CS 50: Introduction to Computer Science I
Harvard College
Fall 2007

Scribe Notes
Week 10 Friday
Anjuli Kannan

Getting User Input (0:00-8:00)

- In froshims.html, we write encode a webpage that allows freshmen to sign up for intramural sports.
- Let's examine the following excerpt contained in the body:

```
<form action="register.php" method="post">
  <table border="0" style="text-align: left;">
    <tr>
      <td>Name:</td>
      <td><input name="name" type="text" /></td>
    </tr>
    <tr>
      <td>Captain:</td>
      <td><input name="captain" type="checkbox" /></td>
    </tr>
    <tr>
      <td>Gender:</td>
      <td><input name="gender" type="radio" value="F" /> F
          <input name="gender" type="radio" value="M" /> M
</td>

    </tr>
    <tr>
      <td>Dorm:</td>
      <td>
        <select name="dorm" size="1">
          <option value=""></option>
          <option value="Apley Court">Apley Court</option>
          <option value="Canaday">Canaday</option>
          <option value="Grays">Grays</option>
          <option value="Greenough">Greenough</option>
          <option value="Hollis">Hollis</option>
          <option value="Holworthy">Holworthy</option>
          <option value="Hurlbut">Hurlbut</option>
          <option value="Lionel">Lionel</option>
          <option value="Matthews">Matthews</option>
          <option value="Mower">Mower</option>
          <option value="Pennypacker">Pennypacker</option>
          <option value="Stoughton">Stoughton</option>
          <option value="Straus">Straus</option>
          <option value="Thayer">Thayer</option>
          <option value="Weld">Weld</option>
          <option value="Wigglesworth">Wigglesworth
    </option>
        </select>
      </td>
    </tr>
  </table>
  <br /><br />
  <input type="submit" value="Register!" />
</form>
```

- First, notice that we create a form object, as indicated by the <form> tag, to contain all fields that will get input form the user

- Inside the <form> tag we use the action attribute to indicate where all the data should be sent to when submitted
- In this case, it is register.php, the page that contains code to process all the data
- Since there is no indication of what directory register.php is in, we (and the browser) will assume it is in the same directory as froshims.html
- The other attribute that we put in the <form> tag is the method of data submission, post
- In the past, we used get, which put the submitted data into the web address (e.g., http://www.google.com/search?q=harvard when we search for harvard)
- But this can be a problem (a) if we submit a lot of data (like a 3 MB image) and (b) if we have some private data that we don't want displayed in the web address (like a password)
- So post is another way of submitting data in which it does not get sent by means of the address
- To layout the components of the form nicely, we draw a table with a border of size 0 (so that it's invisible)
- Each new row of the table is marked by the <tr> tag, while cells within each row are marked by <td>
- Now, how do we actually get data from the user? All objects that will fetch some data (text box, check box, radio button, drop-down menu) are marked by the <input> tag
- Inside the tag we use the the name attribute to give the data a sort of variable name by which we will refer to it later, in register.php
- We use the type attribute to indicate whether the object is a button, checkbox, etc.

Accessing User Input (8:00-17:00)

- Now let's take a look at register.php:

```
#!/home/c/s/cs50/pub/local/i386/php/bin/php-cgi
<?

// validate submission
if ($_REQUEST["name"] == "" ||
$_REQUEST["gender"] == "" ||
$_REQUEST["dorm"] == "")
{
header("Location: http://www.courses.fas.harvard.edu/~cs50/lectures/weeks/10/src/froshims.html");
exit;
}

?>

<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
<title>Registered!</title>
</head>
<body>
You are registered! (Well, not really.)
</body>
</html>
```

- You may notice that it contains a lot of XHTML. This is because even when we write a page in PHP, XHTML is still used to take care of visual design and formatting. PHP just does the behind-the-scenes computations to generate the content.
- PHP is an interpeted language, as opposed to C, which is a compiled language. This means that its instructions can be read and performed on the spot, without compiling to an executable.
- Since PHP is an interpreted language , a little bit of work has to be done when this page is loaded. The line at the top indicates where to find the interpreter, a program that will read the PHP and carry out the computations it describes
- We use the <? ... ?> to embed PHP so that the interpreter knows where it needs to do some work
- Let's take a look at the PHP contained on this page.
- $_REQUEST is an associative array (hash table) that contains all data fetched by both the get and post methods. So when we say $_REQUEST["name"] we are looking up the string "name" in the array of submitted data.
- If we had typed, say, "malan" on the froshims.html page, then calling $_REQUEST["name"] would fetch the string "malan"
- All this PHP code does is checks to see that some data was submitted in each of the fields. If any are the empty string, it wil send the browser back to froshims.html. Otherwise, it will proceed to display the page generated by the XHTML.

Problem Set 7 Preview (17:00-23:00)

- In this problem set, you will write an interface for users to buy and sell stock
- Here we will look at a version already implemented by the staff so that you can see how yours should behave when you are done
- The first thing we can do with it is make a new username and password
- Once we have logged in, we can get a quote of any stock by typing in its abbreviation (the data is fetched from Google Finance)
- Finally, we can buy or sell any number of shares of any stock using some fake money that starts off at $10,000
- Any time we buy or sell, the underlying code must also check to make sure we have enough money/shares to perform the desired transaction
- Once you have implemented this interface, you can use it to maintain your portfolio and see how you stack up on the Bigger Board
- But how are you going to keep track of users, passwords, balances, and portfolios?

Setting Up a Login Page (23:00-32:30)

- You will use SQL to store and query your data
- We will demonstrate the basics of SQL by writing a login page for cs50stud
- First, we copy index.html into a new file login.php
- This is an easy way to avoid retyping (a) those things at the top that always have to be there and (b) basic things like head, body, etc. that we will always need
- Now when we go to /cs50stud/login.php in Firefox, we get the exact same thing we saw before. But this is a PHP file. Why isn't it any different?
- The fact that it is a PHP file just means that the interpreter (specified at the top) is going to go through it and perform any necessary computations before the browser displays it
- But HTML is spit out "raw" by the interpreter, so ultimately we end up with the exact same page as index.html, and Firefox displays it just the same.
- Now, we go back to vim and make some changes. We add a form like from froshims.html that will allow the user to enter a username and password and then send it to login2.php:
- Next, open up login2.php (or just copy everything over again) and, just to make sure we can safely arrive at the page, print something simple like "You've logged in"
- We go back to Firefox, visit login.php, and enter the username "malan" and password "12345"
- Upon clicking submit, we are taken successfully to login2.php. Now that we know our form button is working, we can proceed to implement some logic.
- We go back to vim and change login2.php so that it (a) fetches the username and password using $_GET and (b) prints "Success!" if they are malan and 12345
- Then we go back to Firefox and make sure this works. Now we know that our form is correctly submitting data and login2.php is correctly fetching it.
- Notice how we test incrementally. This is good programming practice.
- But now suppose our database has more than just one user. Does it make sense to hardcode into login2.php all the possible username/password combinations and check them whenever someone tries to log in?
- No. Instead, we will store them in a table. In particular, we still store them in a SQL table.

SQL: Structured Query Language (32:30-45:00)

- SQL allows you to make and interact with a relational database
- A database is composed of tables
- SQL tables consist of rows and columns. The columns are data fields, such as username and password. The rows represent individual objects or entities, such as users.
- You may interact with SQL using the interpreter or using a GUI called myphpadmin
- The problem set will walk you through how to do either of these.
- For now, we log in to myphpadmin and create a new table to keep track of users
- We have just two columns: username and password
- For each of these, we can fill out many different pieces of information on the table creation screen, but we will only concern ourselves with name, length, type, and not null
- For instance, for username, we fill out name as "username", length as 255, type as

VARCHAR, and not null

- We also indicate that username will be the primary key, meaning that it will be the field used to identify any particular row.
- Now we go back and open our login file, which is no longer login2.php, but one prepared by David in advance (and given to you on this problem set)
- The first thing we do in this file is require the common code:

```
    require_once("inc/common.inc");
```

- Then we fetch the data entered on the previous page and use escape to ensure that if they enter code into the text box, it will not get inserted into our code:

```
 $username = mysql_real_escape_string($_POST["username"]);
 $password = mysql_real_escape_string($_POST["password"]);
```

- Next, we write a query for the database. A query is just some instruction about what data to fetch. In this query, we ask the database to look up the username and password that the user entered in the table users.
- This query gets put into the variable $sql using sprintf()

```
 $sql = sprintf("SELECT uid FROM users WHERE username='%s' AND
           password='%s'",$username, $password);
```

- We execute the query, and now $result contains a table with all the rows that matched the query we entered:

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

- If there is one and only row, we have found a match for the username and password, so we can redirect the user to his portfolio.

```
  if (mysql_num_rows($result) == 1)
    {
        // grab row
        $row = mysql_fetch_array($result);

        // cache uid in session
        $_SESSION["uid"] = $row["uid"];

        // redirect to portfolio
        redirect("index.php");
    }
```

- Otherwise, go back to the login page
- You can see the finished project by accessing your problem set code, where all of this is already written for you!