Tue June 4th . You may use at most four late days on this homework. This handout describes Part A, parts B and C are on a separate handout.

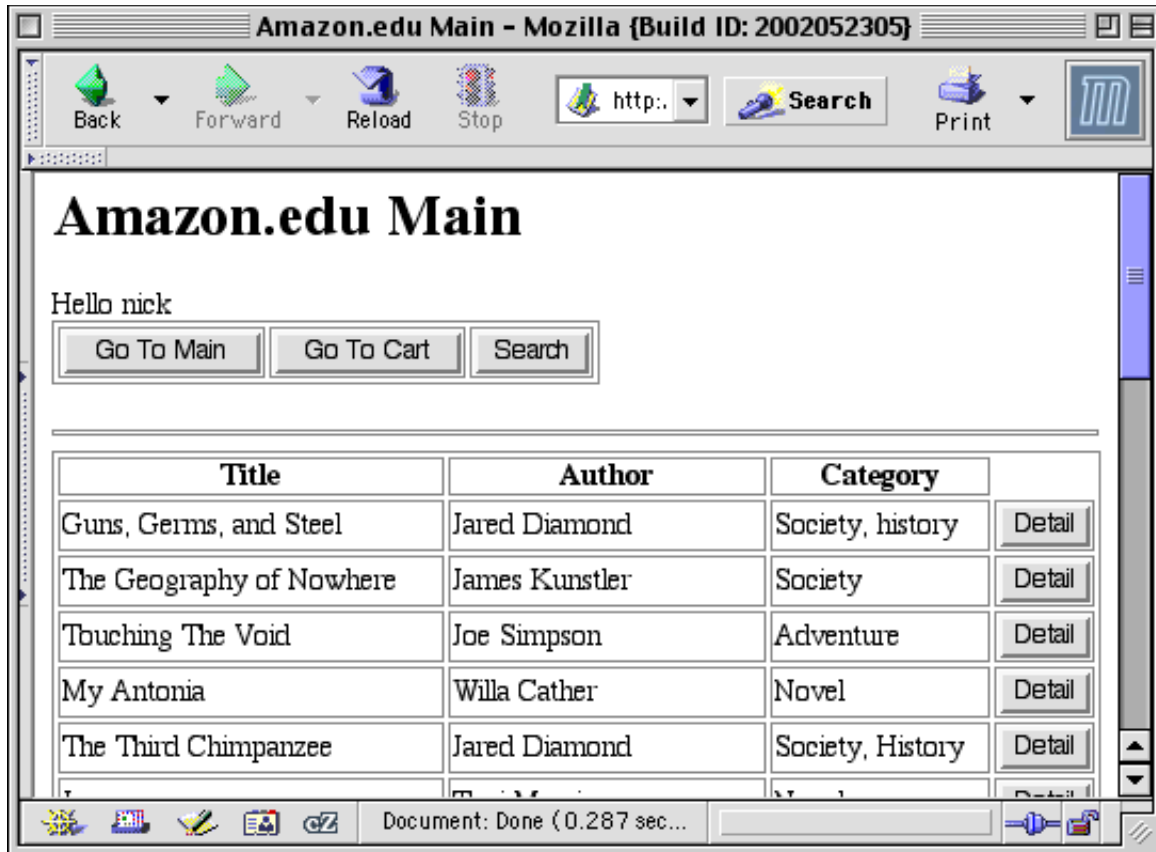
Part A

For Part A, we will build the main part of a little store. Part B will add the check-out feature, and Part C will add in a JavaScript search feature. The solution will use CGI-type logic to create the main page and the cart page, and a JSP to create the detail page. As usual, P/NC students may work in teams of two and do not need to do parts B and C.

1. Main

Part A has a simple 3-page interface. The first page you see is the "main" page which features

- The title and h1 "Amazon.edu Main"
- "Go To Main" and "Go To Cart" navigation buttons (described below). The little "Hello nick" and the "Search" button are from parts B and C -- ignore them for now.
- A table listing all the books in the database by Title, Author, and Category (but omitting the description). There is a "Detail" button for each book. The top row of labels uses "<th>" while the others use "<td>".



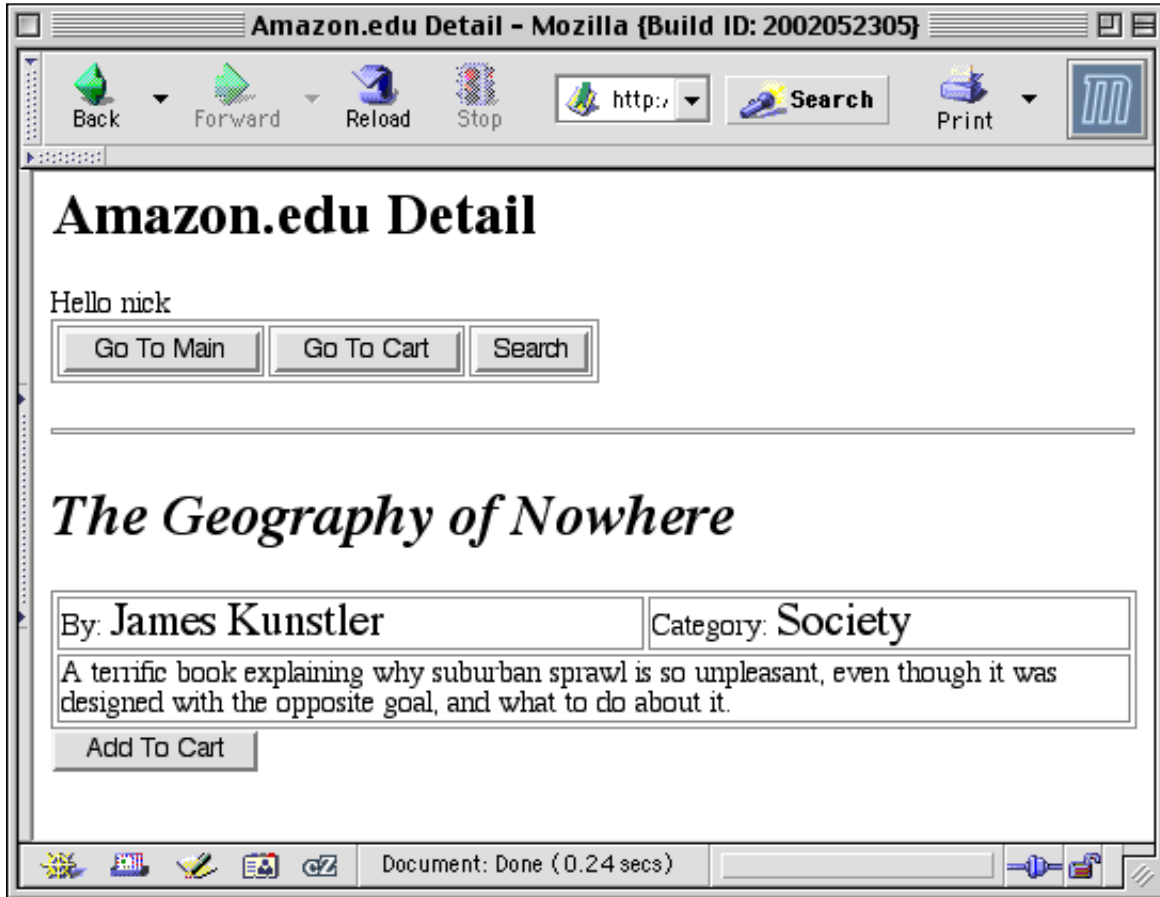
The TextDB class provided for you takes care of the details of reading the book database into memory.

2. Detail

Clicking the detail button for a book leads to a page showing...

- The title and h1 "Amazon.edu Detail"
- The standard buttons
- The information for that book presented with the following format...
 - The title is in a <cite> tag inside an <h2> tag
 - The author and category are in two cells in the first row one of a table. Use ... to make the author and category fonts a little bigger.
 - The description fills the second row of the table. Use <td colspan=2> to get the td to fill out both columns
- Finally, there's a "Add To Cart" button that can add the book to the cart

The detail page will be implemented as a little JSP. The JSP is well suited to building the nested HTML structure of the detail page.



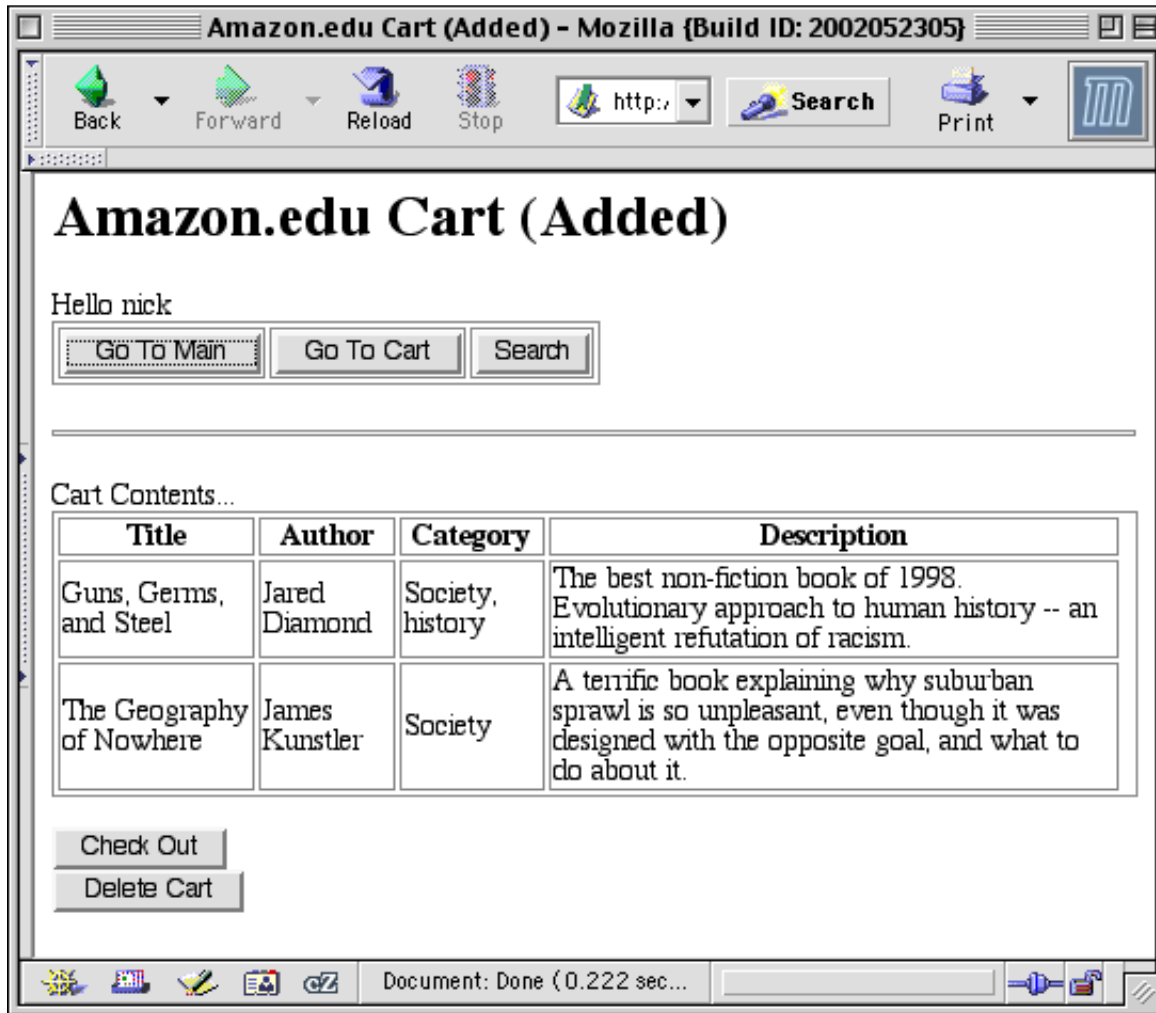
Clicking the Add To Cart button adds the book to the session shopping cart, and goes to the shopping cart page (below) to show the cart state. Clicking the "Go To Cart" button from here and from the main page are both similar to clicking Add To Cart from here. The difference is that the Add To Cart button changes the cart state. Adding a book which has already been added should go to the cart page but without changing the quantity—the only quantities allowed in our little binary shopping hell are 0 and 1.

3. Cart

The Cart page shows a table similar to the main page, but containing the subset of the books that have been bought, and with the word "bought" in bold in the rightmost column. The cart page shows...

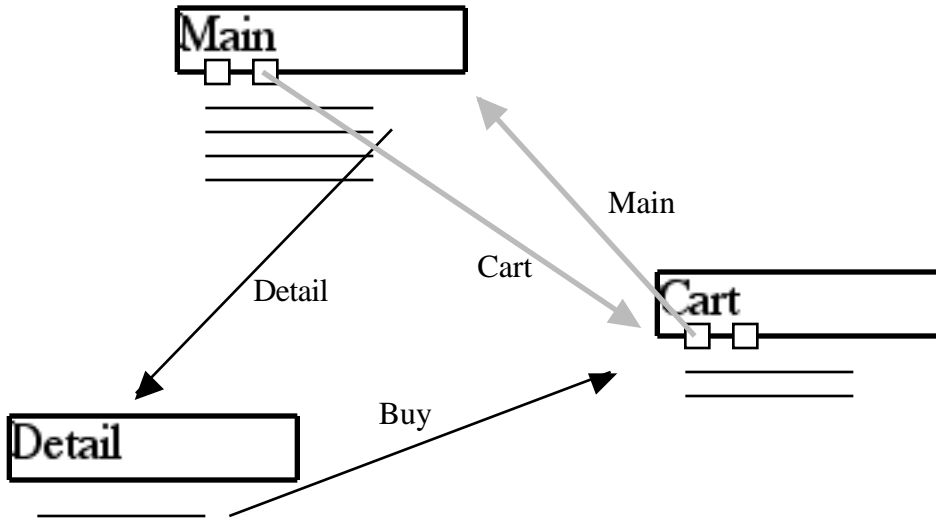
- The title and h1 "Amazon.edu Cart" if coming from the Go To Cart button. If we are coming here because of a click on the buy button, the string "(Added)" should be added to the titling to provide more rational feeling feedback for the buy button.
- The standard buttons

- The table should show the current set of added books — Title, Author, Category, and Description for each one.
- The "Delete Cart" should change the cart to contain zero books and return to the main page. We'll ignore the "Check Out" button for now.



Navigation

Here's a map showing the most common transitions — the book buying path is in black, and the navigation button paths between main and the cart are in gray...



Implementation Ideas

As with the CGI homework, the solution will be to build a single servlet which looks at the request bindings to see what page to generate. Internally, the servlet session feature will be the perfect way to keep track of the current state of the cart.

doGet()

As with the CGI homework, the servlet needs to figure out which page it's supposed to send back. Use `(request.getParameter("buttonName") != null)` to check for bindings in the request.

To keep things organized, separate out the `doDetail()`, `doCart()`, and `doMain()` methods to generate the individual pages.

HTML debugging

Probably the easiest way to debug what is going on is to print debugging messages into the HTML so you can see them in the browser.

Code Re-Use

There's great potential for code re-use here. Don't be afraid to make up your own private utilities with a few parameters if the utility can then be called from several places. Create a utility method to generate the common HTML at the top of every page. Create a row-writing method that you can use to create the table rows for the main page and the cart page.

Session

The session is the perfect way to track the current collection of bought books. Buying books and printing out the contents of the cart can use the ArrayList in the session. Every time the servlet is invoked, the same session will be present. As a result, the Amazon servlet should not store session specific state in traditional instance variables. The session object provides all the necessary storage for the servlet.

Since the database never changes, we do not need to do the nasty "id" scheme as in the CGI homework. The row number will be sufficient to identify each book.

TextDB

The TextDB classes provided for you does the work of reading the text database into memory. The servlet should have a single "db" instance variable that points to a TextDB object that contains all the row data. The first time doGet() is called, create the TextDB object, passing the name of the file it should read from...

```
if (db == null) {
    db = new TextDB("books.txt");
}
```

The TextDB is created the first time and stored in the db ivar. For subsequent calls to doGet(), the db is ready. Your code can send two messages to the db to get data out of it..

- int size() -- the number of rows in the db.
- String[] getRow(int num) -- return an array of strings representing one row. Row 0 is the column headers for the database, and the subsequent rows are data.

Detail Page

There's a simple BookBean class provided for you. Create an instance of BookBean on the request for the JSP. Your JSP should be called "/detail.jsp". If there are errors in your JSP, the error messages generated will be quite cryptic. JSPs are very sensitive to syntax -- compare your JSP with the lecture example carefully to check your syntax.

Setup Logistics

To see how to run sevlets/JSPs on leland, see the "servlet setup" instructions on the course page. The starter files have the routine steps done for your for the version of tomcat installation in the CS193i course page.

Parts (B) and (C) will be in a separate handout.