Click on the navbar

**
Cookies
**

I, the user, click on a link. Behind the scenes, that sends a request to a controller servlet.

Get request to Cookies servlet

Cookies Servlet doGet method response.sendRedirect ("/cookies.jsp"); This forwards the request to the view (the jsp)

The controller servlet decides where to forward that request. In this case, it decides to send the request to the cookies view, where the user is expected to fill a form.

Get request

cookies.jsp:
<form method="POST" action="/Cookies">
form sends information to /Cookies

After filling out that form, the form information is packaged into parameters and sent BACK to the cookies controller -- but this time it's a post request so it gets handled by doPost

Post request to /Cookies (goes to doPost method)

Packages the information from the form in parameters named according to the name attribute of the input element, so input type="text" name="name" will create a parameter called "name"

Cookies servlet doPost method receives the Post request form cookies.jsp. We use the request.getParameter method to retrieve the information that was sent by the form. Now we set session attributes to hold the form info.
HttpSession session = request.getSession();
session.setAttribute("name", name);
Now we can display the user's name and favorite cookie on the cookies.jsp

the Cookies controller doPost extracts the parameters using request.getParameter, and sets them as session attributes. The information is now available throughout the app

to /Deals

response.sendRedirect("/Deals"); sends a get request to /Deals. We don't need to send a request object, because session information is automatically sent in all get requests in the headers.

In the /Deals doGet: String name = (String) request.getSession().getAttribute("name"); We now have access to the information the form.