COP 4610L: Applications in the Enterprise
Fall 2006

Introduction to PHP – Part 2

Instructor : Mark Llewellyn
markl@cs.ucf.edu
ENG3 236, 407-823-2790
http://www.cs.ucf.edu/courses/cop4610/fall2006

School of Electrical Engineering and Computer Science
University of Central Florida

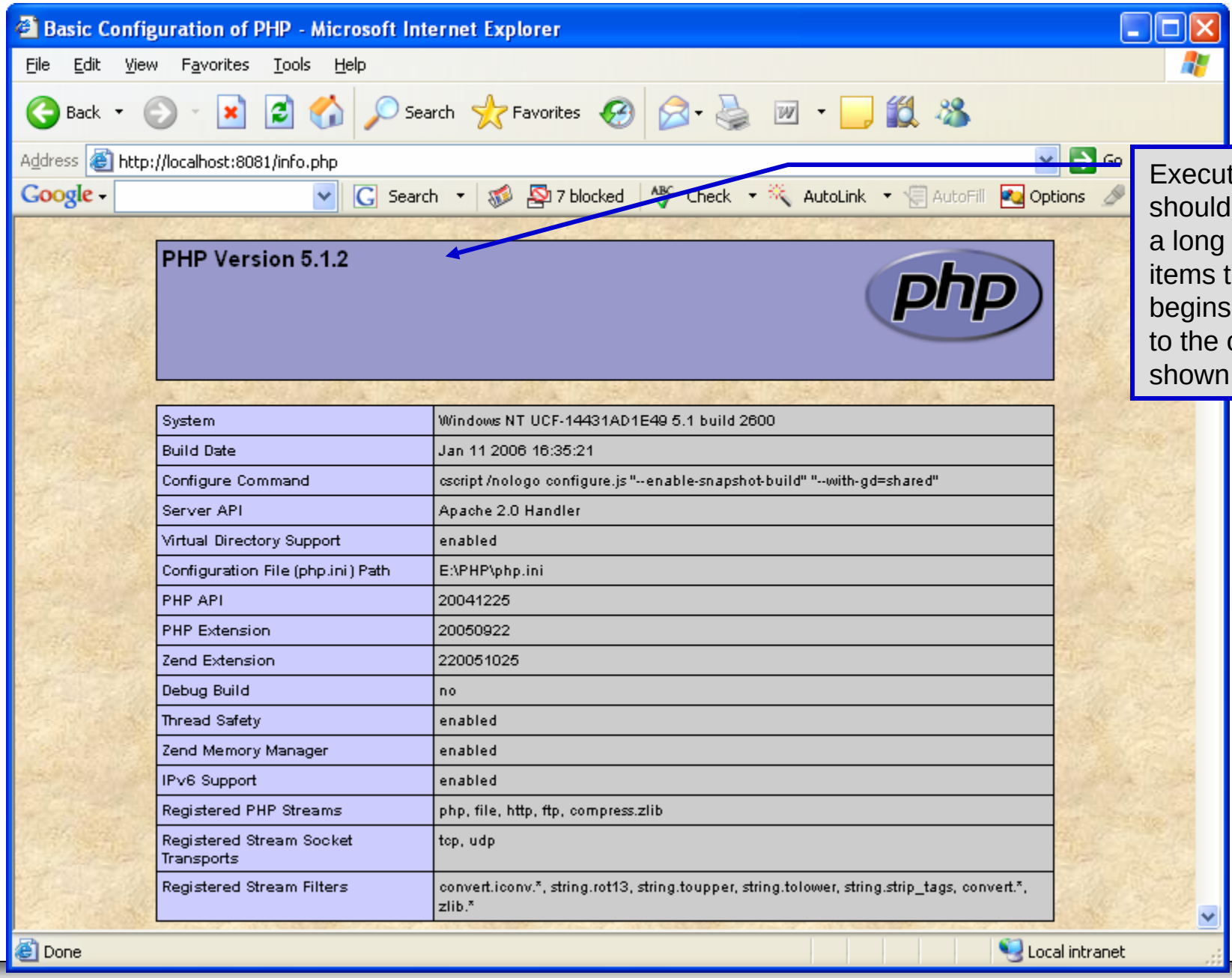
Checking Your PHP Set-up

- Once you get your web server (Apache) and PHP installed, the simplest way to test your installation is to create a PHP file and execute it.
- Create a PHP file containing the following single line:

```
<?php  phpinfo()  ?>
```

- Save this file in the `htdocs` folder in Apache (there will already be some files in this folder).
- Start the Apache server running and then access the PHP file through the browser with the following url:

http://localhost:8081/info.php



Execution should produce a long list of items that begins similar to the one shown.

Verifying a Username and Password Using PHP

- It is often the case that a private website is created which is accessible only to certain individuals.
- Implementing privacy generally involves username and password verification.
- In the next example, we'll see an XHTML form that queries a user for a username and password. The fields USERNAME and PASSWORD are posted to the PHP script `verify.php` for verification.
  - For simplicity, data is not encrypted before sending it to the server.
  - For more information on PHP encryption functions visit: http://www.php.net/manual/en/ref.mcrypt.php.

password.html – page 1

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

<!-- password.html                    -->
<!-- XHTML form sent to password.php for verification -->

<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title>Verifying a username and a password.</title>
    <style type = "text/css">
      td { background-color: #DDDDDD }
    </style>
  </head>
  <body style = "font-family: arial">
    <p style = "font-size: 18pt">
    <font color=red><B> Welcome to the COP 4610 High Security WebPage </B></font><HR>
    <p style = "font-size: 13pt">
      Type in your username and password below.
      <br />
      <span style = "color: #0000FF; font-size: 10pt;
        font-weight: bold">
        Note that password will be sent as plain text - encryption not used in this application
      </span>
    </p>
```


password.html – page 2

```
<!-- post form data to password.php -->
<form action = "password.php" method = "post">
  <br />
    <table border = "3" cellspacing = "3"  style = "height: 90px; width: 150px;
    font-size: 10pt" cellpadding = "1">
    <tr>
      <td colspan = "3">  <strong>Username:</strong>  </td>
    </tr>
    <tr>
      <td colspan = "3"> <input size = "40" name = "USERNAME"
          style = "height: 22px; width: 115px" />            </td>
    </tr>
    <tr>
      <td colspan = "3">  <strong>Password:</strong>    </td>
    </tr>
    <tr>
      <td colspan = "3">  <input size = "40" name = "PASSWORD"
          style = "height: 22px; width: 115px"  type = "password" /> <br/></td>
    </tr>
 <tr>
      <td colspan = "1">
        <input type = "submit" name = "Enter"  value = "Enter" style = "height: 23px;
          width: 47px" />       </td>
      <td colspan = "2">  <input type = "submit" name = "NewUser"   value = "New User"
          style = "height: 23px" />
      </td>
    </tr>
 </table>     </form>     <HR>   </body>    </html>
```


password.php – page 1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- password.php      -->
<!-- Searching a database for usernames and passwords. -->

<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <?php
      extract( $_POST );
      // check if user has left USERNAME or PASSWORD field blank
      if ( !$USERNAME || !$PASSWORD ) {
        fieldsBlank();
        die();
      }
      // check if the New User button was clicked
      if ( isset( $NewUser ) ) {
        // open password.txt for writing using append mode
        if ( !( $file = fopen( "password.txt", "a" ) ) ) {

          // print error message and terminate script
          // execution if file cannot be opened
          print( "<title>Error</title></head><body>
            Could not open password file
            </body></html>" );
          die();
        }
```


password.php – page 2

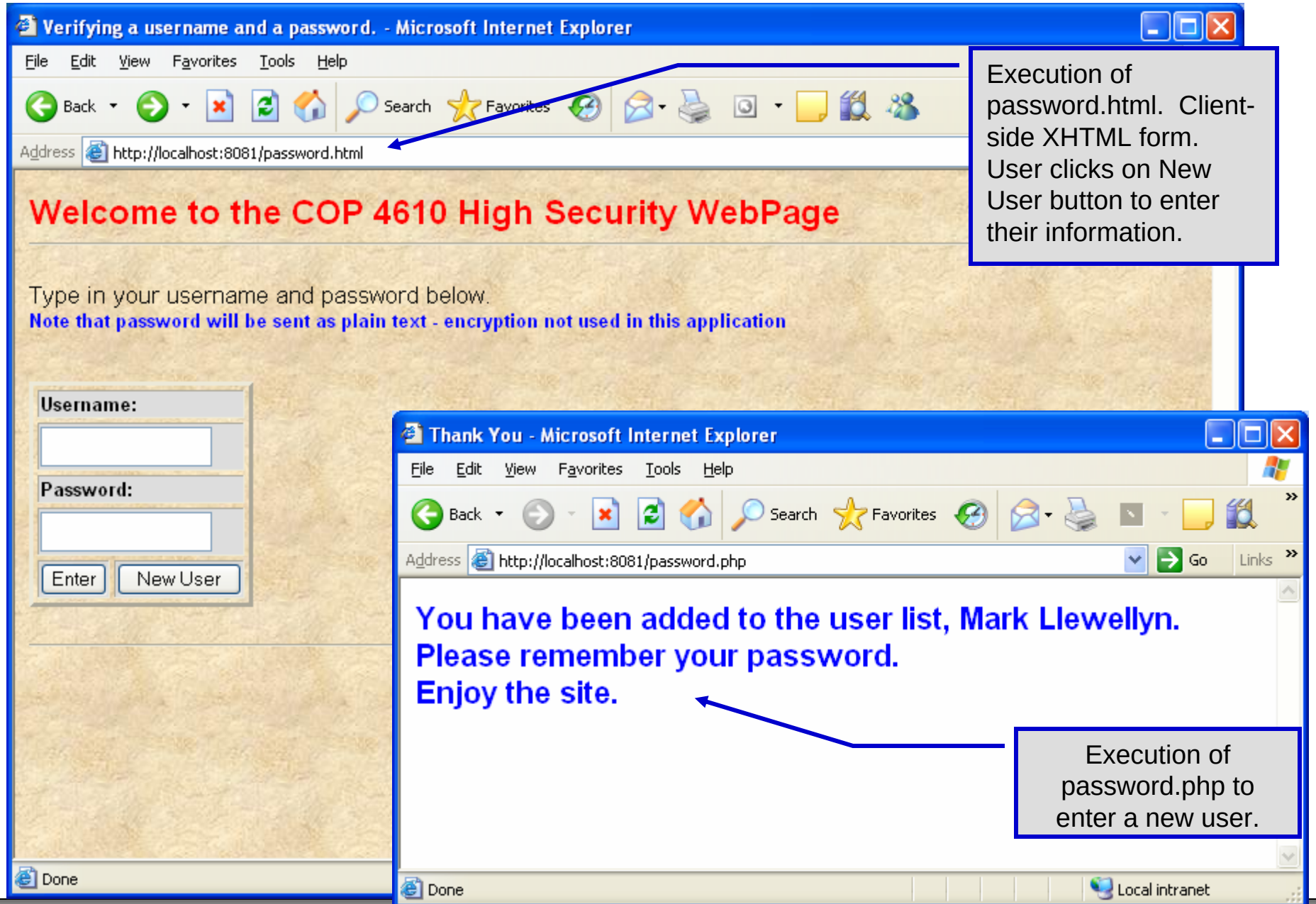
```
   // write username and password to file and call function userAdded
   fputs( $file, "$USERNAME,$PASSWORD\n" );
   userAdded( $USERNAME );
}
else {

   // if a new user is not being added, open file
   // for reading
   if ( !( $file = fopen( "password.txt",  "r" ) ) ) {
      print( "<title>Error</title></head>
         <body>Could not open password file
         </body></html>" );
      die();
   }

   $userVerified = 0;

   // read each line in file and check username and password
   while ( !feof( $file ) && !$userVerified ) {

      // read line from file
      $line = fgets( $file, 255 );

      // remove newline character from end of line
      $line = chop( $line );

      // split username and password using comma delimited string
      $field = split( ",", $line, 2 );
```

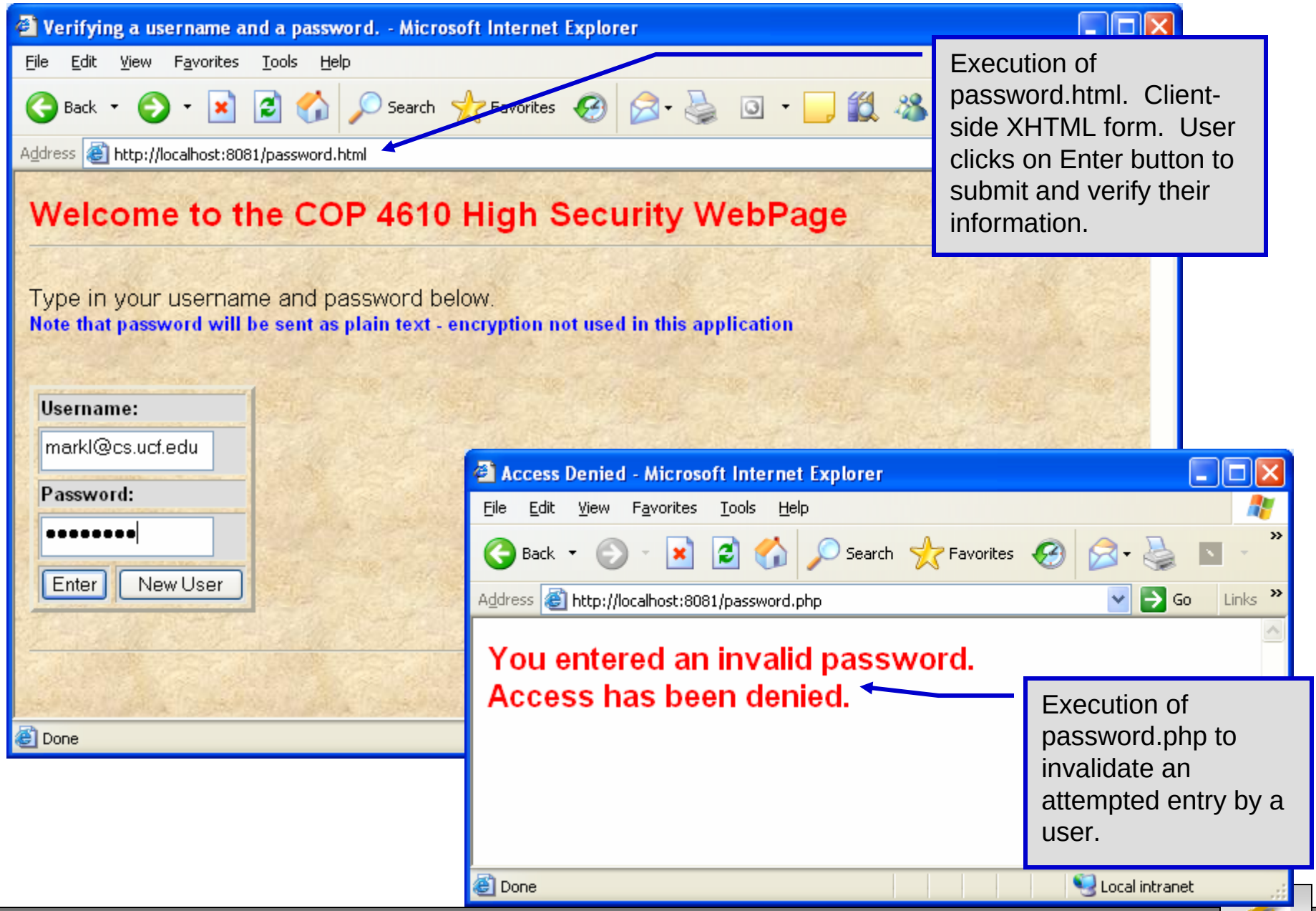

```
      // verify username
      if ( $USERNAME == $field[ 0 ] ) {
         $userVerified = 1;

         // call function checkPassword to verify user's password
         if ( checkPassword( $PASSWORD, $field ) == true )
            accessGranted( $USERNAME );
         else
            wrongPassword();
      }
   }

   // close text file
   fclose( $file );

   // call function accessDenied if username has not been verified
   if ( !$userVerified )
      accessDenied();
}

 // verify user password and return a boolean
function checkPassword( $userpassword, $filedata )
{
   if ( $userpassword == $filedata[ 1 ] )
      return true;
   else
      return false;
}
```


password.php – page 4

```
// print a message indicating the user has been added
function userAdded( $name )  {
   print( "<title>Thank You</title></head>
      <body style = \"font-family: arial;
      font-size: 1em; color: blue\">
      <strong>You have been added
      to the user list, $name.  Please remember your password.
      <br />Enjoy the site.</strong>" );
}

 // print a message indicating permission has been granted
function accessGranted( $name )  {
   print( "<title>Thank You</title></head>
      <body style = \"font-family: arial;
      font-size: 1em; color: blue\">
      <strong>Permission has been
      granted, $name. <br />
      Enjoy the site.</strong>" );
}
// print a message indicating password is invalid
function wrongPassword()   {
   print( "<title>Access Denied</title></head>
      <body style = \"font-family: arial;
      font-size: 1em; color: red\">
      <strong>You entered an invalid
      password.<br />Access has
      been denied.</strong>" );
}
```

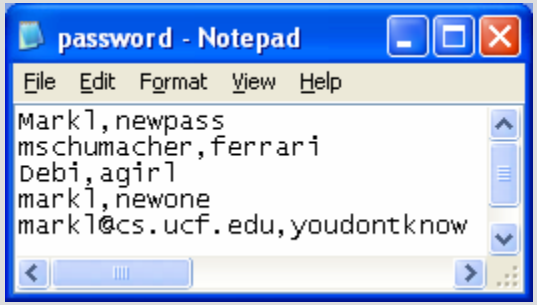

```
// print a message indicating access has been denied
        function accessDenied()   {
           print( "<title>Access Denied</title></head>
              <body style = \"font-family: arial;
              font-size: 1em; color: red\">
              <strong>
              You were denied access to this server.
              <br /></strong>" );
        }

         // print a message indicating that fields
         // have been left blank
        function fieldsBlank()   {
           print( "<title>Access Denied</title></head>
              <body style = \"font-family: arial;
              font-size: 1em; color: red\">
              <strong>
              Please fill in all form fields.
              <br /></strong>" );
        }
     ?>
  </body>
</html>
```



Verifying a username and a password. - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8081/password.html
Welcome to the COP 4610 High Security WebPage
Type in your username and password below.
Note that password will be sent as plain text - encryption not used in this application
Username:
Password:
Enter
New User
Done
Execution of password.html. Client-side XHTML form. User clicks on New User button to enter their information.
Thank You - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8081/password.php
Go
Links
You have been added to the user list, Mark Llewellyn.
Please remember your password.
Enjoy the site.
Done
Local intranet
Execution of password.php to enter a new user.


Verifying a username and a password. - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8081/password.html
Welcome to the COP 4610 High Security WebPage
Type in your username and password below.
Note that password will be sent as plain text - encryption not used in this application
Username:
markl@cs.ucf.edu
Password:
Enter
New User
Done
Execution of password.html. Client-side XHTML form. User clicks on Enter button to submit and verify their information.
Access Denied - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8081/password.php
Go
Links
You entered an invalid password.
Access has been denied.
Done
Local intranet
Execution of password.php to invalidate an attempted entry by a user.

How password.php Works

- The PHP script `password.php` verifies the client's username and password by querying a database. For this example, the "database" of usernames and passwords is just a text file (for simplicity). Existing users are validated against this file, and new users are appended to it.



The password.txt "database"

- Whether we are dealing with a new user is determined by calling function `isset` to test if variable `$NewUser` has been set.
- When the user submits the password.html form to the server, they click either Enter or New User button. After calling function extract, either variable `$NewUser` or `$Enter` is created depending on which button was selected. If `$NewUser` has not been set, we assume the user clicked Enter.

PHP and Database Connectivity

- PHP offers built-in support for a wide variety of database systems from Unix DBM through relational systems such as MySQL to full size commercial systems like Oracle.
- We'll continue to use MySQL as the underlying database system so that you can easily compare the work we've done with MySQL using Java servlets and JSPs.
- Before you go any further in these notes you must configure PHP to access MySQL databases. Beginning with PHP 5, MySQL is not enabled by default in PHP, nor is the MySQL library bundled with PHP.
  - Versions of MySQL greater than 4.1.0 use MySQLi extensions.
  - Versions of MySQL less than 4.1.0 use MySQL extensions.

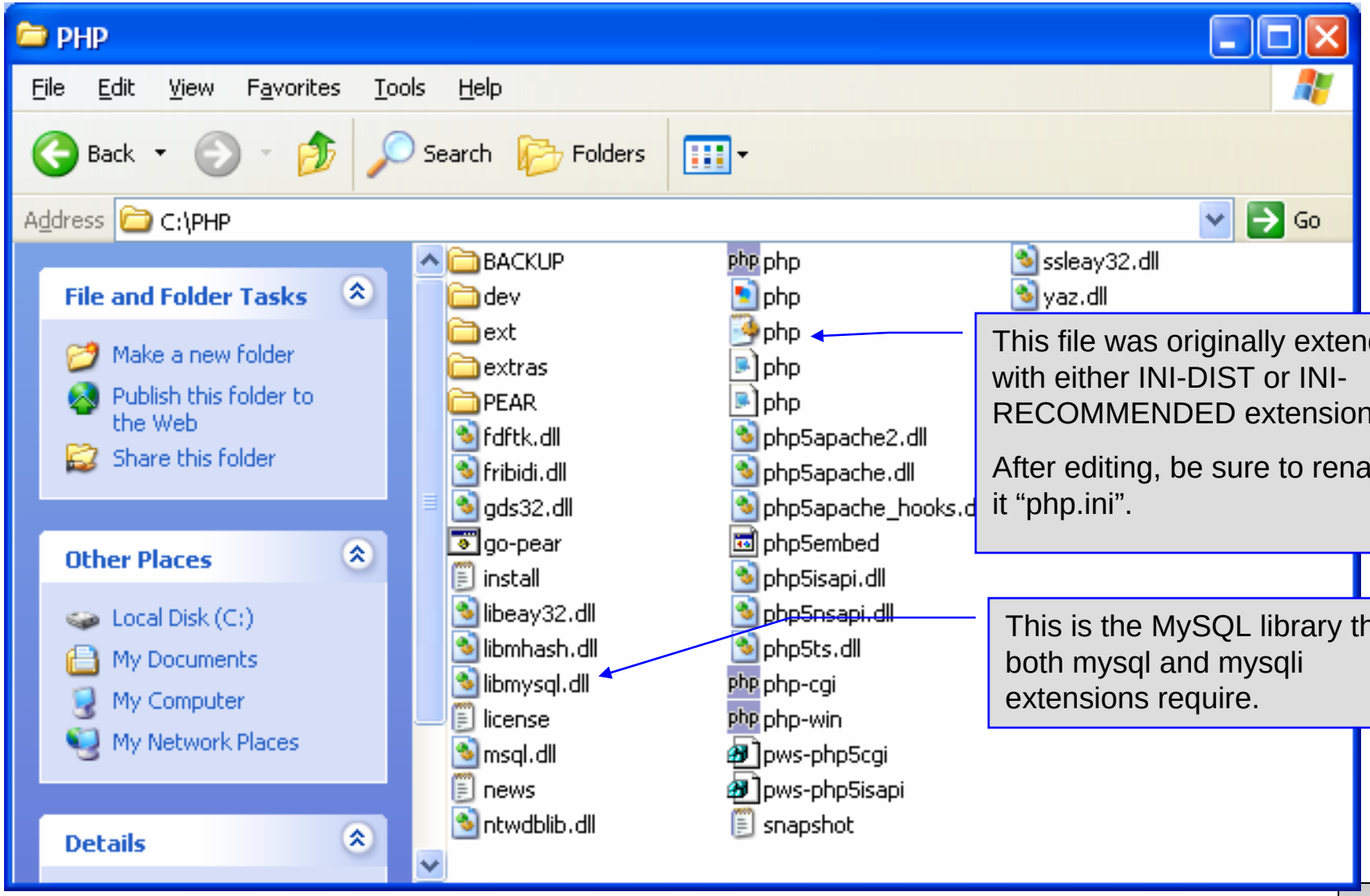
PHP and Database Connectivity (cont.)

- You need to do two things to get PHP to recognize MySQL:

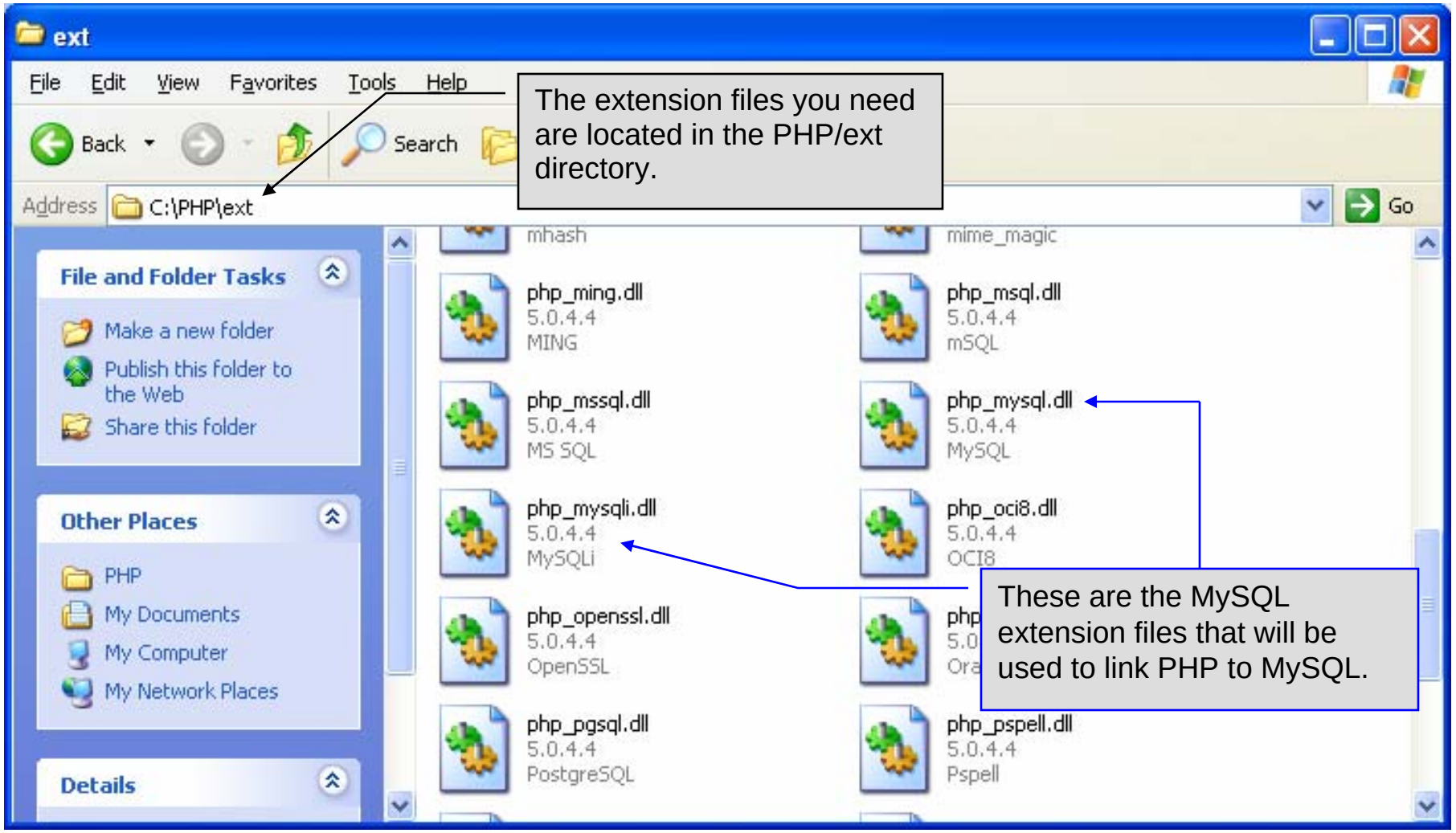
1. Set the Path statement to include `C:/php` (you should have already done this!) This enables the runtime environment to access the `libmysql.dll` and/or `libmysqli.dll` files in the PHP directory.

2. Edit the `php.ini` file to enable the extension `php_mysql.dll` (and/or extension `php_mysqli.dll`). To accomplish this search down through this file until you find the extensions (probably about ½ of the way through the file). They are all currently commented out (each line begins with a ;), simply remove the semicolon in from of the correct extension names. Be sure to rename the file `php.ini` if you haven't already done so. (See next page for example.)

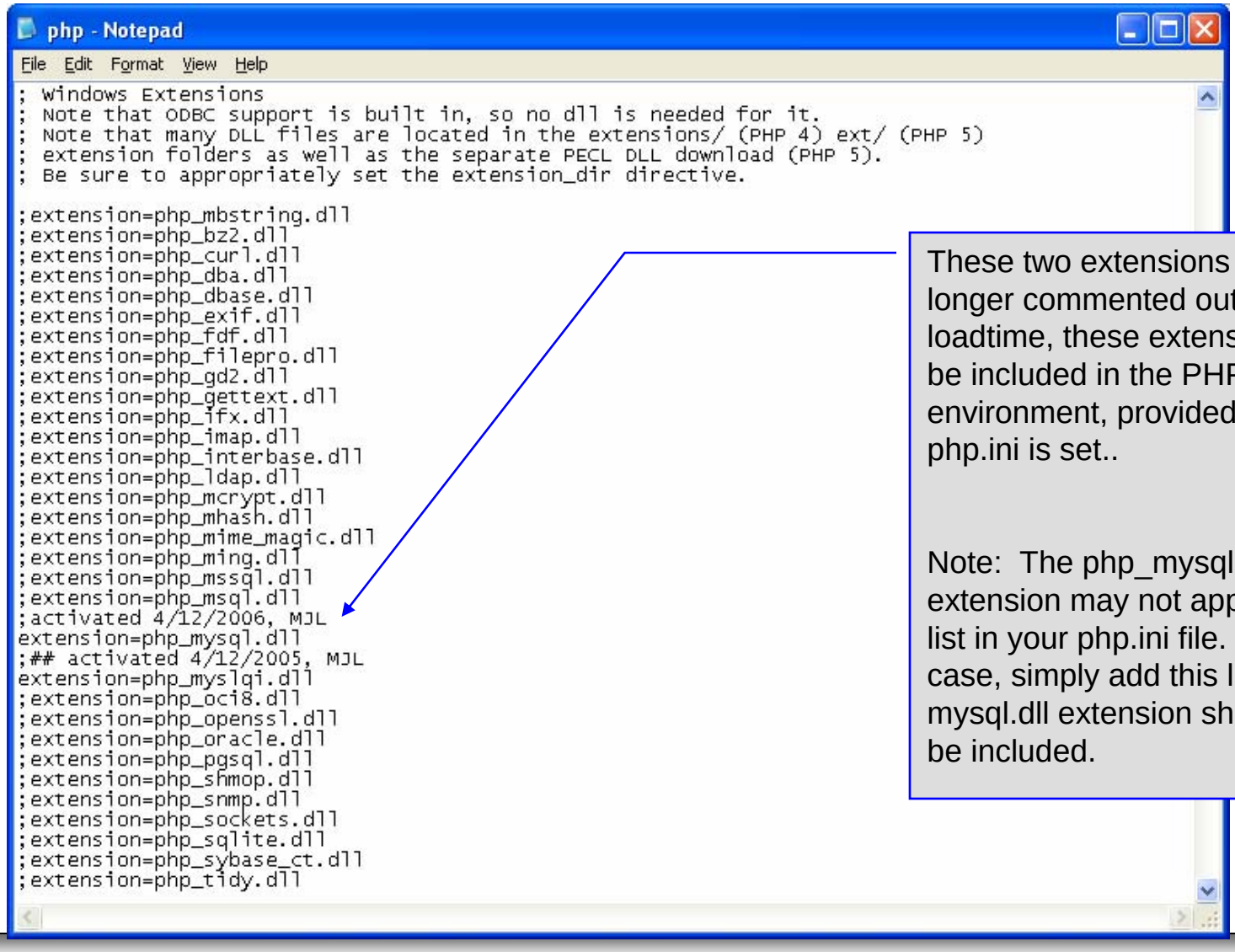
PHP and Database Connectivity (cont.)



PHP and Database Connectivity (cont.)

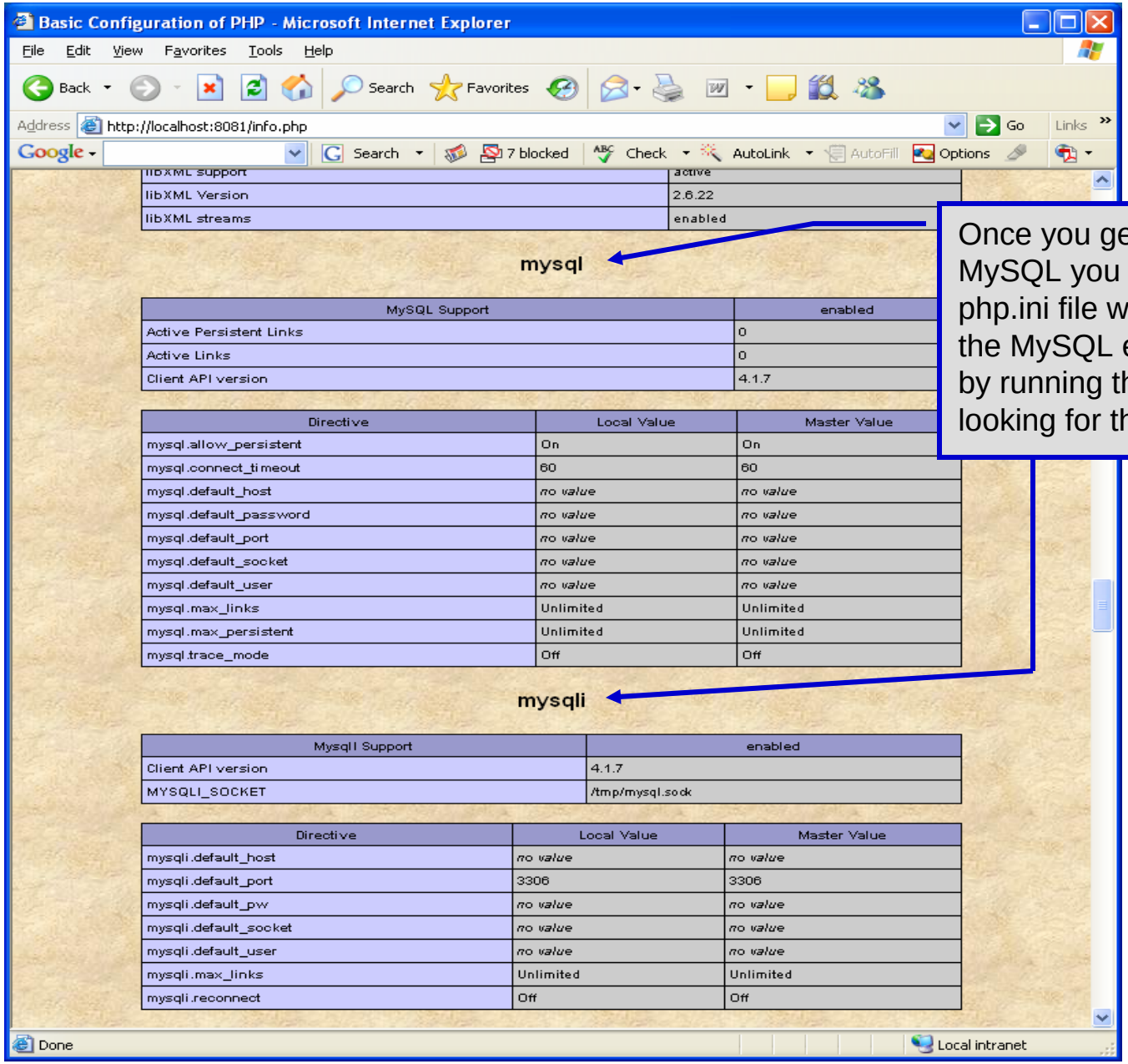


PHP and Database Connectivity (cont.)



These two extensions are no longer commented out. At loadtime, these extensions will now be included in the PHP environment, provided that the file php.ini is set..

Note: The php_mysqli.dll extension may not appear in this list in your php.ini file. If this is the case, simply add this line. The mysql.dll extension should already be included.



Once you get PHP configured for MySQL you can verify that the php.ini file was properly read and the MySQL extensions are loaded by running the info.php script and looking for these entries.

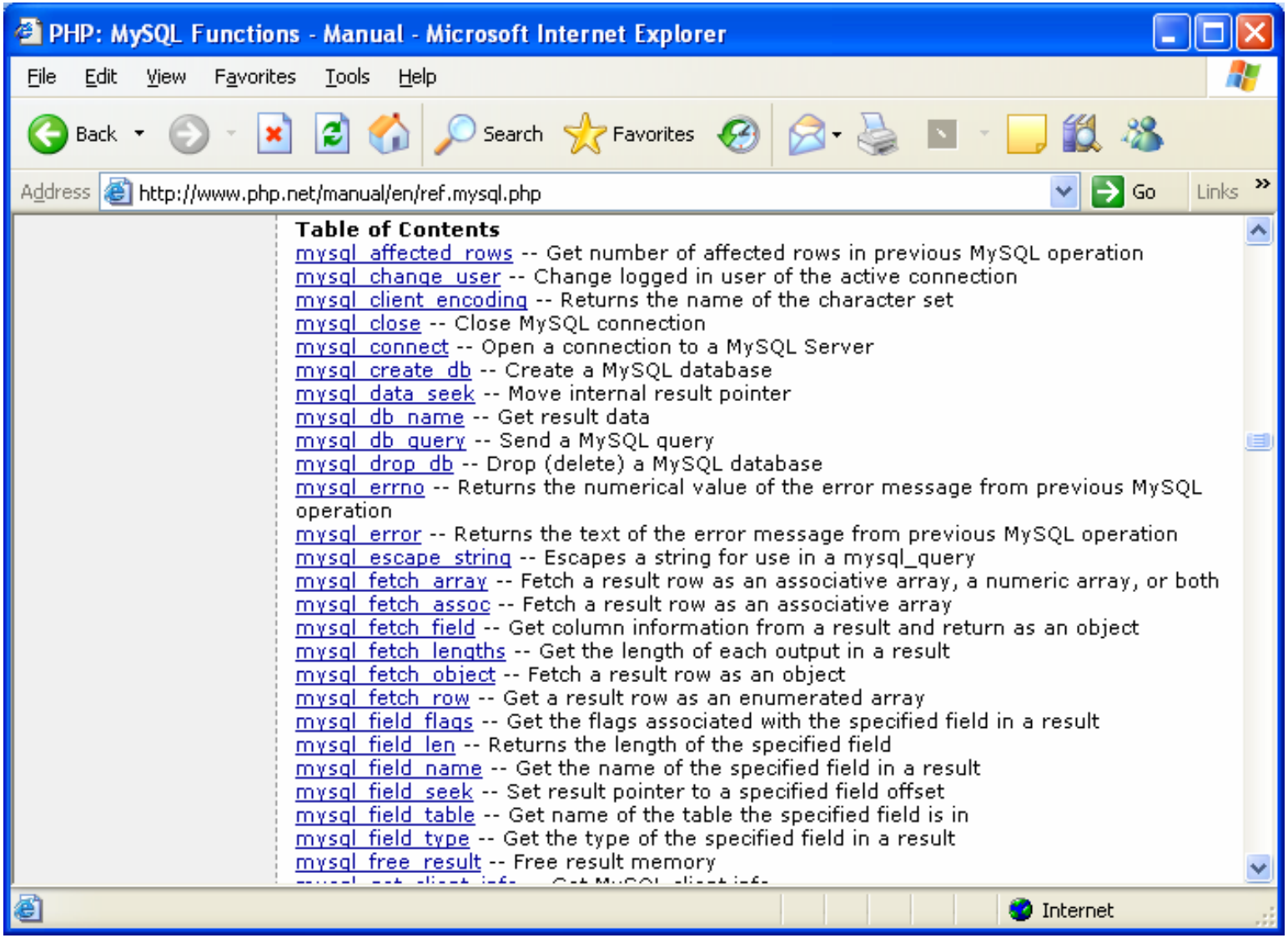
PHP and Database Connectivity (cont.)

- PHP contains a fairly extensive set of commands that can be used to access and manipulate MySQL databases.
- A very brief listing of some of these commands appears on the next page.
- For a complete listing see:

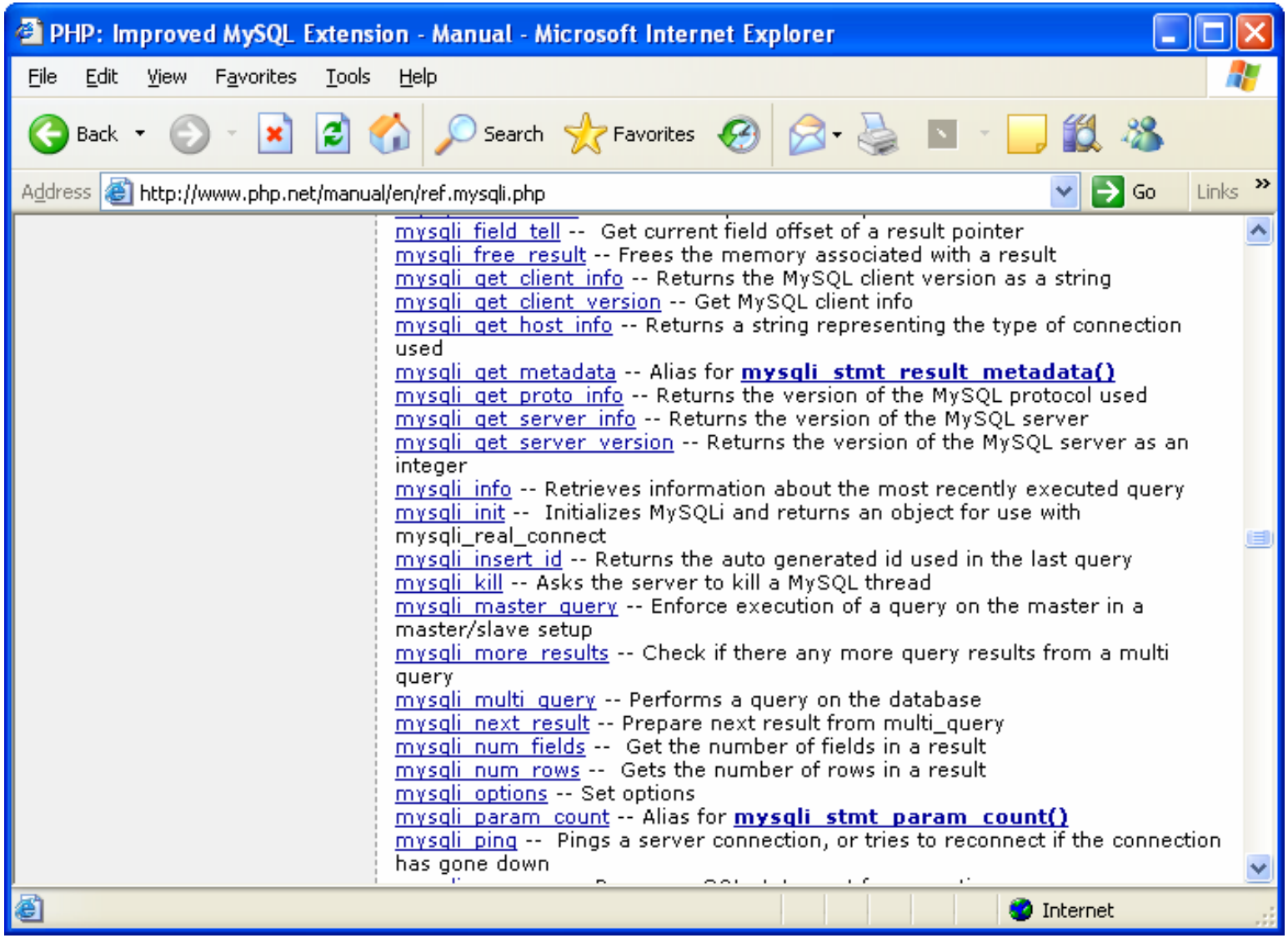
  http://us2.php.net/manual/en/print/ref.mysql.php.

  http://us2.php.net/manual/en/print/ref.mysqli.php.

Portion of `mysql.dll` Extension



Portion of `mysqli.dll` Extension



PHP and Database Connectivity (cont.)

- Now that you have PHP set to accept MySQL extensions, let's connect to the bike database that we used for examples with Java servlets and JSPs.
- The following example is a simple database connection process in PHP where the client interacts with the database from an XHTML form that simply asks them to select which attributes from the bikes table that they would like to display. This is done through the `data.html` file.
- When the client clicks the submit query button, the `database.php` script executes by connecting to the database, posting the query, retrieving the results, and displaying them to the client.

data.html

Client side

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- data.html     -->
<!-- Querying a MySQL Database From a PHP Script -->

<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>      <title>Sample Database Query From PHP</title>   </head>
  <body style = "background-color: #545454" background=image1.jpg >
    <h2 style = "font-family: arial color: blue"> Querying a MySQL database from a PHP Script. </h2>
    <form method = "post" action = "database.php">
      <p>Select a field to display:
        <!-- add a select box containing options for SELECT query   -->
        <select name = "select">
          <option selected = "selected">*</option>
          <option>bikename</option>
          <option>size</option>
          <option>color</option>
          <option>cost</option>
          <option>purchased</option>
          <option>mileage</option>
        </select>
      </p>
      <input type = "submit" value = "Send Query"   style = "background-color: blue;
        color: yellow; font-weight: bold" />
    </form>
  </body>   </html>
```


database.php

Server side

Page 1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- database.php        -->
<!-- Program to query a database and send results to the client.     -->

<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>     <title>Database Search Results</title>   </head>

  <body style = "font-family: arial, sans-serif"
    style = "background-color: #4A766E" background=image1.jpg link=blue vlink=blue>
    <?php

      extract( $_POST );

      // build SELECT query
      $query = "SELECT " . $select . " FROM bikes";

      // Connect to MySQL
      if ( !( $database = mysqli_connect( "localhost",
        "root", "root", bikedb ) ) )
        die( "Could not connect to database" );
```

Default query is to select the attributes chosen by the client for use in a SELECT query.

Connect to MySQL database. URL, username, password, and database all specified.

database.php

Server side

Page 2

```
   // query bikedb database
   if ( !( $result = mysql_query( $database, $query ) ) ) {
      print( "Could not execute query! <br />" );
      die( mysql_error() );
   }
?>

<h3 style = "color: blue">
Database Search Results</h3>
<table border = "1" cellpadding = "3" cellspacing = "3"
   style = "background-color: #00FFFF">  <!-- ADD8E6  -->

   <?php
         // fetch meta-data
         $metadata = mysqli_fetch_fields( $result);
         print("<tr>");
         for ($i=0; $i<count($metadata); $i++){
                  print("<td>");
                  printf("%s",$metadata[$i]->name);
                  print("</td>");
         }
         print("</tr>");
```

Get metadata for the query

Display metadata in the top row of the table

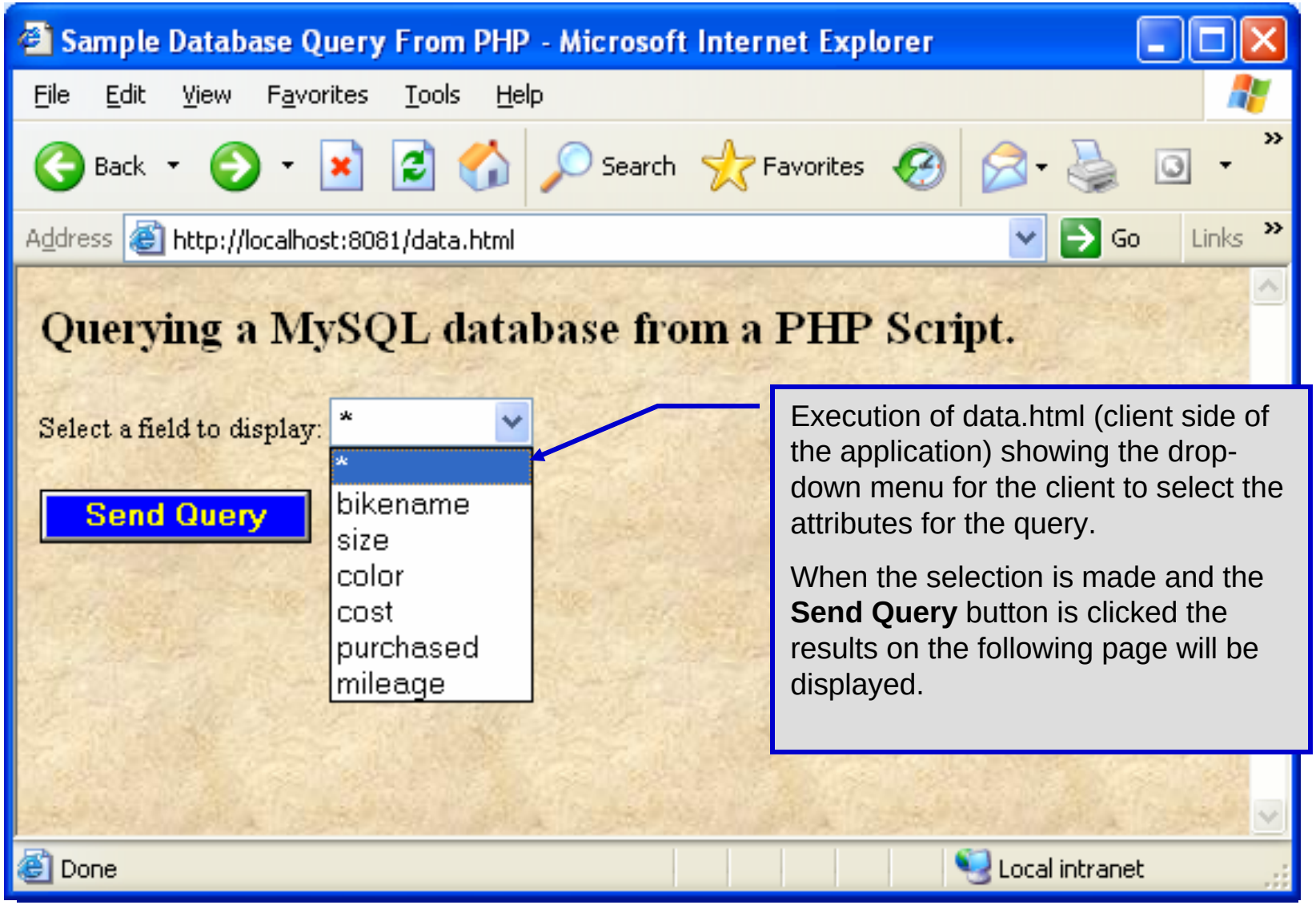
database.php

Server side

Page 3

```
      // fetch each record in result set
         for ( $counter = 0;
            $row = mysql_fetch_row( $result );
            $counter++ ){
            // build table to display results
            print( "<tr>" );
            foreach ( $row as $key => $value )
               print( "<td>$value</td>" );
            print( "</tr>" );
         }
         mysql_close( $database );
      ?>
   </table>
   <br />Your search yielded <strong>
         <?php print( "$counter" ) ?> results.<br /><br /></strong>
         <h5>Please email comments to
         <a href = "mailto:markl@cs.ucf.edu">
                              markl@cs.ucf.edu
          </a>
         </h5>
</body></html>
```

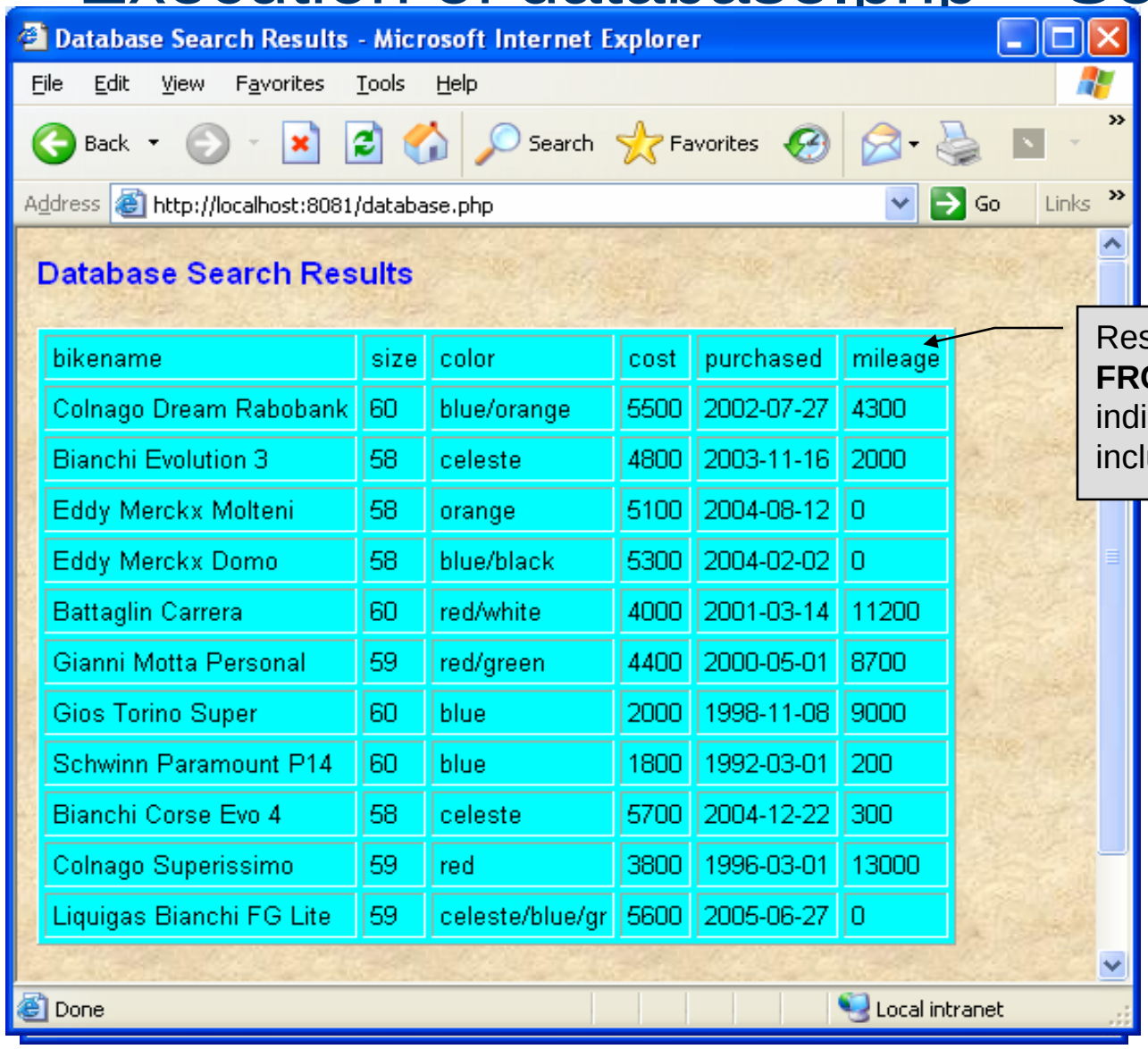

Execution of data.html – Client side



Execution of data.html (client side of the application) showing the drop-down menu for the client to select the attributes for the query.

When the selection is made and the **Send Query** button is clicked the results on the following page will be displayed.

Execution of database.php – Server side



bikename	size	color	cost	purchased	mileage
Colnago Dream Rabobank	60	blue/orange	5500	2002-07-27	4300
Bianchi Evolution 3	58	celeste	4800	2003-11-16	2000
Eddy Merckx Molteni	58	orange	5100	2004-08-12	0
Eddy Merckx Domo	58	blue/black	5300	2004-02-02	0
Battaglin Carrera	60	red/white	4000	2001-03-14	11200
Gianni Motta Personal	59	red/green	4400	2000-05-01	8700
Gios Torino Super	60	blue	2000	1998-11-08	9000
Schwinn Paramount P14	60	blue	1800	1992-03-01	200
Bianchi Corse Evo 4	58	celeste	5700	2004-12-22	300
Colnago Superissimo	59	red	3800	1996-03-01	13000
Liquigas Bianchi FG Lite	59	celeste/blue/gr	5600	2005-06-27	0

Results of query **SELECT * FROM bikes**. Display indicates that 10 rows were included in the result.

Cookies

- A cookie is a text file that a Web site stores on a client's computer to maintain information about the client during and between browsing sessions.
- A Web site can store a cookie on a client's computer to record user preferences and other information that the Web site can retrieve during the client's subsequent visits. For example, many Web sites use cookies to store client's zipcodes. The Web site can retrieve the zipcode from the cookie and provide weather reports and news updates tailored to the user's region.
- Web sites also use cookies to track information about client activity. Analysis of information collected via cookies can reveal the popularity of Web sites or products.

Cookies (cont.)

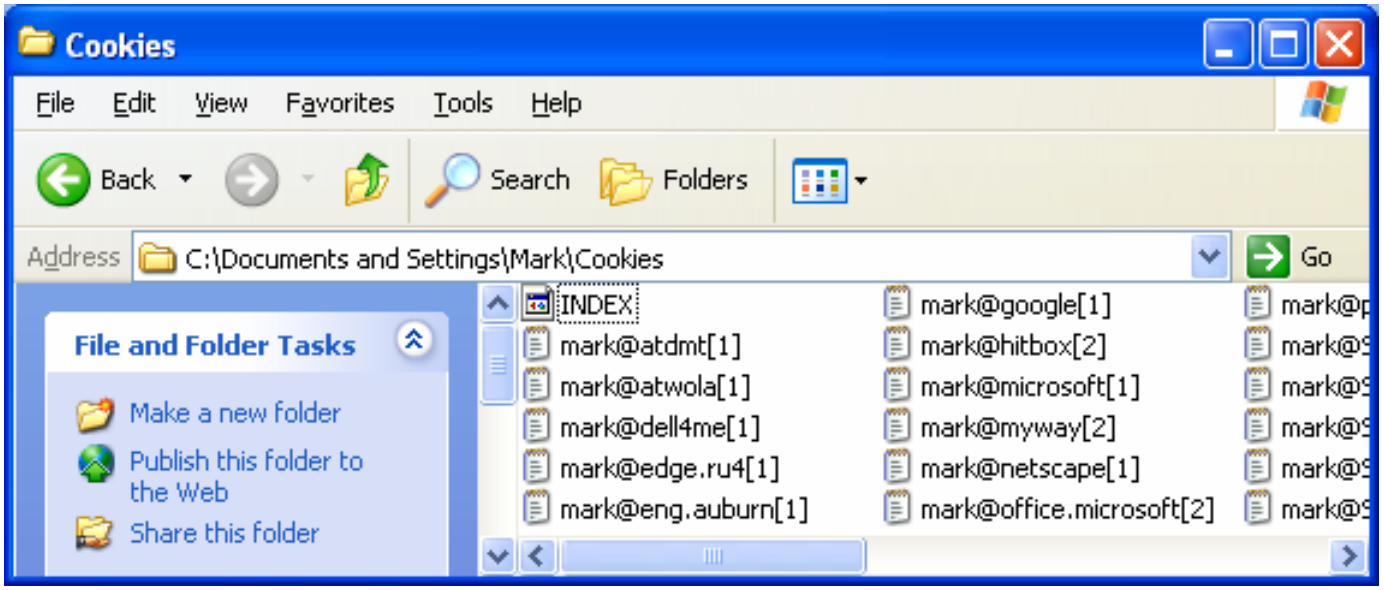
- Marketers use cookies to determine the effectiveness of advertising campaigns.
- Web sites store cookies on users' hard drives, which raises issues regarding security and privacy. Web sites should not store critical information, such as credit-card numbers or passwords, in cookies, because cookies are just text files that anyone can read.
- Several cookie features address security and privacy concerns. A server can access only the cookies that it has placed on the client.
- A cookies has an expiration date, after which the Web browser deletes it.

Cookies (cont.)

- Users who are concerned about the privacy and security implications of cookies can disable them in their Web browsers. However, the disabling of cookies can make it impossible for the user to interact with Web sites that rely on cookies to function properly.
- Information stored in the cookie is sent to the Web server from which it originated whenever the user requests a Web page from that particular server. The Web server can send the client XHTML output that reflects the preferences or information that is stored in the cookie.
- The location of the cookie file varies from browser to browser. Internet Explorer places cookies in the Cookies directory located at `C:\Documents and Settings\...\Cookies`

Cookies (cont.)

- After a cookie is created, a text file is added to this directory. While the name of the file will vary from user to user a typical example is shown below.



- The contents of a cookie are shown on page 43.

Cookies (cont.)

- Now let's create the code necessary to create our own cookie.
- In this example, a PHP script is invoked from a client-side HTML document. The HTML document creates a form for the user to enter the information that will be stored in the cookie. (Often the information that is stored in a cookie will be extracted from several different areas and may involved tracking the client's actions at the Web site.)
- Once the user has entered their information, when they click the Write Cookie button, the `cookies.php` script executes.
- The XHTML document and the PHP script are shown on the next pages. The XHTML document `cookies.html` is on page 36 and the PHP script `cookies.php` appears on page 37.

cookies.html – page 1

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

<!-- cookies.html -->
<!-- Writing a Cookie        -->

<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title>Writing a cookie to the client computer</title>   </head>

  <body style = "font-family: arial, sans-serif;
    background-color: #856363"  background=image1.jpg>
    <h2>Click Write Cookie to save your cookie data.</h2>

    <form method = "post" action = "cookies.php" style = "font-size: 10pt"
            background-color: #856363">
      <strong>Name:</strong><br />
      <input type = "text" name = "NAME" /><br />
      <strong>Height:</strong><br />
      <input type = "text" name = "HEIGHT" /><br />
      <strong>Favorite Color:</strong><br />
      <input type = "text" name = "COLOR" /><br />
      <p>
        <input type = "submit" value = "Write Cookie"  style = "background-color: #0000FF;
          color: yellow;  font-weight: bold" /></p>
    </form>
  </body>  </html>
```

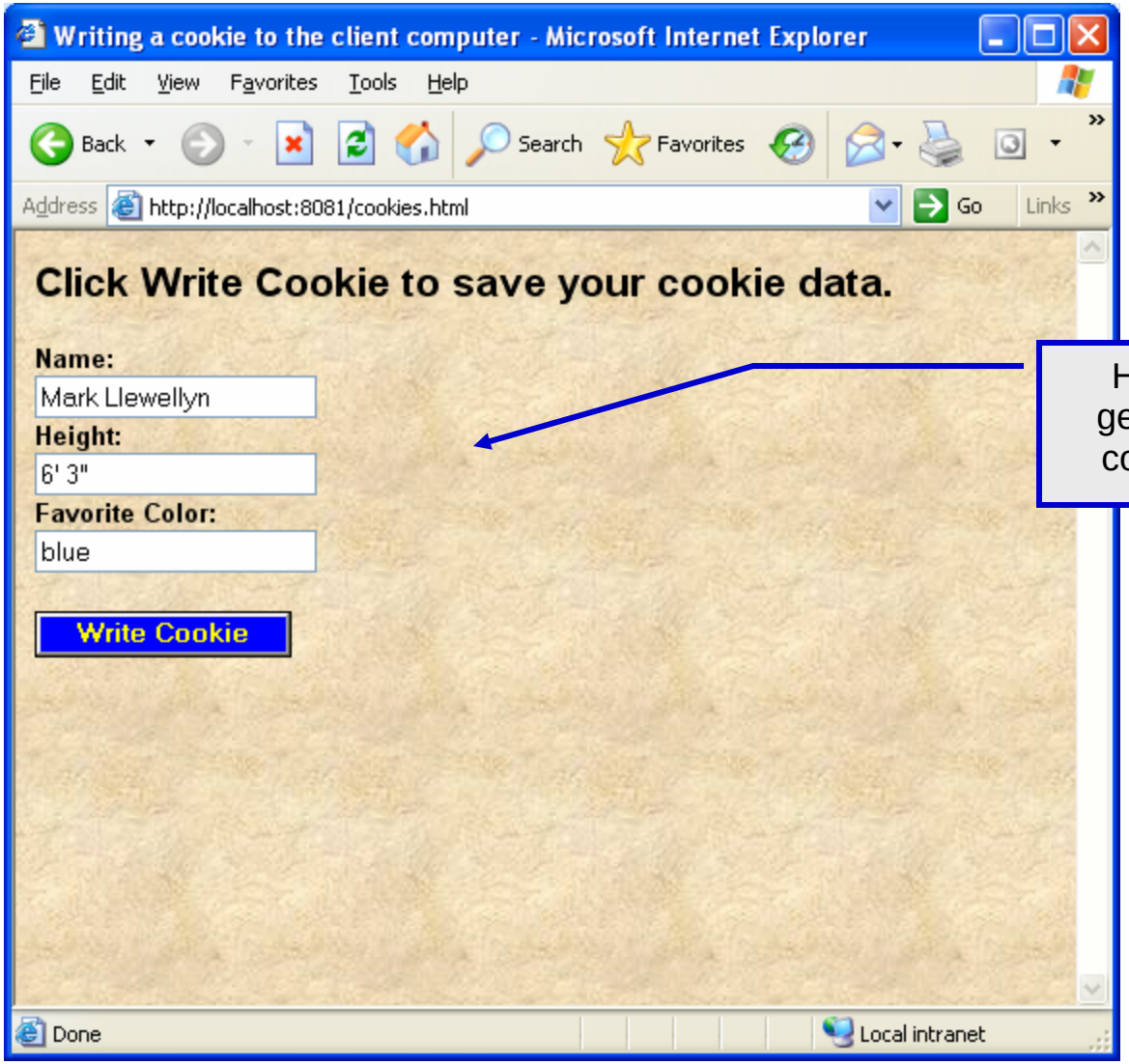

cookies.php – page 1

```
<?php
   // cookies.php
   // Program to write a cookie to a client's machine
    extract( $_POST );
   // write each form field's value to a cookie and set the
   // cookie's expiration date
   setcookie( "Name", $NAME, time() + 60 * 60 * 24 * 5 );
   setcookie( "Height", $HEIGHT, time() + 60 * 60 * 24 * 5 );
   setcookie( "Color", $COLOR, time() + 60 * 60 * 24 * 5 );
?>

<!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>Cookie Saved</title>   </head>
   <body style = "font-family: arial, sans-serif", background=image1.jpg>
      <p><b>The cookie has been set with the following data:</b></p>
      <!-- print each form field's value -->
      <br /><span style = "color: blue">Name:</span>
         <?php print( $NAME ) ?><br />
      <span style = "color: blue">Height:</span>
         <?php print( $HEIGHT ) ?><br />
      <span style = "color: blue">Favorite Color:</span>
      <span style = "color: <?php print( "$COLOR\">$COLOR" ) ?>
      </span><br />
      <p>Click <a href = "readCookies.php">here</a>  to read the saved cookie.</p>
   </body>  </html>
```

Function setcookie sets the cookies to the values passed from the cookies.html form. Function setcookie prints XHTML header information and therefore it needs to be called before any other XHTML (including comments) is printed.

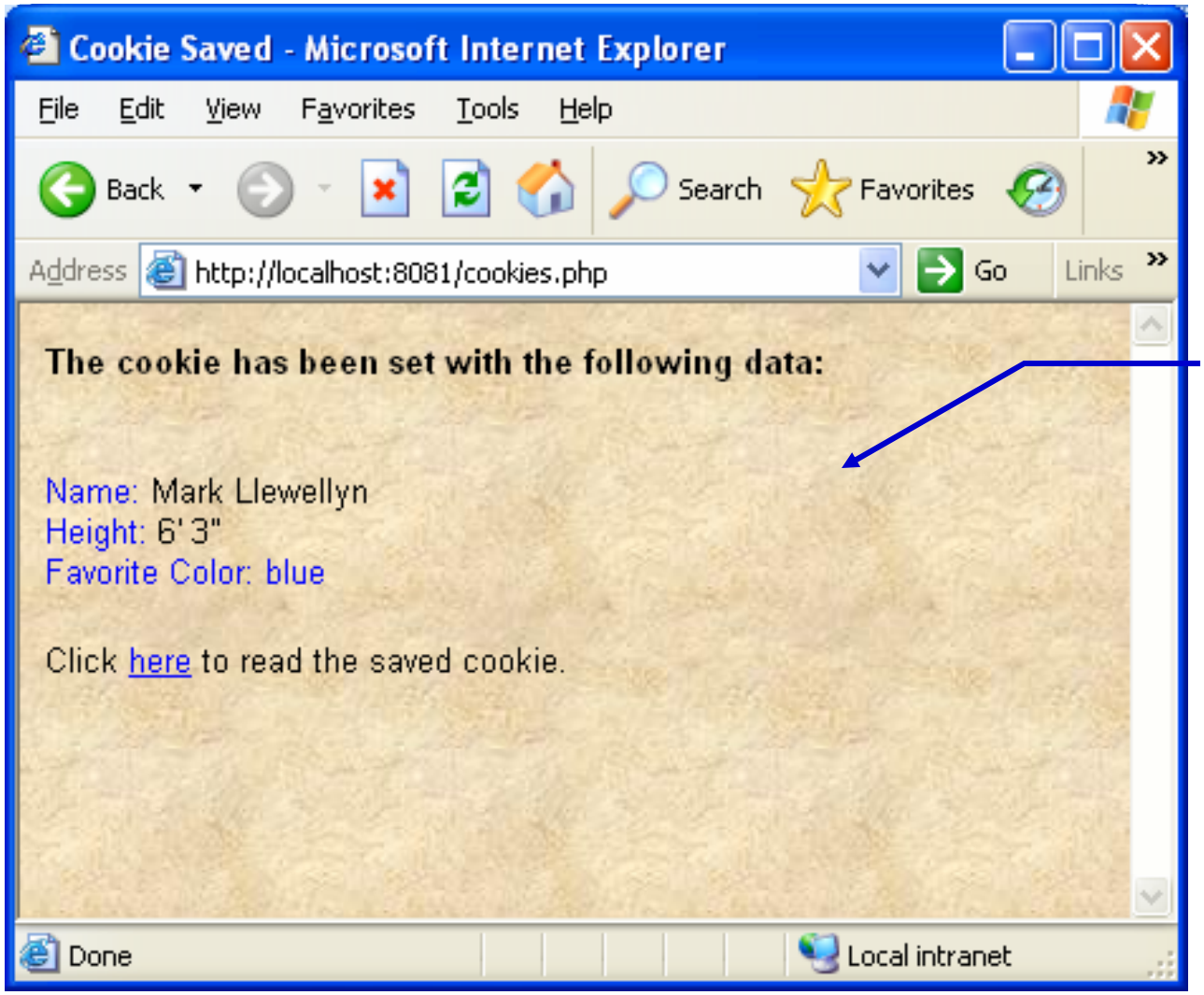
The third argument to setcookie is optional and indicates the expiration date of the cookie. In this case it is set to expire 5 days from the current time. Function time returns the current time and then we add to this the number of seconds after which the cookie is to expire.

Cookies (cont.)



HTML form generated by cookies.html

Cookies (cont.)



Output from cookies.php script showing the values in the newly created cookie.

Cookies (cont.)

- Once the cookie has been created, the cookies.php script gives the user the chance to view the newly created cookie by invoking the readCookies.php script from within the cookies.php script by clicking on the link.
- The readCookies.php script code is illustrated on the next page followed by the output from the execution of this PHP script.

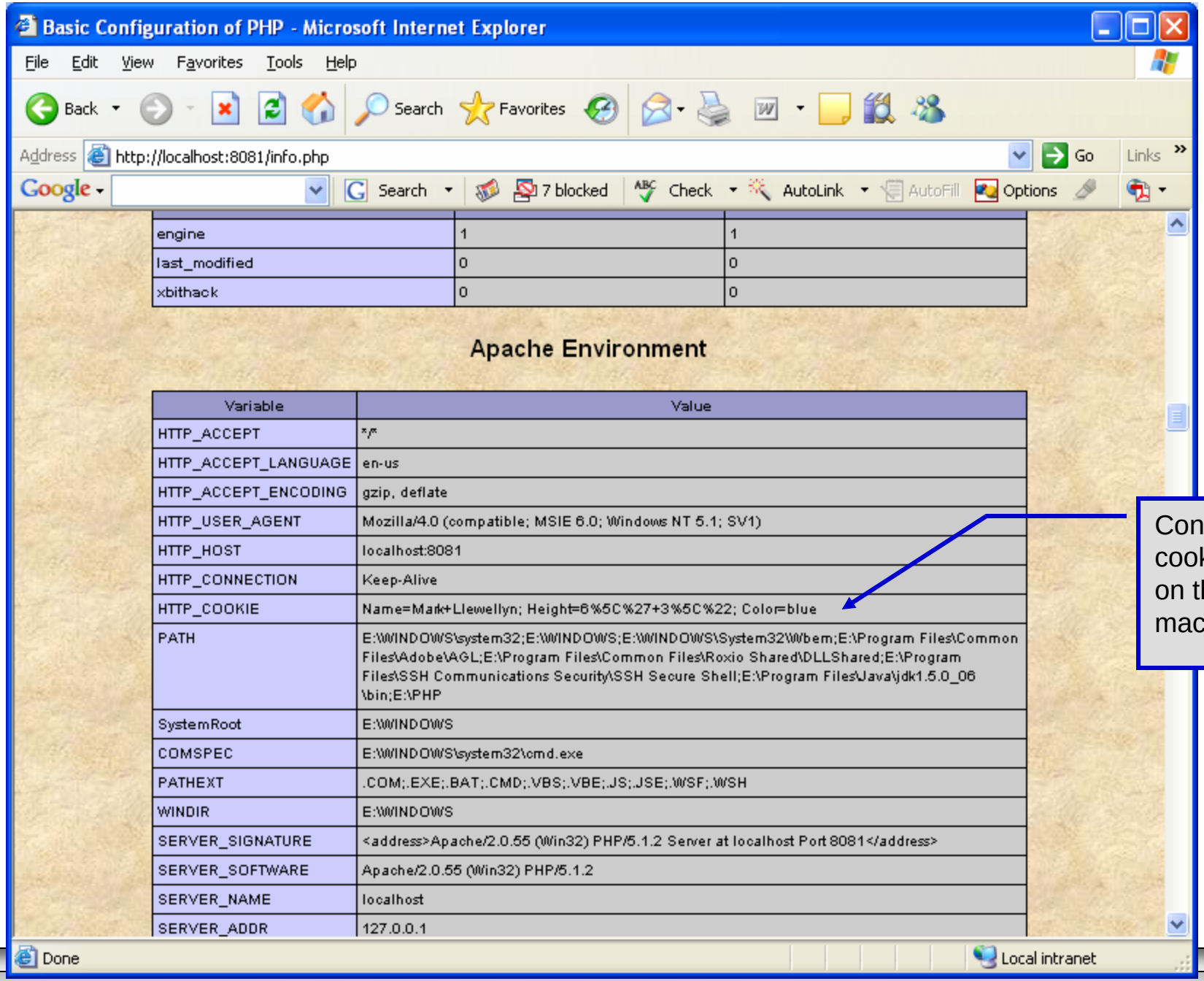
readCookies.php – page 1

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

<!-- readCookies.php                    -->
<!-- Program to read cookies from the client's computer -->

<html xmlns = "http://www.w3.org/1999/xhtml">
  <head><title>Read Cookies</title></head>

  <body style = "font-family: arial, sans-serif" background=image1.jpg>
    <p>
      <strong>   The following data is saved in a cookie on your computer.
     </strong>
    </p>
    <table border = "5" cellspacing = "0" cellpadding = "10">

      <?php
        // iterate through array $_COOKIE and print
        // name and value of each cookie
        foreach ( $_COOKIE as $key => $value )
          print( "<tr>
            <td bgcolor=\"#F0E68C\">$key</td>
            <td bgcolor=\"#FFA500\">$value</td>
            </tr>" );
      ?>
    </table>
  </body>  </html>
```

Superglobal array holding cookie.

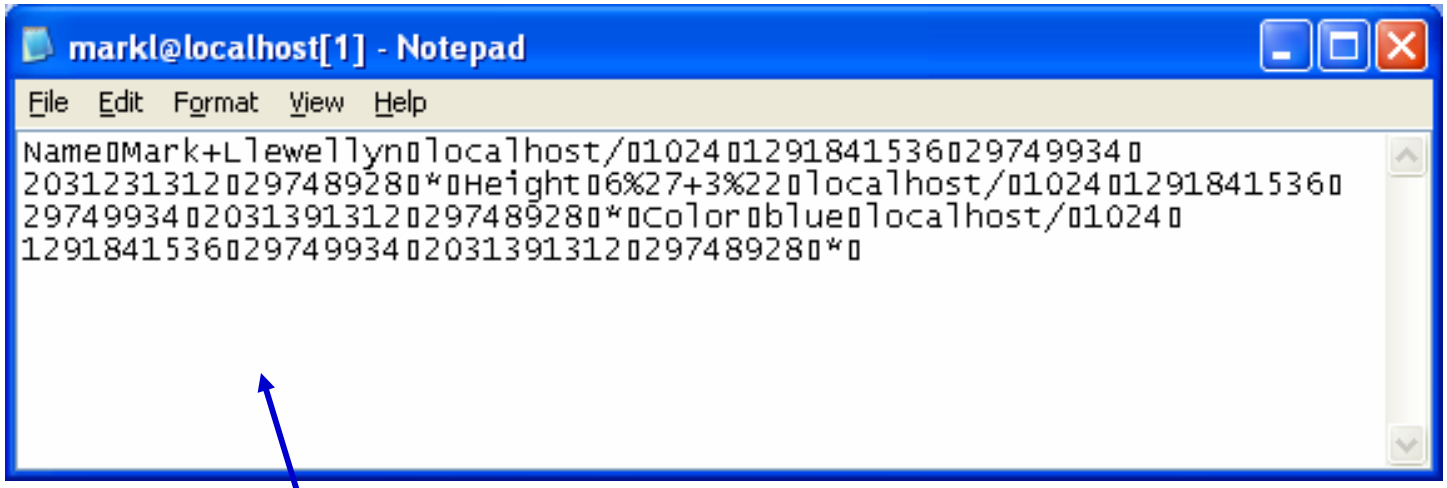
Cookies (cont.)

Output from the readCookies.php script.



Contents of the cookie stored on the client machine.

Cookies (cont.)



Actual text file holding cookie data for the cookie that was created in this example.