Developing Web Applications

HTML, PHP and SQL

Orlando Karam
okaram@spsu.edu

SOUTHERN POLYTECHNIC STATE UNIVERSITY

Our Web applications

- Use HTML to communicate with the browser
- PHP to generate that HTML and
- SQL to perform database acctions

Web Basics

- On the client, you use a Web Browser which requests a page from a web server
- The browser uses a URL to identify the resource.
- URL
  - protocol://server/page?param1=value1;p2=v2
  - normally, protocol is http so
  - http://okaram.spsu.edu/curri/abc.php
  - URL may have other fields (port, pwd etc)

Handling web requests

SPSU

- Conceptually, the server reads a file that corresponds to the URL and sends that file to the browser
- In reality, it may be *running a program* that corresponds to the URL
- If the URL has *?var=val* at the end, those will be passed to the program as arguments
- The request may also contain other name=value pairs as part of a POST request

Our setup

- http://okaram.spsu.edu/~yourusername maps to your public_html folder
- if the url's file name ends in .php the server will pass it through the PHP interpreter
  - but only if you're using a web browser, not if accessing the file through sftp
- A PHP file is mostly HTML, but the pieces of PHP code get executed and their *output* is inserted in the file instead of the PHP code
- The web browser gets only HTML, never sees the PHP code

Basic HTML

- HTML is basically text, with special markers
- We use *tags* to mark the text and provide document structure
- A tag opens with <tag> and closes with </tag>
  - –ie, <html></html>,   etc
  - –the contents within the tag gets modified by the tag
- We can nest tags
- Tags may have parameters
  - –parameters go inside the <>, name=value
  - –ie <img src=”logo.jpg”>

More HTML

- HTML derives from SGML
- Later, XML came which simplifies the model
- Latest versions of HTML are based on XML
  - –means we need to add a preamble marking the doc as xml
  - –we need to close all our tags, or do <tag/> for tags that do not need closing
- Old way doesn't need preamble etc
  - –That's what I do, since it's easier :)
- Fight between specifying appearance vs document structure

Basic Document Structure

```
<html>
  <head>
    <title> xxx </title>
  </head>
  <body>
  contents go here
  may use other tags
  </body>
</html>
```

```
<html>
<head>
    <title> blah </title>
 </head>
<body>
<p>stuff here</p>
<a href="http://a.com">
link</a>
 <img src="fig.png">
</body>
</html>
```

Basic tags

- Paragraphs, <p>
  - –not required in old style, but works :)
- Appearance
  - –bold, <b>
  - –italics <i>
  - –center
- Document Structure
  - –Headings, <h1>, <h2> ...
  - –Lists <ol>, <ul>,
- Images <img src=””>
- Hyperlinks <a href=””>

HTML Tables

- <table>
  - –<table border=”2”
  - –<tr> table row
  - –<th> table header
  - –<td> table data (one cell)
- Example
  - –<table>
    - <tr> <th> field1</th> </tr>
    - <tr> <td> value1</td> </tr>
    - <tr> <td> value1</td> </tr>
  - –</table>

HTML Forms

- Forms allow us to send input to web pages (apps)
- in HTML we define <form action=””>
  - –the action defines which program or page gets called when submitting
- We specify <inputs> within the form
  - –text, password, hidden,**submit**
  - –ie <input name=”age” type=”text”>
  - –check references for more input types :)

PHP

- Developed for web apps
- Embed PHP code within HTML
  - –tags <?php ?> delimit php code
  - –output of php code gets inserted in the doc instead of the code
  - –the browser sees the *output* not the php code
- Web server needs to execute the PHP files
- Syntax is similar to C

Basic syntax

- Dynamic typing
  - –uses $ to denote variables, so $i=3 or $a=$b+$c
  - –no need to declare variables
- Arrays
  - –$arr=array(); creates new array
  - –$arr[0]=5 array access
- Maps/Dictionaries
  - –Like arrays, but indices can be strings etc
  - –$map[‘joe’] careful of your word processor :)

String interpolation

- When referencing a variable inside a string, its value gets inserted
  - –print “hello $name”
  - –doesn’t work within single quotes ‘hello $name’
- Remember, you can escape characters \”

```
$link=”http://spsu.edu”
print “<a href=\”$link\”>$link</a>
```

Control structures

- Same as C, but $ before variables
- if($age>10) ...
- for($i=0; $i<10; ++$i) ...
- foreach iterates over collections (arrays)
  - $a=array(10,20,30);
  - foreach($a as $elem)
    - print $elem;
- Can define functions
  - function foo($a,$b)
- Also OO features, but we won't cover those
  - except use -> to access field, $obj->age

Simple Example

```
<html>
 <head> <title> Simple PHP Program</title> </head>
<body>
<?php

$age=22;
$name="Clay";
if($age>21)
    print("$name, you can legally drink alcohol<br>\n");
for($i=0; $i<10; ++$i)
    print("hello $name<br>\n");
?>
</body>
</html>
```

PHP and Forms

- Can create HTML forms that will pass your input to PHP programs
- Easier to have two separate files, an HTML file for the form and a PHP file
- In the PHP code, can use the pseudo-variables $_GET, $_POST and $_REQUEST
  - they are maps, so do $_GET['age']
  - need to map to the names of the inputs in your HTML forms

Sample form

```
<html>
   <head>   <title>my form</title>    </head>
<body>
 <form action="simple-form.php">
  Your name: <input type="text" name="name"><p>
  Your Age: <input type="text" name="age"><p>
  How many times ? <input type="text" name="times"><p>
<input type="submit">
</form>
</body>
</html>
```

Sample form

```
<html>
<head><title> Simple PHP Program</title> </head>
<body>
<?php
    $age=$_GET['age'];
    $name=$_REQUEST['name'];
    $n=$_REQUEST['times'];

function msgCanDrink($name,$age)
{
    if($age>21)
        return "$name, you can legally drink <br>\n";
    else
        return"Sorry $name,no alcohol for you<br>\n";
}
    print(msgCanDrink($name,$age));

    for($i=0; $i<$n; ++$i) {
        print("hello $name<br>\n");
    }
?>
</body>
</html>
```

Connecting to the database

- You'd write PHP code that executes SQL statements
- Within PHP, your SQL statement is just a string
- We'll use the specific postgresql (pg_) functions
  - there are abstraction layers, but ...
  - pg_connect connects to the database
  - pg_query, pg_query_params execute queries
    - insert etc are also queries here :)
  - pg_numrows tells you how many rows were returned
  - pg_fetch_object, pg_fetch_* fetches one specific row

Display all values in table

```
<html>
<head>
   <title> Forms </title>
</head>
<body>

<?php
      $conn=pg_connect("host=localhost dbname=test user=curri password=1qaz2sx");
      $query="select * from student1";
      $res=pg_query($conn,$query);
      echo "<table border=\"2\">\n";
      echo "<tr><th>Name</th> <th>Age</th></tr>\n";

      for($i=0; $i<pg_numrows($res);++$i) {
            $obj=pg_fetch_object($res,$i);
            echo "<tr><td>$obj->name</td><td>$obj->age</td></tr>\n";
      }
      echo "</table>";
?>

</body>
</html>
```

Now with inputs

- Normally, you want to access the database based on user input
- Need HTML form for the user input
- Need to use pg_query_params in PHP
  - executes a query with parameters
  - avoids SQL injections
  - uses $1 $2 etc in the query
  - your code passes an array to pg_query params with corresponding values

Inserting (form)

```
<html>
<head>
  <title>my form</title>
</head>

<body>
<form action="db_insert.php">
Id: <input type="text" name="id">
Your name: <input type="text" name="name"><p>
Your Age: <input type="text" name="age"><p>

<input type="submit">
</form>
</body>
</html>
```

Inserting (php)

```
<html>
<head>
  <title>my form</title>
</head>
<body>
<?php
    $id=$_GET['id'];
    $name=$_GET['name'];
    $age=$_GET['age'];
    $conn=pg_connect("host=localhost dbname=test user=curri password=1qaz2sx");
    $query="INSERT INTO Student1(id,Name,Age) VALUES ($1,$2,$3)";
    pg_query_params($query,array($id,$name,$age));
?>
</body>
</html>
```