PHP: Sessions and PostgreSQL Connectivity

Tom Kelliher, CS 318

Feb. 18, 2002

1 Administrivia

Announcements

Assignment

Catch up on the reading!!!

From Last Time

SQL queries.

Outline

1. Introduction.
2. Sessions.
3. PostgreSQL connectivity.
4. Example code walk-through.

Coming Up

PHP/PostgreSQL lab.

2 Introduction

1. HTTP is a stateless protocol.

   (a) What does this mean?

   (b) What are the consequences?

2. Mechanisms for retaining state (persistence):

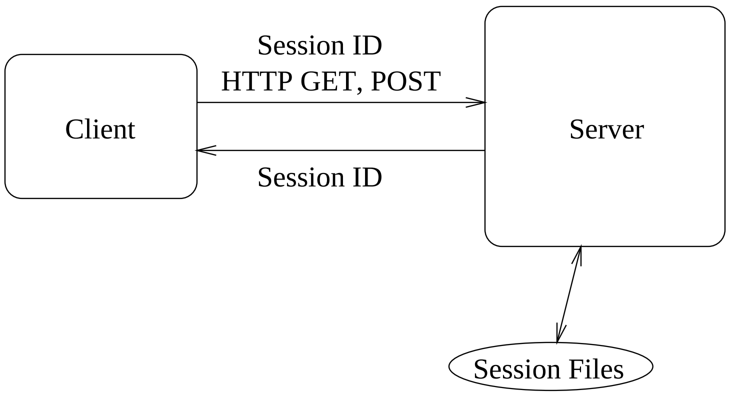
   (a) Hidden fields in forms.

   (b) Cookies.

   (c) Sessions.

   Advantages, disadvantages.

3. HTTP/PHP session information transfer model:



   (a) HTTP GET: parameters passed as part of URL:

```
http://phoenix.goucher.edu/process.php?name=tom
```

i. Accessed through `_GET` associative array in PHP:

```
$name = $_GET["name"];
```

ii. Session ID passed as GET parameter:

```
echo "<A href=\"http://phoenix.goucher.edu/process.php?"
     . SID . "\">";
```

(b) HTTP POST: parameters passed into script via `stdin`.

i. Accessed through `_POST` associative array.

(c) Session variables are maintained on the server and accessed by referring to a session ID and using the `_SESSION` associative array.

3 Sessions

1. Sessions exist until browser is closed or PHP garbage collector removes the session data file.

2. Establishing a session and writing session variables:

```
session_start();

$_SESSION["username"] = $username;
$_SESSION["password"] = $password;
```

   (a) `session_start()` and new/resumed sessions.

3. The session ID constant: `SID`.

4. Checking to see if a session variable already exists:

```
if (isset($_SESSION["username"])
   $username = $_SESSION["username"];
else
   $_SESSION["username"] = $username;
```

5. Deleting a session variable (enhanced security):

   ```
   unset($_SESSION["username"]);
   ```

   Also possible to delete entire session — see online docs.

6. Avoiding garbage collection:

   (a) Garbage collector invoked by *any* `session_start()`.

   (b) Session files older (mod time) than 24 minutes are reclaimed.

   (c) Avoiding garbage collection? Read/write a session variable.

4 PostgreSQL Connectivity

1. Processing model:

   (a) Establish connection, receive handle.

   (b) Send SQL query, receive results "array."

   (c) Process results array.

   (d) Free results array.

   (e) Repeat as needed.

   (f) Close connection.

2. Establishing a connection:

   ```
   $handle = pg_connect("dbname=databaseName user=userName password=pwd");
   ```

   Check handle status!! Why handles? (Script could have multiple DB connections open.)

3. Sending a query:

   ```
   $result = pg_exec($handle, "query string");
   ```

   Check result status!!

4. Determining the size of a result: `pg_numrows($result)`, `pg_numfields($result)`.

5. Accessing the result:

   ```
   $item = pg_result($result, $row, $field);
   $item = pg_result($result, $row, "fieldName");
   ```

   `$row` and `$field` are 0-based numeric indices. `fieldName` is an associative array-style index.

6. Freeing a result, closing a connection:

   ```
   pg_freeresult($result);
   pg_close($handle);
   ```

5 Example Code Walk-through

Refer to Class Materials section of course web site.

Things to note for each file:

1. `login.html`:

   (a) Form tag: method and action.

   (b) Input tags: types and names.

2. `authenticate.php`:

   (a) Debugging notes.

(b) Retrieval of username, password. Associative array.

(c) Database connection and error checking.

(d) Sending a query and error checking.

(e) Accessing query results. Associative array.

Why the check on `pg_numrows()`?

(f) Establishing the session and saving session variables.

(g) Passing `SID` back to the server as a GET parameter.

`SID` will be empty when we resume the session.

(h) Freeing the result and closing the database. Why?

3. `query.php`:

(a) Retrieving session variables.

(b) Iterating through the result.