PHP AND MYSQL
HUY NGUYEN (PHD)

MySQL

- Open Source database server
  - Runs on many platforms (Unix & Windows)
- Networked server – no fancy GUI like MS Access.
  - You can find *clients* that provide a GUI.
- Great for small to medium-sized applications

MySQL Installation/Configuration

- You install the server, and provide a *root* password.
- Now you need a client to do anything!
  - Create databases, view databases, etc.
- Windows MySQL server comes with a command-line client
  - You need to learn all the commands, and type them in manually…

phpMyAdmin

- A MySQL client written in PHP
- Via the web you can manage:
  - Manage Databases
  - Manage MySQL users
  - Submit queries (SQL)
- A great way to learn SQL!

Php – MySQL Support

- You need a version of PHP that includes the MySQL module
  - Included by default on most php distributions.
- Documentation of all mysql functions/objects is available via php.net.

Opening a MySQL database

```
$username=“fred”;  $password=“fred”;
$database=“eiw”;
mysql_connect("localhost",$username,$password);

@mysql_select_db($database) or die( "Unable to
   select database");
```

Assumes that the machine running the server is “localhost”

You can easily use a MySQL server that is running on a remote machine.

Submitting a query to the server

```
$query = "SELECT uid from users WHERE
  username = ‘fred’";

$res = mysql_query($query);

// no result - no user so return false

if (! $res) {
   … no result (error!)
}
```

Using the results of a query

- In ASP/ODBC we used a recordset object to access the results of a query.
- Mysql_query returns something similar – an object we can use to get at the rows of the result of the query.

Accessing

- **`mysql_numrows($res)`** : number of rows in the result.

- **`mysql_result($res,$index,$field)`**
  - Returns a single column value
  - $index is the row
  - $field is the field name (column name)

Example

```
$res = mysql_query(“SELECT * FROM users”);
if (! $res) { …handle error…}

$numrows = mysql_numrows($res)

for ($i=0;$i<$numrows;$i++) {
   $name = mysql_result($res,$i,”username”);
   $pass = mysql_result($res,$i,”password”);
   … Do something with $name and $password…
}
```

Sample Code

- Simple e-commerce site
  - Users login in (with just a user name)
  - View products (including search)
  - Add products to shopping cart
  - Remove products from shopping cart

Database Tables

- **`users`** – information about all the customers that use the system.
- **`products`** – information about all the products we sell.
- **`cartentries`** – shopping cart items (relates a user to a product)

Table: `users`

- **`uid`**: integer id number (autoincrement)
- **`firstname, lastname`**: strings. varchar(20)
- **`username`**: string – login name. varchar(20)
- **`email`**: string. varchar(30)

Table: `products`

- **`pid`**: integer id number (autoincrement)
- **`name`**: string – product name. varchar(30)
- **`price`**: floating point number.

Table: cartentries

- `uid`: integer user id number
- `pid`: integer product id number
- `quantity`: integer (# products).

Some Queries

- Get list of all products:
  - `“SELECT * FROM products”`
- Get list of all the entries in joe’s (user 22) shopping cart:
  - `“SELECT * FROM cartentries WHERE uid=22”`

- Check the actual code in the demo for more complex queries…

Sample Code: `main.php`

- `main.php`: the main program
  - takes care of the session (session variable `userid`)
  - Determines what the query is and takes appropriate action.
    - Many actions defined in other PHP files that are included using `require`
  - Generates the base HTML for the document (including a small “header”).

Database Code: `db.php`

- php functions that interact with the database. This file is always “required” by main.php.
- Creates connection to the mysql server.
- Functions `login`, `product_list`, `show_cart`, `add_to_cart`, `remove_from_cart` and some HTML generating functions.

`add.php`

- Called from `main.php` when user is adding an item to cart:
  - `require("add.php")`
- Takes care of the logic for adding an item to the shopping cart for current user.
  - Makes sure item exists.

`login.php`

- Called from `main.php` when user is trying to log in:
  - `require("login.php")`
- Takes care of the logic for login process:
  - Decides what to send back if valid/invalid login.

`logout.php`

- Called from `main.php` when user is trying to log out:
  - `require("logout.php")`
- Takes care of the logic for log out:
  - Terminates the session.

`plist.php`

- Called from `main.php` when user wants to see a list of products:
  - `require("plist.php")`
- Just calls `product_list` function provided by `db.php`

`remove.php`

- Called from `main.php` when user is trying to remove an item from shopping cart:
  - `require("remove.php")`
- Gets user id (from session) and product id (from HTTP query)
- Calls remove_from_cart
- Sends back resulting cart as HTML.

`search.php`

- Called from `main.php` when user is trying to search for products:
  - `require("search.php")`
- If a search query is found in the HTTP query, processes the search.
- If no search query found, sends back a form that can be used to submit a search.

`show.php`

- Called from `main.php` when user is trying to see their shopping cart in:
  - `require("show.php")`

- Just calls `show_cart` function found in `db.php`