PHP Session Management

Prof. Jim Whitehead

CMPS 183 – Spring 2006

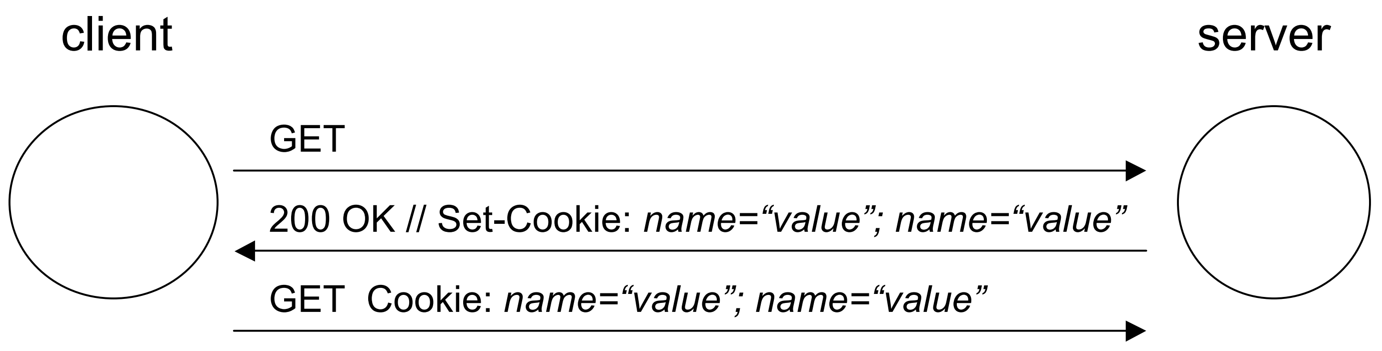
May 5, 2006

Session Management

- In Web applications, it is frequently desirable to:
  - Remember if a user has visited a site previously
  - Remember the sequence of pages visited during a session
  - Associate information with a user session
    - Like a shopping basket
- To accomplish these things, need to have "stateful" connections between client and server
  - That is, association information (state) with the connection
  - HTTP was originally designed to not be stateful, leading to the addition of the Cookie mechanism

Cookies

- A mechanism for adding state to HTTP requests.
- Permits *name, value* pairs to be preserved across multiple HTTP requests.
- Basic approach:
  - Server sends a Set-Cookie HTTP header in its response.
  - Value of the header includes name, value pairs.
  - The client stores the name, value pairs, and proactively sends them to the server (in the Cookie header) with every request thereafter.
  - The Cookie protocol piggybacks on top of HTTP
    - Information flow is in the reverse direction of most HTTP requests


client
server
GET
200 OK // Set-Cookie: name="value"; name="value"
GET Cookie: name="value"; name="value"

Cookies in PHP

- How Cookies are set in PHP:
  - Use built-in setcookie() function. Most simple form is:
  - setcookie(name, value)
- It is also possible to specify when the cookie will expire, the URL path and domain for which it applies (defaults to those of request URI).
- Accessing Cookie values:
  - $local_variable = $_COOKIE[name]
- To delete a cookie, use setcookie with the same cookie name, and an expiration value in the past.

Problems with Cookies

- Problems with Cookies
  - Browsers can refuse to accept cookies.
  - Additionally, it adds network overhead to send lots of information back and forth.
  - There are also limits to the amount of information that can be sent
  - Some information you just don’t want to save on the client’s computer.

PHP Sessions

- The solution: store session information on the server, and have the client only store an identifier for its information as stored on the server.
- The identifier is known as a session ID. The session ID is stored using a cookie (can be passed as a GET parameter as well)
- The server then uses the session ID to retrieve the information it has stored on the server.
- Session information is typically stored in files on the server, though options exist for using shared memory, and also writing your own handlers (e.g., to use a database for storage)

Using Sessions in PHP

- To start a session:
  - session_start()
  - Creates a session identifier
  - Session identifier is passed between client and server either as a Cookie, or in GET parameters
- Then, can create, access, and modify session variables:
  - $_SESSION[session_var_name] = value;
  - $_SESSION is only available once you call session_start()
  - $local_variable = $_SESSION[session_var_name];
  - Can check if session variable is set by using isset();
- To end a session:
  - session_destroy();

Security of Session Data

- In general, cannot guarantee that session data will remain private
- Often, the session data files can be read by any web application on the same server
- The session ID can be grabbed by looking at the GET parameters (for GET-based passing of the session ID), or by eavesdropping the on-the-wire protocol (to get the cookie with the session ID)
  - If the session holds a password, someone can then “replay” the session ID back to the server
- Cookie data, though stored on the client side, are sent across the wire in-the-clear
  - Client machines might be compromised, such as by malicious software inadvertently downloaded, or by a virus