

COP 4610L: Applications in the Enterprise Spring 2005

Introduction to PHP – Part 1

Instructor : Mark Llewellyn
markl@cs.ucf.edu
CSB 242, 823-2790
<http://www.cs.ucf.edu/courses/cop4610L/spr2005>

School of Computer Science
University of Central Florida



Introduction to PHP

- PHP is officially known as PHP: Hypertext Preprocessor and is very rapidly becoming the most popular server-side scripting language for creating dynamic web pages.
- PHP was created in 1994 by Rasmus Lerdorf (who currently works for Linuxcare, Inc. as a senior open-source researcher) to track users at his Web site. Lerdorf originally called it Personal Home Page Tools in a package he released in 1995. It eventually became an Apache Software Foundation project.
- PHP2 featured built-in database support and form handling. In 1997, PHP3 was released and featured a new parser which substantially increased performance and led to an explosion in PHP use.



Introduction to PHP (cont.)

- PHP4 featured the Zend Engine and was considerably faster and more powerful than its predecessors and further enhanced the popularity of PHP.
- The current release is PHP5.0.4 and features the Zend Engine 2, which provides further increases in speed and functionality. You can download the latest version of PHP at www.php.net. For more details on the Zend Engine 2 see www.zend.com.
- Today more than 17 million domains utilize PHP technology.
- All of the examples we'll be looking at use the latest stable version of PHP which is 5.0.4.



Introduction to PHP (cont.)

- The power of the Web resides not only in serving content to users, but also in responding to requests from users and generating Web pages with dynamic content.
- Interactivity between the user and the server has become a crucial part of Web functionality. While other languages can also perform these functions, PHP was written specifically for interacting with the Web.
- PHP code is embedded directly into XHTML documents. This allows the document author to write XHTML in a clear, concise manner, without having to use multiple `print` statements, as is necessary with other CGI-based languages.

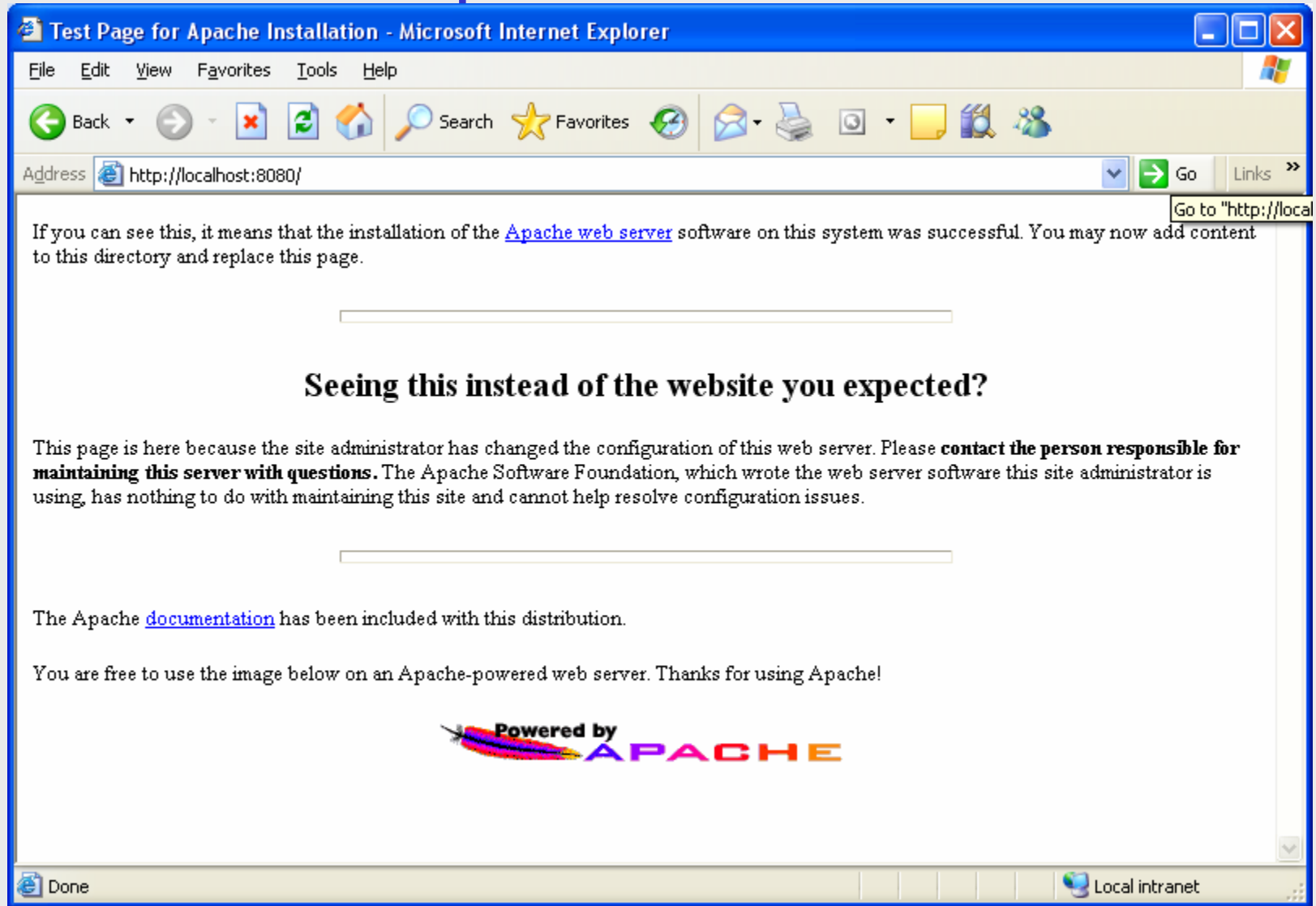


Introduction to PHP (cont.)

- PHP script file names usually end with `.php`, although a server can be configured to handle other file extensions.
- To run a PHP script, PHP must first be installed on your system. Download PHP5.0.x from www.php.net. (Most recent version is 5.0.4, but any of the 5.0.x versions should be ok.)
- Although PHP can be used from the command line, a Web server is required to take full advantage of the scripting language. I would suggest the Apache server available from www.apache.org. (Note: this is not the Tomcat server you've already used.) Current version is 2.0.54 which just fixed a few simple bugs from some of the earlier versions. I would expect that any of the 2.0.x versions would be ok for what we will be doing.



Apache Server



Apache Server Set-up

- Once you get the Apache Server downloaded and running on your machine...you've seen the screen on the previous page, you'll need to configure Apache to work with PHP.
- There are a couple of steps required to accomplish this task:
 1. Assume that you've downloaded PHP and placed it in the directory `c:/php`.
 2. Add the PHP directory to the PATH.
 3. Setup a valid configuration file for PHP. Do the following:
 - a) Copy `php.ini-recommend` inside `c:/php` and rename it to `php.ini`.



Apache Server Set-up (cont.)

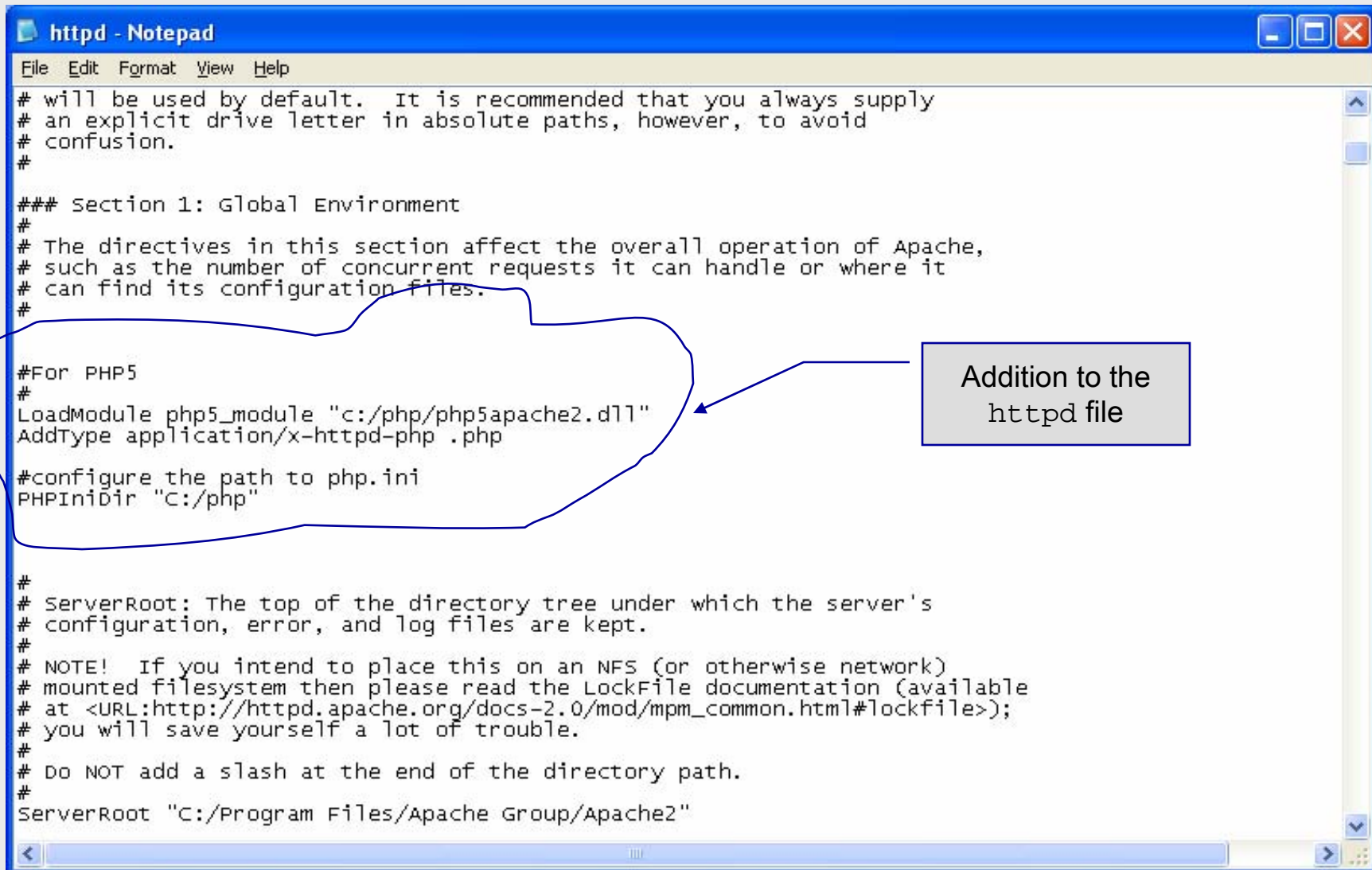
4. Install PHP as an Apache module by doing the following:
 - a) Edit the Apache `httpd.conf` file found in the `Apache conf` directory.
 - b) Add the following lines to this file in Section1: Global Environment. (screen shot on next page shows location of this edit)

```
#For PHP5
#
LoadModule php5_module "c:/php/php5apache2.dll"
AddType application/x-httpd-php .php
#configure the path to php.ini
PHPIniDir "C:/php"
```

5. Once these steps are completed, Apache is configured to run PHP (basic components – more later). When you've completed these steps, you can begin writing PHP code.



Apache Server Set-up (cont.)



```
File Edit Format View Help
# will be used by default. It is recommended that you always supply
# an explicit drive letter in absolute paths, however, to avoid
# confusion.
#
### Section 1: Global Environment
#
# The directives in this section affect the overall operation of Apache,
# such as the number of concurrent requests it can handle or where it
# can find its configuration files.
#
#For PHP5
#
LoadModule php5_module "c:/php/php5apache2.dll"
AddType application/x-httpd-php .php

#configure the path to php.ini
PHPIniDir "C:/php"

#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# NOTE! If you intend to place this on an NFS (or otherwise network)
# mounted filesystem then please read the LockFile documentation (available
# at <URL:http://httpd.apache.org/docs-2.0/mod/mpm_common.html#lockfile>);
# you will save yourself a lot of trouble.
#
# Do NOT add a slash at the end of the directory path.
#
ServerRoot "C:/Program Files/Apache Group/Apache2"
```



A PHP Test Example

Create this file named `hello.php` and save it to the `htdocs` folder in Apache. Then start the Apache server, enter the URL: <http://localhost:8080/hello.php> and you should see output similar to that shown on the next slide.

```
<html>
<head>
<title>Hello From PHP</title>
</head>
<body style = "font-family: arial, sans-serif;
    background-color: #856363" background=image1.jpg>
<h1> Hello From PHP</h1>
```

```
<?
    print "Current Information";
    phpInfo();
?>
```

```
</body>
</html>
```

This is
PHP




Microsoft Internet Explorer window titled "Hello From PHP". The address bar shows `http://localhost:8080/hello.php`.

Hello From PHP

Current Information

PHP Version 5.0.4



System	Windows NT MARK-PC 5.1 build 2600
Build Date	Mar 31 2005 02:44:34
Configure Command	<code>csript /nologo configure.js "--enable-snapshot-build" "--with-gd=shared"</code>
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
PHP API	20031224
PHP Extension	20041030
Zend Extension	220040412
Debug Build	no
Thread Safety	enabled
IPv6 Support	enabled
Registered PHP Streams	php, file, http, ftp, compress.zlib
Registered Stream Socket Transports	tcp, udp

The default directory for the `php.ini` file will be the system directory `C:\WINDOWS` unless you set the path to the `c:\php` directory using the technique shown on page 7



Finding the details of your PHP set-up

PHP Version 5.0.4



System	Windows NT MARK-PC 5.1 build 2600
Build Date	Mar 31 2005 02:44:34
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--with-gd=shared"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\PHP\php.ini
PHP API	20031224
PHP Extension	20041030

The default directory for the `php.ini` file has been changed via the `httpd` file from within Apache. The values in `php.ini` will now be used to configure PHP under Apache.



A First PHP Example

- The following two pages illustrate a simple PHP “hello world” program.
- In PHP, code is inserted between the scripting delimiters `<?php` and `?>`. PHP code can be placed anywhere in XHTML markup, as long as the code is enclosed in these scripting delimiters.



welcome.php Example

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

```
<!-- welcome.php -->  
<!-- XHTML file containing a PHP script. -->
```

```
<?php  
  $name = "Mark";    //php declaration and assignment  
>
```

PHP code
declaring a
variable.

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

```
<!-- head section of document -->  
<head>  
  <title>A Simple PHP Document</title>  
</head>
```

```
<!-- body section of document -->  
<body style = "font-size: 2em">  
  <hr>  
  <font color = blue><h1> Generating HTML From PHP </h1></font color>  
  <p>
```



welcome.php Example

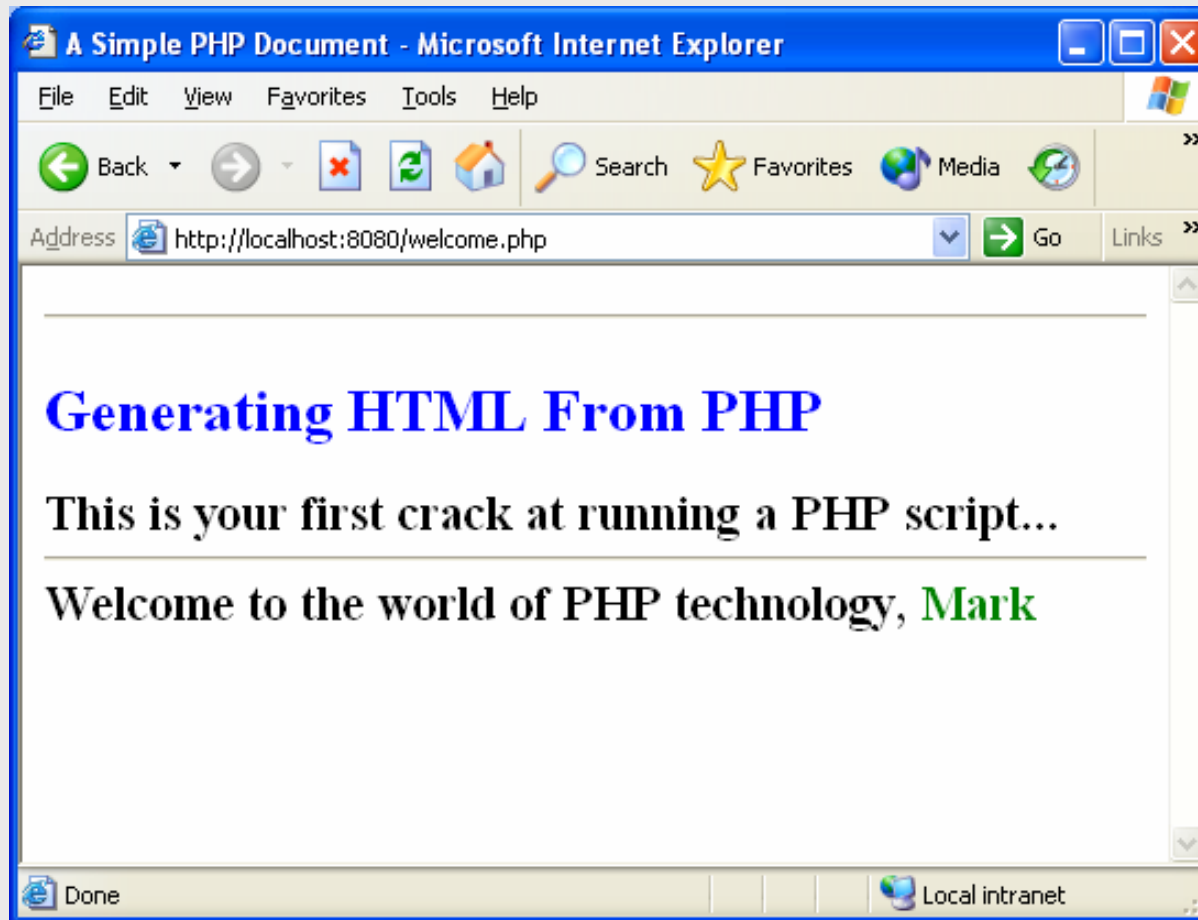
```
<strong>
  <!--print variable name's value in the message-->
  {
  <?php
    print("This is your first crack at running a PHP script...");
    print("<HR>");
    print("Welcome to the world of PHP technology, ");
  ?>
  <font color = green>
  {
  <?php
    print("$name");
  ?>
  </font color>
  }
  </strong>
</p>
</body>
</html> <!-- end XHTML document -->
```

PHP code

PHP code



welcome.php Example Output



Viewing Client/Server Environment Variables

- Knowledge of a client's execution environment is useful to system administrators who want to provide client-specific information.
- Environment variables contain information about a script's environment, such as the client's web browser, the HTTP host and the HTTP connection.
 - The table on the next page summarizes some of the superglobal arrays defined by PHP.
- The XHTML document on page 18 displays the values of the client's environment variables in a table. PHP stores the environment variables and their values in the `$_ENV` array. Iterating through the array allows one to view all of the client's environment variables.



Some Superglobal Environment Arrays

Variable Name	Description
<code>\$_SERVER</code>	Data about the currently running server.
<code>\$_ENV</code>	Data about the client's environment.
<code>\$_GET</code>	Data posted to the server by the <code>get</code> method.
<code>\$_POST</code>	Data posted to the server by the <code>post</code> method.
<code>\$_COOKIE</code>	Data contained in cookies on the client's computer.
<code>\$GLOBALS</code>	Array containing all global variables.



env.php Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- env.php -->
<!-- Program to display environment variables -->

<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title>Environment Variables Display</title>
  </head>

  <body>
    <table border = "0" cellpadding = "2" cellspacing = "0"
      width = "100%">
      <?php
        // print the key and value for each element
        // in the $_ENV array
        foreach ( $_ENV as $key => $value )
          print( "<tr><td bgcolor = \"#11bbff\">
            <strong>$key</strong></td>
            <td>$value</td></tr>" );
      ?>
    </table>
  </body>
</html>
```

Iterate through the `$_ENV` array to list all of the environment variable for the client system.



Output from executing env.php

ALLUSERSPROFILE	C:\Documents and Settings\All Users
APPDATA	C:\Documents and Settings\markl\Application Data
CATALINA_HOME	c:\Program Files\Apache Software Foundation\Tomcat 5.5
CLASSPATH	c:\program files\java\jdk1.5.0;c:\Program Files\apache software foundation\Tomcat 5.5\common\lib;c:\Program Files\Java\jdk1.5.0\bin;
CommonProgramFiles	C:\Program Files\Common Files
COMPUTERNAME	MARK-PC
ComSpec	C:\WINDOWS\system32\cmd.exe
FP_NO_HOST_CHECK	NO
HOMEDRIVE	Z:
HOMEPATH	\
HOMESHARE	\csserver\home\markl
JAVA_HOME	c:\Program Files\Java\jdk1.5.0
LOGONSERVER	\CSSERVER
NUMBER_OF_PROCESSORS	2
OS	Windows_NT
Path	C:\Program Files\Common Files\Sonic Shared;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Program Files\MySQL\MySQL Server 4.1\bin;c:\php;c:\Sun\AppServer\bin;;c:\php;c:\php\ext
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
PROCESSOR_ARCHITECTURE	x86
PROCESSOR_IDENTIFIER	x86 Family 15 Model 2 Stepping 9, GenuineIntel
PROCESSOR_LEVEL	15
PROCESSOR_REVISION	0209



Form Processing and Business Logic

- XHTML forms enable web pages to collect data from users and send it to a web server for processing.
- Interaction of this kind between users and web servers is vital to e-commerce applications. Such capabilities allow users to purchase products, request information, send and receive web-based email, perform on-line paging and take advantage of various other online services.
- The XHTML document on the next few pages collects information from a user for the purposes of adding them to a mailing list.
- The PHP file on page XX validates the data entered by the user through the form and “registers” them in the mailing list database.



form.html Example

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

```
<!-- form.html -->
```

```
<!-- Form for use with the form.php program -->
```

This XHTML document generates the form that the user will submit to the server via form.php

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

```
<head>
```

```
<title>Sample form to take user input in XHTML</title>
```

```
</head>
```

```
<body>
```

```
<h1>This is a sample registration form.</h1>
```

Please fill in all fields and click Register.

```
<!-- post form data to form.php -->
```

```
<form method = "post" action = "form.php">
```

```
<img src = "images/user.gif" alt = "User" /><br />
```

```
<span style = "color: blue">
```

```
  Please fill out the fields below.<br />
```

```
</span>
```

```
<!-- create four text boxes for user input -->
```

```
<img src = "images/fname.gif" alt = "First Name" />
```

```
<input type = "text" name = "fname" /><br />
```



```
<img src = "images/lname.gif" alt = "Last Name" />
<input type = "text" name = "lname" /><br />
<img src = "images/email.gif" alt = "Email" />
<input type = "text" name = "email" /><br />
<img src = "images/phone.gif" alt = "Phone" />
<input type = "text" name = "phone" /><br />
<span style = "font-size: 10pt">
  Must be in the form (555)555-5555</span>
<br /><br />
<img src = "images/downloads.gif"
  alt = "Products" /><br />

<span style = "color: blue">
  Which publication would you like information about?
</span><br />

<!-- create drop-down list containing magazine names -->
<select name = "magazine">
  <option>Velo-News</option>
  <option>Cycling Weekly</option>
  <option>Pro Cycling</option>
  <option>Cycle Sport</option>
    <option>RadSport</option>
    <option>Mirror du Cyclisme</option>
</select>
<br /><br />
```



```
<img src = "images/os.gif" alt = "Operating System" />
<br /><span style = "color: blue">
  Which operating system are you currently using?
<br /></span>
<!-- create five radio buttons -->
<input type = "radio" name = "os" value = "Windows XP"
  checked = "checked" />
  Windows XP
<input type = "radio" name = "os" value =
  "Windows 2000" />
  Windows 2000
<input type = "radio" name = "os" value =
  "Windows 98" />
  Windows 98<br />
<input type = "radio" name = "os" value = "Linux" />
  Linux

<input type = "radio" name = "os" value = "Other" />
  Other<br />

<!-- create a submit button -->
<input type = "submit" value = "Register" />
</form>

</body>
</html>
```



form.php Example

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

```
<!-- form.php -->
```

```
<!-- Read information sent from form.html -->
```

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

```
<head>
```

```
<title>Form Validation</title>
```

```
</head>
```

```
<body style = "font-family: arial,sans-serif">
```

```
<?php
```

```
extract($_POST);
```

```
// determine whether phone number is valid and print an error message if not
```

```
if ( !ereg( "^\([0-9]{3}\)[0-9]{3}-[0-9]{4}$",
```

```
$phone ) ){
```

```
print( "<p><span style = \"color: red; font-size: 2em\">
```

```
INVALID PHONE NUMBER:</span><br />
```

```
A valid phone number must be in the form
```

```
<strong>(555)555-5555</strong><br />
```

```
<span style = \"color: blue\">
```

```
Click the Back button, enter a valid phone number and resubmit.<br /><br />
```

```
Thank You.</span></p></body></html>" );
```

```
die(); // terminate script execution
```

```
}
```

```
?>
```

Function `extract` (`associativeArray`) creates a variable-value pair corresponding to each key-value pair in the associative array `$_POST`.

See page 28 for explanation of regular expressions.

Function `die()` terminates script execution. An error has occurred, no need to continue.

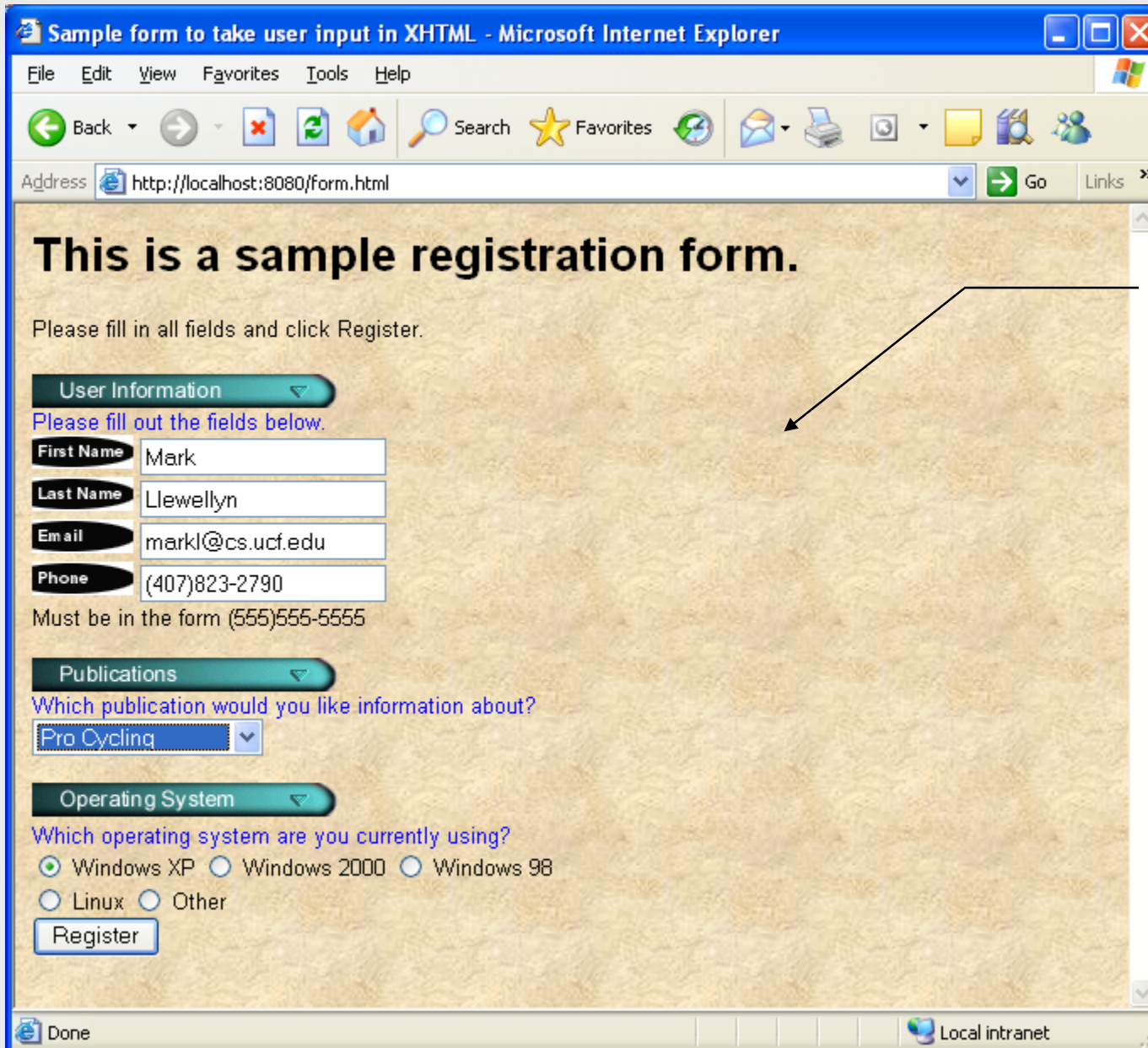


```

<p>Hi
  <span style = "color: blue"> <strong> <?php print( "$fname" ); ?> </strong> </span>.
  Thank you for completing the survey.<br />
  You have been added to the <span style = "color: blue">
    <strong> <?php print( "$magazine " ); ?> </strong> </span> mailing list.
</p>
<strong>The following information has been saved in our database:</strong><br />
<table border = "0" cellpadding = "0" cellspacing = "10">
  <tr>
    <td bgcolor = "#ffffaa">Name </td>
    <td bgcolor = "#ffffbb">Email</td>
    <td bgcolor = "#ffffcc">Phone</td>
    <td bgcolor = "#ffffdd">OS</td>
  </tr>
  <tr>
    <?php
      // print each field's value
      print( "<td>$fname $lname</td> <td>$email</td> <td>$phone</td> <td>$os</td>" );
    ?>
  </tr>
</table>
<br /><br /><br />
<div style = "font-size: 10pt; text-align: center">
  This is only a sample form.  You have not been added to a mailing list.
</div>
</body>
</html>

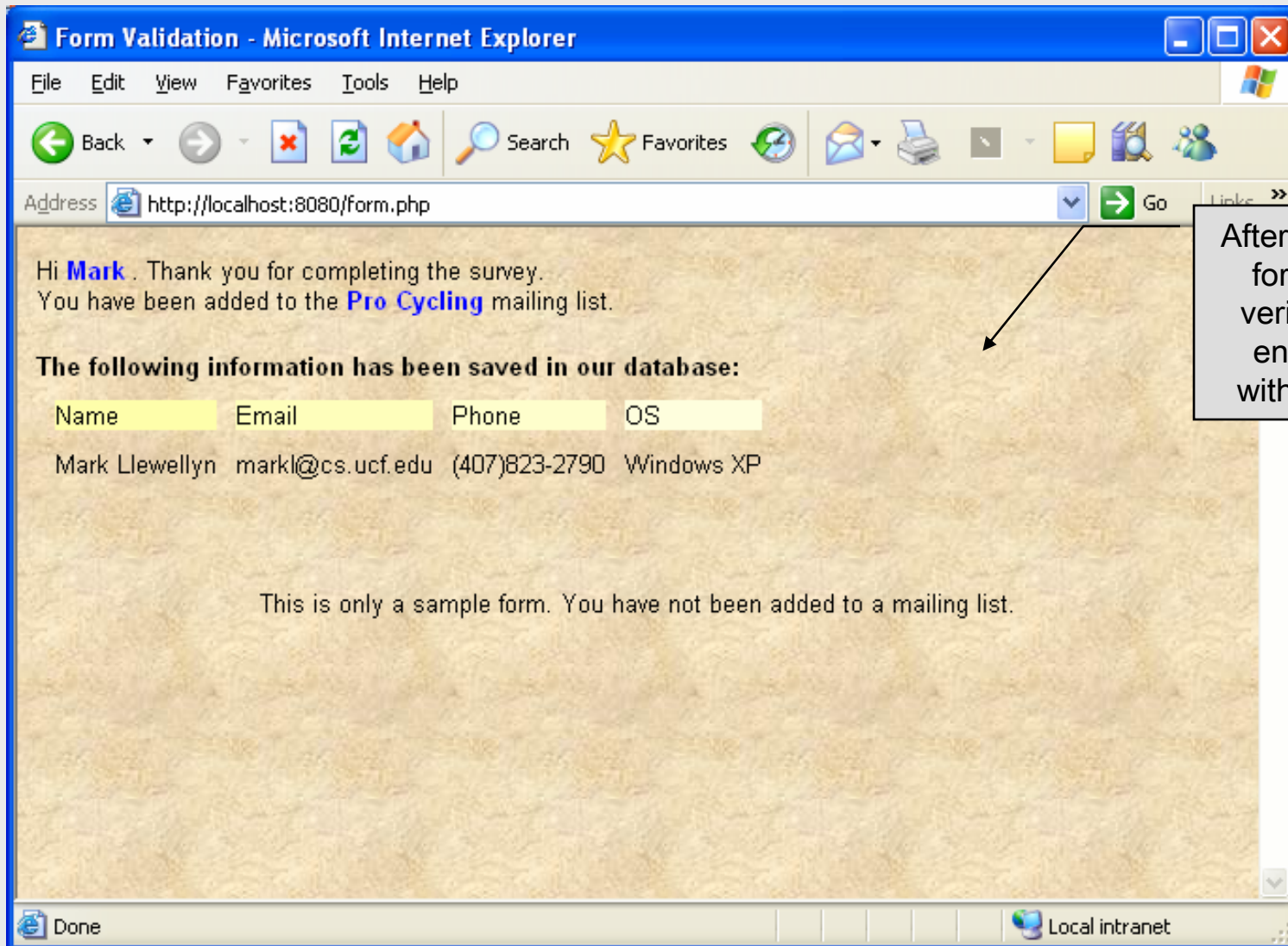
```





Execution of form.html within a web browser





After execution of form.php has verified correct entries made within the form.



Sample form to take user input in XHTML - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://localhost:8080/form.html

This is a sample registration form.

Please fill in all fields and click Register.

User Information

Please fill out the fields below.

First Name Mark

Last Name Llewellyn

Email markl@cs.ucf.edu

Phone 407-823-2790

Must be in the form (555)555-5555

Publications

Which publication would you like information about?

Pro Cycling

Operating System

Which operating system are you currently using?

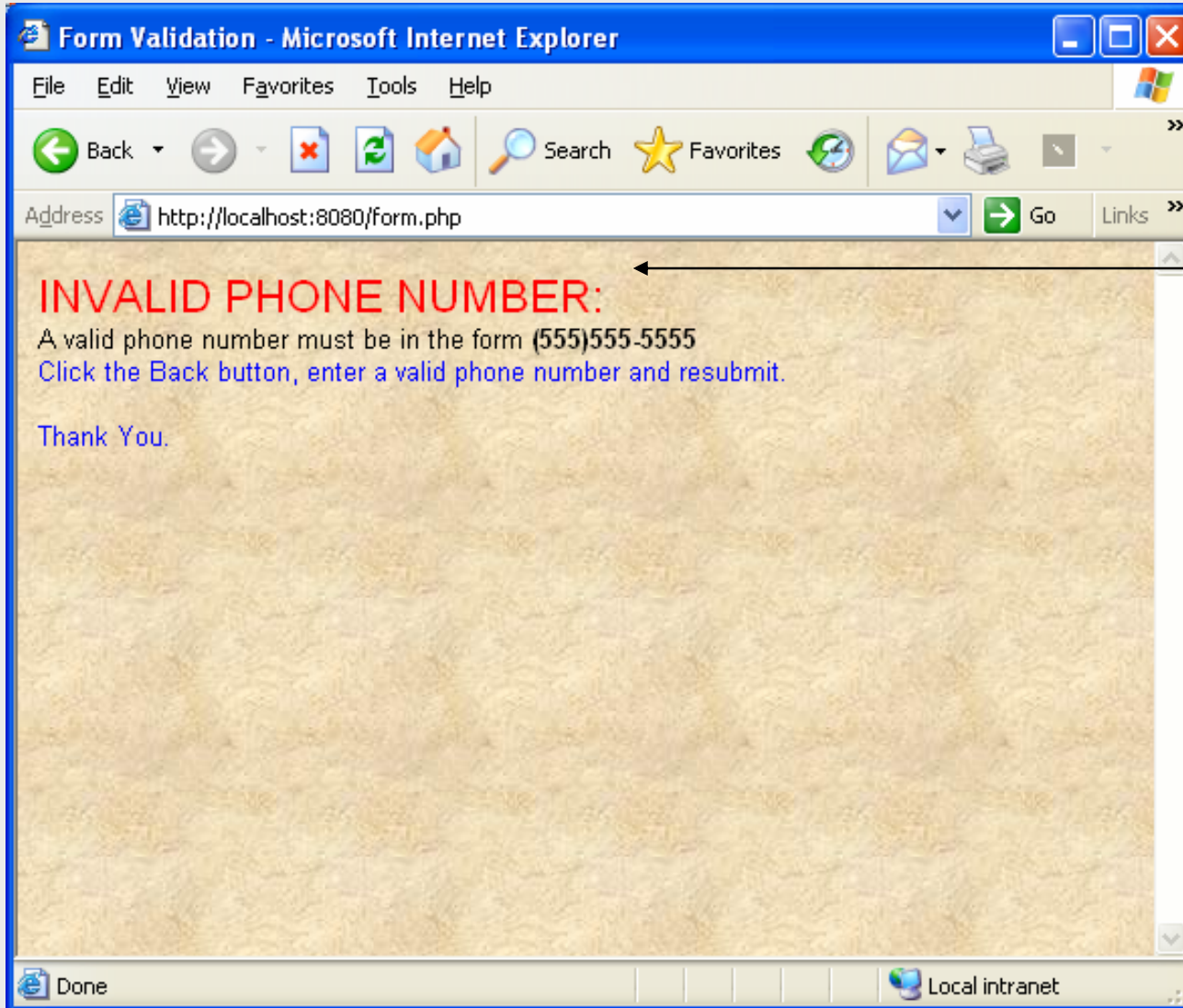
Windows XP Windows 2000 Windows 98

Linux Other

Register

User enters an improperly formatted telephone number in the form.





form.php issues error regarding improperly formatted telephone number.



How the Form Example Works

- The `action` attribute of the form element, indicates that when the user clicks the `Register` button, the form data will be posted to `form.php` for processing.
- Using `method = "post"` appends the form data to the browser request that contains the protocol (i.e., HTTP) and the requested resource's URL. Scripts located on the web server's machine (or accessible through the network) can access the form data sent as part of the request.
- Each of the form's input fields are assigned a unique name. When `Register` is clicked, each field's name and value are sent to the web server.
- Script `form.php` then accesses the value for each specific field through the global array `$_POST`.



How the Form Example Works (cont.)

- The superglobal arrays are associative arrays predefined by PHP that hold variable acquired from the user input, the environment, or the web server and are accessible in any variable scope.
 - If the information from the form had been submitted via the HTTP method `get`, then the superglobal array `$_GET` would contain the name-value pairs.
- Since the HTML form and the PHP script “communicate” via the name-value pairs, it is a good idea to make the XHTML object names meaningful so that the PHP script that retrieves the data is easier to understand.



Register_globals

- In PHP versions 4.2 and higher, the directive `register_globals` is set to `Off` by default for security reasons.
- Turning off `register_globals` means that all variables sent from an XHTML form to a PHP document now must be accessed using the appropriate superglobal array (either `$_POST` or `$_GET`).
- When this directive was turned On, as was the default case in PHP versions prior to 4.2, PHP created an individual global variable corresponding to each form field.



Validation of Form Generated Data

- The form example illustrates an important concept in the validation of user input. In this case, we simply checked the validity of the format of the telephone number entered by the client user.
- In general, it is crucial to validate information that will be entered into database or used in mailing lists. For example, validation can be used to ensure that credit-card numbers contain the proper number of digits before the numbers are encrypted to a merchant.
- In this case, the form.php script is implementing the **business logic** or **business rules** for our application.



Pattern Matching in PHP

- For powerful string comparisons (pattern matching), PHP provides functions `ereg` and `preg_match`, which use regular expressions to search a string for a specified pattern.
- Function `ereg` uses **Portable Operating System Interface (POSIX) extended regular expressions**.
 - POSIX-extended regular expressions are a standard to which PHP regular expression conform.
- Function `preg_match` provides **Perl-compatible regular expressions**.
- Perl-compatible regular expressions are more widely used than POSIX regular expressions. PHP's support for Perl-compatible regular expressions eases migration from Perl to PHP. The following examples illustrate these concepts.



expression.php - Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- expression.php -->
<!-- Using regular expressions -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title>Regular expressions</title>
  </head>
  <body>
    <?php
      $search = "Now is the time";
      print( "Test string is: '$search'<br /><br />" );
      // call function ereg to search for pattern 'Now' in variable search
      if ( ereg( "Now", $search ) )
        print( "String 'Now' was found.<br />" );

      // search for pattern 'Now' in the beginning of the string
      if ( ereg( "^Now", $search ) )
        print( "String 'Now' found at beginning of the line.<br />" );

      // search for pattern 'Now' at the end of the string
      if ( ereg( "Now$", $search ) )
        print( "String 'Now' was found at the end of the line.<br />" );
```

^ matches at beginning
of a string

\$ matches at end of a
string



```
// search for any word ending in 'ow'
if ( ereg( "[[:<:]]([a-zA-Z]*ow)[[:>:]]", $search,
    $match ) )
    print( "Word found ending in 'ow': " .
        $match[ 1 ] . "<br />" );

// search for any words beginning with 't'
print( "Words beginning with 't' found: " );

while ( eregi( "[[:<:]](t[[:alpha:]]+)[[:>:]]",
    $search, $match ) ) {
    print( $match[ 1 ] . " " );

    // remove the first occurrence of a word beginning
    // with 't' to find other instances in the string
    $search = ereg_replace( $match[ 1 ], "", $search );
}

print( "<br />" );
?>
</body>
</html>
```

Uses a regular expression to match a word ending in "ow".



Output From `expression.php` - Example

