

Servlets - Part 3

URL Re-Writing

Suppose cookies are not supported by the client

Could put the session id (or other info) on each URL sent back to the client...

`http://foo.com/servlet/cart?id=123xyz?a=b`

URL Parse

The server needs to parse out the parts of the above url, to see that the servlet to run is `/servlet/cart`, the session is `123xyz`, and the form bindings are `a=b`

encodeURL()

The servlet `HttpServletResponse` object responds to an `encodeURL()` message that will re-write the session id onto a url for you if necessary. The `ServletContainer` will pick up this info on later requests automatically to determine the right session.

URL Re-Writing is a Hack

The server needs to re-write every url on the page, since we don't know which the client will click.

This makes generating the HTML more complex throughout

If the user clicks off our site, to site X, and then clicks back to our site from X - the session id is lost. Notice that with cookies, this case actually works.

When most browsers and users support cookies, then url-rewriting should probably just go out of use.

Amazon Demo

Surf over to amazon and login

Look through your cookies to find your session id.

Now look at the URL at the top of the browser and look in the generated HTML to see how URL-rewriting tries to maintain the session

Servlet Redirect

The `Servlet Request` and `Response` objects have support for all sorts of common server-side operations.

The response responds to a `sendRedirect()` message which does a 301 redirect for you.

This commits (sends) the response, so do not put further info on the response header after redirecting (for example, do not add cookies after redirecting).

Cookies vs. Security

Don't store sensitive data in the cookie - the cookie is stored on the client in a way which is visible and editable by people with access to the client machine.

- NO store password on cookie

- NO store credit-card number on cookie

- NO store `auth=yes` state on cookie

Ok to store things that are used in with server side info to do thing securely...

OK store username on cookie, so recognize user next time they log in (we will ask them for their password if they try to do something significant, in case it's not the right person)

OK store session-id on cookie (as above, we'll ask for a password if we're not sure that it's the right person)

Note that the exposure of using cookies are the same as using hidden fields. In both cases, we depend on the site not to store sensitive info to the client side.

Cookies vs. Privacy

Session cookies enable a site to track that the HTTP requests over, say, an hour, all go together.

Persistent cookie enable a site to track that several sessions belong to the same person.

Cookies enable the site to get a better picture of what a browser looks at, possibly across multiple sessions.

For a site like Amazon, they can try to correlate the surfing with their buying records - try to show products that will be appealing.

The point: cookies enable the site to get a better picture of the surfing pattern. Plain web-logs (without cookies) provide similar though not as good data.

DoubleClick - Web Bugs

Suppose the foo.com page includes an small image from doubleclick.com. bar.com also includes the image.

The image response can set a cookie, which is included on all later requests.

DoubleClick can build up a picture of the set of sites that one person visits.

This seems annoying.

In reality, it is more likely that they are developing stats like "browsers on site X also tend to visit site Y with percentage Z" type stats, but without tying it to a particular person - like nielson ratings for TV.

Privacy vs. Utility

People have an instinct that they don't want vendors to know anything.

IMHO, this is a misguided instinct, but we will take that up later in the security/privacy lectures.

Certainly, surfing amazon, buying books, etc. does generate information about the buyer.