UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Session Management

innovative • entrepreneurial • global

Persistent State in HTTP Servlets

- HTTP transactions are made in isolation of one another
  - do not have a mechanism for keeping track of a request or request data sent using a web browser
  - said to be “stateless”
- **Benefit**
  - Client browsers do not notice when a server goes down and comes up quickly
- **Drawback**
  - difficult to produce groups of pages for collecting information to produce picture of the user’s web experience

Session Tracking Methods

1) Cookies

- small size of information left by the server at client machine (in browser cookies repository)
- **misinformation about cookies**
  - Never interpreted or executed
  - browsers generally only accept 20 cookies per site
  - and 300 cookies and limited to 4 kilobytes per size
  - cannot be used to fill up someone's disk or launch other denial of service attack
- problem
  - user disable browser cookies
    - to protect privacy

Session Tracking Methods

2) URL Rewriting.

- append some extra data on the end of each URL that identifies the session, and the server associate that session identifier with data it has stored about that session.
- Excellent solution with browsers that don't support cookies or where the user has disabled cookies.
- However, it has most of the same problems as cookies, namely that the server-side program has a lot of straightforward but tedious processing to do.
- In addition, you have to be very careful if the user leaves the session and comes back via a bookmark or link, the session information can be lost.

Session Tracking Methods

3) Hidden form fields.

- HTML forms have an entry that looks like the following:

<INPUT TYPE="HIDDEN" NAME="session" VALUE="...">

- This means that, when the form is submitted, the specified name and value are included in the GET or POST data.
- This can be used to store information about the session.
- However, it has the major disadvantage that it only works if every page is dynamically generated, since the whole point is that each session has a unique identifier.

Session Tracking Methods

4) HttpSession API

- high-level interface built on top of cookies or URL-rewriting.
- use cookies if the browser supports them,
- automatically revert to URL-rewriting when cookies are unsupported or explicitly disabled.
- servlet author doesn't need to bother with many of the details,
  - doesn't have to explicitly manipulate cookies
  - or information appended to the URL,
  - automatically given a convenient place to store data that is associated with each session.

HTTPSession

- Provides a way to identify a user across more than one page request
- create a session between an HTTP client and an HTTP server.
- session persists for a specified time period, across more than one connection or page request from the user.
- usually corresponds to one user
- allows servlet to
  - View and manipulate information about a session, such as the **session identifier**, **creation time**, and **last accessed time**
  - **Bind objects** to sessions, allowing user information to persist across multiple user connections

HTTPSession

- Looking up the session object associated with the current request,
  - **`HttpSession session = request.getSession();`**
    - Returns the current session associated with this request, or if the request does not have a session, creates one.
  - **`HttpSession session = request.getSession(boolean param);`**
    - **param=true** - to create a new session for this request if necessary;
    - **param=false** to return null if there's no current session

HTTPSession

- Binds an object to this session, using the name specified.
- If an object of the same name is already bound to the session, the object is replaced
  - **void setAttribute(java.lang.String name, java.lang.Object value)**

- Returns the object bound with the specified name in this session, or null if no object is bound under the name
  - **java.lang.Object getAttribute(java.lang.String name)**

HTTPSession

- Specifies the time, in seconds, between client requests before the servlet container will invalidate this session. A negative time indicates the session should never timeout.
  - **void setMaxInactiveInterval(int interval)**
    - **interval** in seconds
    - default **30** minutes

- Invalidates this session and unbinds any objects bound to it (remove current session)
  - **void invalidate()**