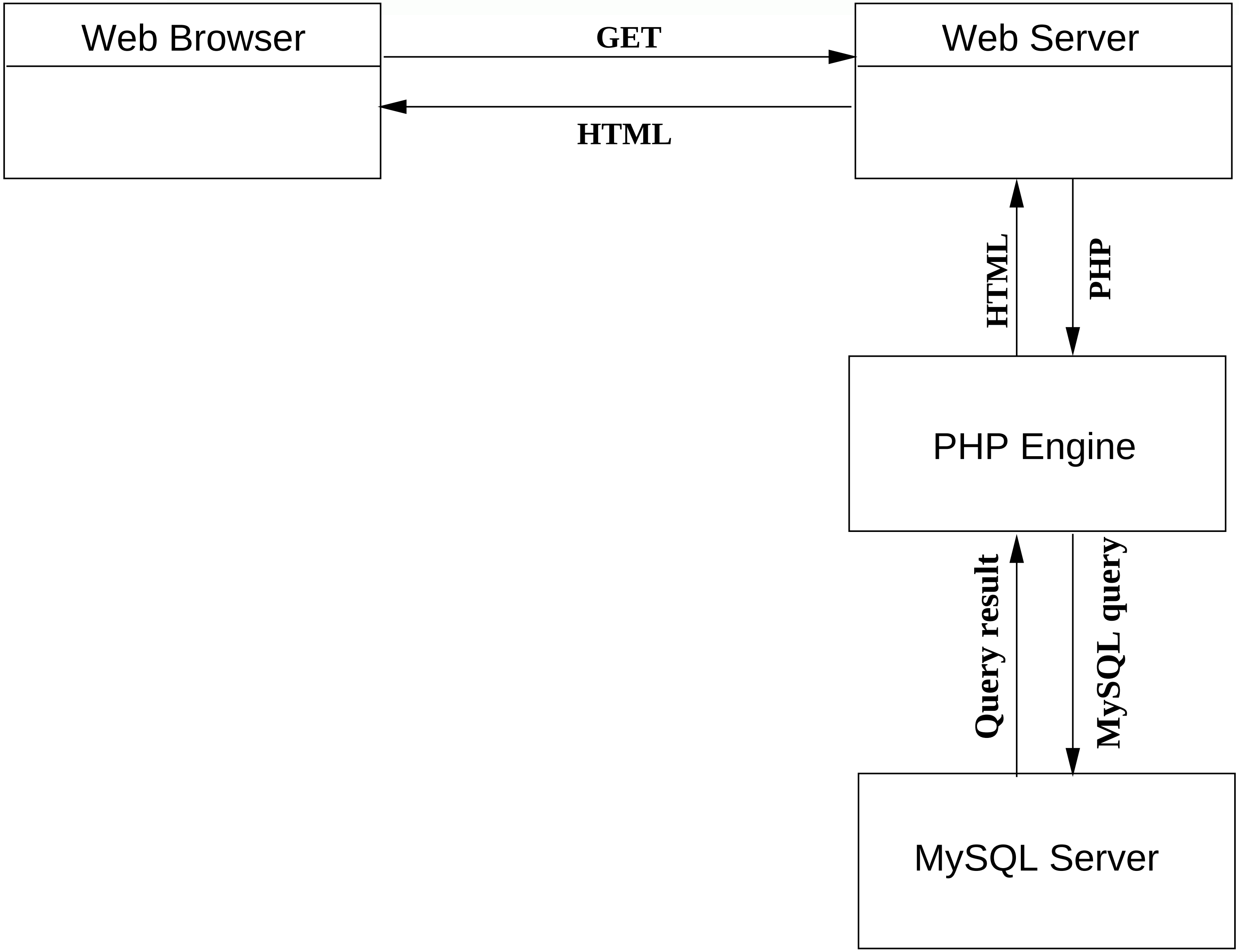
Universiteti i Prishtinës

Programimi në Internet

PHP and MySQL

Dr. Ing. Lule Ahmedi

Web Database Architecture



Server-Side vs. Client-Side Execution

When linking databases and Web pages, most of the time

- Server-side code
- Embedded code with server-side execution

Server-Side vs. Client-Side Execution

When linking databases and Web pages, most of the time

- Server-side code
- Embedded code with server-side execution

Why not client-side execution?

Server-Side vs. Client-Side Execution

When linking databases and Web pages, most of the time

- Server-side code
- Embedded code with server-side execution

Why not client-side execution?

- Your thoughts here..

Server-Side vs. Client-Side Execution

When linking databases and Web pages, most of the time

- Server-side code
- Embedded code with server-side execution

Why not client-side execution?

- Clients are usually too far away
- Servers should not trust the clients
- Servers are designated to serve applications that might well require a lot more horsepower
- **A disadvantage:**
  - Server might be overloaded while clients sit idle

Running Example: Bookstore Catalog

```
<html>
<head><title>Book-O-Rama Catalog Search</title></head>
<body><h1>Book-O-Rama Catalog Search</h1>
<!-- This is the catalog search form -->
<form action="results.php" method=post>
  Choose Search Type:<br/>
  <select name="searchtype">
    <option value="author">Author</option>
    <option value="title">Title</option>
    <option value="isbn">ISBN</option>
  </select></br>

  Enter Search Term:</br>
  <input type="text" name="searchterm"></br>
  <input type="submit" value="Search">
</form>
</body>
</html>
```

Running Example: Bookstore Catalog

```
<html>
<head><title>Book-O-Rama Search Results</title></head>
<body><h1>Book-O-Rama Search Results</h1>
<!-- This is the catalog search script -->
<?php
  // create short variable names
  $searchtype=$HTTP_POST_VARS['searchtype'];
  $searchterm=$HTTP_POST_VARS['searchterm'];

  /****************************************************/
  /**  Here should be the script for querying.
       We'll next discuss the basic steps as follows **/
  /****************************************************/
  // checking and filtering input data
  // establishing a connection
  // choosing a database
  // querying
  // retrieving the results
  // disconnecting
?>
</body></html>
```

Checking and Filtering Input Data

```
$searchterm = trim($searchterm);

if (!$searchtype || !$searchterm)
{
  echo ’You have not entered search details.’;
  echo ’Please go back and try again.’;
  exit;
}

$searchtype = addslashes($searchtype);
$searchterm = addslashes($searchterm);
```

- `trim()` - is used to strip any whitespace
- `$searchterm` - is checked although it comes from an HTML - sensible screen data
- `addslashes()` - when submitting user input to a database
- `stripslashes()` - when returning data from a database

Setting Up a Connection

```
@ $db = mysql_pconnect('localhost','bookorama','bookorama12');

if (!$db)
{
  echo 'Error: Could not connect to database.';
  echo 'Please try again later.';
  exit;
}
```

- resource mysql_pconnect([string *host* [:*port*] [:/*socketpath*] [,
  string *user* [, string *password*]]]);
- mysql_connect() - a nonpersistent connection, closes
  - when a script finishes execution, or
  - through the mysql_close() function call
- Limited # of connections that can exist at the same time

Why use persisten connections? To save time and server overhead.

Choosing a Database to Use

```
mysql_select_db('books');
```

- bool mysql_select_db(string *database*, [resource *database_connection*]);

Querying the Database

```
$query =
 "select * from books
  where ".$searchtype." like ’%".$searchterm."%’";

$result = mysql_query($query);
```

- resource mysql_query(string *query*, [resource *database_connection*]);

Retrieving the Query Results

```
$num_results = mysql_num_rows($result);

echo ’<p>Number of books found: ’.$num_results.’</p>;

for ($i=0; $i < $num_results; $i++)
{
  $row = mysql_fetch_array($result);
  echo ’<p><strong>’.($i+1).’. Title: ’;
  echo htmlspecialchars(stripslashes($row[’title’]));
  echo ’</strong><br/>Author: ’;
  echo stripslashes($row[’author’]);
  echo ’<br/>ISBN: ’;
  echo stripslashes($row[’isbn’]);
  echo ’<br/>Price: ’;
  echo stripslashes($row[’price’]);
  echo ’</p>’;
}
```

- `htmlspecialchars()` - serves to encode characters with special meanings in HTML, like ’&’, ’<’, ’>’, ’"’, etc.

Disconnecting From the Database

- `mysql_close(`*database_connection*`);`

Inserting Data Into the Database

```
<html><head><title>Book-O-Rama - New Book Entry</title></head>
<body><h1>Book-O-Rama - New Book Entry<h1/>
<form action="insert_book.php" method=post>
<table border="0">
<tr>
  <td>ISBN</td>
  <td><input type="text"
    name="isbn" size="13" maxlength="13"><br/></td></tr>
<tr>
  <td>Author</td>
  <td><input type="text"
    name="author" size="30" maxlength="30"><br/></td></tr>
<tr>
  <td>Title</td>
  <td><input type="text"
    name="title" size="30" maxlength="60"><br/></td></tr>
<tr>
  <td>Price</td>
  <td><input type="text"
    name="price" size="7" maxlength="7"><br/></td></tr>
<tr>
  <td colspan="2"><input type="submit"
    value="Register"></td></tr>
</table></form></body></html>
```

Inserting Data Into the Database (cont.)

```
 ..
$isbn = addslashes($isbn);
 ..
$price = doubleval($price);

 ..

$query = "insert into books values
  (’".$isbn."’, ’".$author."’, ’".$title."’, ’".$price."’)";

$result = mysql_query($query);

if ($result)
  echo mysql_affected_rows().’ book inserted into database.’;
 ..
```

Other Useful PHP–MySQL Functions (cont.)

Freeing up resources:

- `mysql_free_result(resource` *result*`);`
- Example: `mysql_free_result($result);`

Creating and deleting databases:

- `bool mysql_create_db(string` *database*`, [resource` *database_connection*`]);`
- `bool mysql_drop_db(string` *database*`, [resource` *database_connection*`]);`

Other PHP-Database Interfaces

- A set of libraries to connect to Oracle, PostgreSQL, Informix, Microsoft SQL, Sybase, etc.
  - They all share common principles of connecting and querying
  - Differ slightly on functionality
- If there is no support, i.e. a specific library available in PHP for a given database, use the generic ODBC functions
  - Open Database Connectivity - a standard for connections to databases
  - Limited functionality set due to being designated to work with everything

Using PEAR DB - A Generic Database Interface

PHP offers also database abstraction classes such as Metabase or PEAR::DB

- They allow you to use the same function names for each different type of database
- The PEAR::DB abstraction layer is the core component of PEAR

The difference is basically syntactic, e.g.,

- To connect:
  ```
   $db = DB::connect($dsn, true);
  ```
  where $dsn = "mysql://$user:$pass$host/$db_name";
- To retrieve result rows:
  ```
  $row = $result->fetchRow(DB_FETCHMODE_ASSOC);
  ```