

# PHP 03

Coding Style

Jump Menus (JavaScript)

Creating Memory

Cookies

Writing to a file

Including Code

Simple String Operations

Based on Rasmus Lerdorf & Kevin Tatroe: Programming PHP. Sebastopol: O'Reilly, 2002.

and <http://php.net>

W05/161B/Sauter

# PHP Programming Styles [E resources/PHP/style]

There are various possibilities for mixing HTML and PHP code. If the page contains predominantly HTML tags with only some dynamic PHP elements, the **XML programming style** is recommended to use. This way, the `<?php ... ?>` construct is used whenever a dynamic PHP element is required.

The same result can be achieved using the `<?php ... ?>` construct only once by **echoing** out all content, both dynamic and static. This is appropriate for pages using predominantly PHP code, e.g. to process forms, calculate, etc. Styles can be mixed, consistency however is highly recommended.

Remember that PHP can be used not only to generate HTML content, JavaScript, CSS, even pdfs and images can be built by PHP at runtime.

# XML Style [E resources/PHP/style]

```
<html>
<head>
  <title>PHP Coding XML Style</title>
</head>

<body>
  <form action="next.php" method="post">
    <input name="check1" type="checkbox" value="<?php echo $Check1value; ?>">
    <input name="text1" type="text" value="<?php echo $text1; ?>" size="20">
    <input name="submit" type="submit" value="send it">
  </form>
</body>
</html>
```

# Echoing Content [E resources/PHP/style]

```
<html>
<head>
  <title>PHP Coding Style</title>
</head>

<body>
  <?php
  echo '
  <form action="next.php" method="post">
    <input name="check1" type="checkbox" value="'; echo $Check1value; echo '">
    <input name="text1" type="text" value="'; echo $text1; echo '" size="20">
    <input name="submit" type="submit" value="send it">
  </form>
  ';
  ?>
</body>
</html>
```

# Echoing Content [E resources/PHP/style]

```
<html>
<head>
  <title>PHP Coding Style</title>
</head>

<body>
  <?php
  echo '
  <form action="next.php" method="post">
    <input name="check1" type="checkbox" value="'. $Check1value. '">
    <input name="text1" type="text" value="'. $text1. '" size="20">
    <input name="submit" type="submit" value="send it">
  </form>
  ';
  ?>
</body>
</html>
```

# JavaScript: Jump Menu

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script language="JavaScript">
<!--
    function MM_jumpMenu(targ,selObj,restore) {
        eval(targ+".location='"+selObj.options[selObj.selectedIndex].value+"'");
        if (restore) selObj.selectedIndex=0;

        // the JavaScript function eval() converts a string into a statement
    }
//-->
</script>
</head>
<body bgcolor="#FFFFFF">
<form name="form1">
    <p><font face="Arial, Helvetica, sans-serif" size="7">here</font></p>
    <p>
        <select name="menu1" onChange="MM_jumpMenu('parent',this,1)">
            <option value="here.html" selected>here</option>
            <option value="there.html">there</option>
        </select>
    </p>
</form>
</body>
</html>
```

# Creating Memory

Dynamic web pages are built at runtime. User input can be processed and direct feedback returned.

In order to memorize data provided by users, a method to store data is required. Besides databases, there are two other ways to store data:

Cookies

External files (e.g. text, html, etc.)

# Cookies [E resources/PHP/cookies]

Cookies are a mechanism for storing data in the remote browser. They are normally used to track or identify returning users.

A cookie requires a unique **name**, a **value**, an **expiration date** (if not set, the cookie expires when quitting the browser), a **path**, and a **domain** (only cookies from that domain under this path will be returned)

You can set cookies using the **setcookie()** function.

Cookies are part of the HTTP header, so **setcookie()** must be called before any output is sent to the browser or you will get an error message.

# Setting Cookies [E resources/PHP/cookies]

```
<?php
```

```
    setcookie ("161B_email", "your@email.com", time()+(60*60*24*10));  
    setcookie ("161B_cookie_date", strftime("%Y-%m-%d"), time()+(60*60*24*10));
```

```
?>
```

note: The function `time ()` returns the current time measured in the number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT).

# Reading Out Cookies [E resources/PHP/cookies]

The cookies are store in the array `$_COOKIE`. The key to access the value of the cookie is the cookie name.

```
<?php
    // Print an individual cookie
    echo $_COOKIE["161B_email"];
    echo $_COOKIE["161B_cookie_date"];
    echo "<br>";

    // Another way to debug/test is to view all cookies
    print_r($_COOKIE);

    /* the print_r () function prints human-readable
       information about a variable */
?>
```

# Writing into and reading from a file

The method of writing to an external file can be an alternative for specific purposes. An example would be a simple text log file, often used for guest book applications.

Depending on the data format of the external file generated by PHP, it can also store XML, variable and values etc. and share the data with other dynamic pages or Flash movies.

The PHP function `include ()` allows to include the content of a specified file into an PHP page. E.g.

```
include 'variables.php';  
include 'header.html';  
include 'javaScript.js';  
include 'guestbook.txt';
```

# PHP functions

`fopen ()`

Opens a specified file

`fwrite ()`

Used to read and write into the file. Different modes define the way how the file is handled. (next slide)

`fclose ()`

Closes the file

# fopen() modes

'r': Open for reading only; place the file pointer at the beginning of the file.

'r+': Open for reading and writing; place the file pointer at the beginning of the file.

'w': Open for writing only; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.

'w+': Open for reading and writing; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.

'a': Open for writing only; place the file pointer at the end of the file. If the file does not exist, attempt to create it.

'a+': Open for reading and writing; place the file pointer at the end of the file. If the file does not exist, attempt to create it.

'x': Create and open for writing only; place the file pointer at the beginning of the file. If the file already exists, the fopen() call will fail by returning FALSE and generating an error. If the file does not exist, attempt to create it. PHP 4.3.2 and later, and only works for local files.

'x+': Create and open for reading and writing; place the file pointer at the beginning of the file. If the file already exists, the fopen() call will fail by returning FALSE and generating an error. If the file does not exist, attempt to create it. PHP 4.3.2 and later, and only works for local files.

## [E resources/PHP/guestbook]

```
<HTML>
<HEAD>
  <TITLE>Simple Guestbook</TITLE>
</HEAD>
<BODY>
  <FORM ACTION="guestbook.php" METHOD="post">
    <P>Date</P>
    <INPUT TYPE="text" NAME="datum" VALUE="">
    <P>Text</P>
    <TEXTAREA NAME="text" COLS="72" ROWS="5"></textarea><BR>
    <INPUT TYPE="submit" VALUE="send">
  </FORM>

  <?php
  // defining the file where the data is stored. Check permissions!
  $source = "data.txt";
  // checking if the date has been provided
  if(isset($text)) {
    // opening the file, "a" stands for
    $file = fopen($source,"a");
    fwrite($file,"<BR>".$datum."<BR>".$text."<BR><BR>");
    fclose($file);
  } else {
    echo"please enter your text.";
  }
  include($source);
  ?>
</BODY>
</HTML>
```

# String Interpolation

Note: Double quotes and single quotes:

If a string is defined in a double quote it is subject to variable interpolation, which means the process of replacing the variable names in the string with the values of those variables.

```
$who = 'Daniel';  
$where = 'here';  
echo "$who was $there";
```

Daniel was here

```
echo '$who was $there';  
$who was $where
```

# Manipulating Strings - substr()

```
$name = 'Fred Flintstone';  
echo (substr($name, 6, 4));
```

lint

```
echo (substr($name, 11));
```

tone

# substr\_replace ()

```
$greeting = "good morning folks";  
$farewell = substr_replace($greeting, "bye", 5, 7);  
good bye folks
```

If you use the length value of 0 (last parameter), you can insert without deleting

# Searching Strings: strpos ()

strpos () function finds the first occurrence of a small string in a larger string

strrpos () finds the last occurrence of a character in a string, e.g.:

```
$record = "Fred,Flintstone,35,Wilma";  
$pos = strrpos($record, ","); // find last comma  
echo ("The last comma in the record is at $pos");
```

The last comma in the record is at position 18

see also: strstr(), strchr(), strspn(), strcspn()

# Connecting Strings

```
$my_string = "The beginning ";  
$my_string .= "of the end";  
echo $my_string;
```

The beginning of the end

# Random

Random functions are used to archive variety. Use it wisely and try to balance the outcome by limiting the range of possibilities.

```
echo rand(5, 15);
```

The first parameter defines the minimum, the second parameter the maximum value.