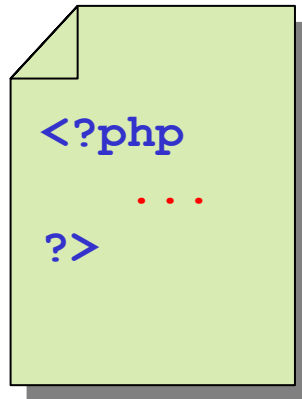


COSC 2206 Internet Tools

Simple PHP Web Applications And Server Environment





Simple PHP WEB applications

Hello world and date scripts

PHP info

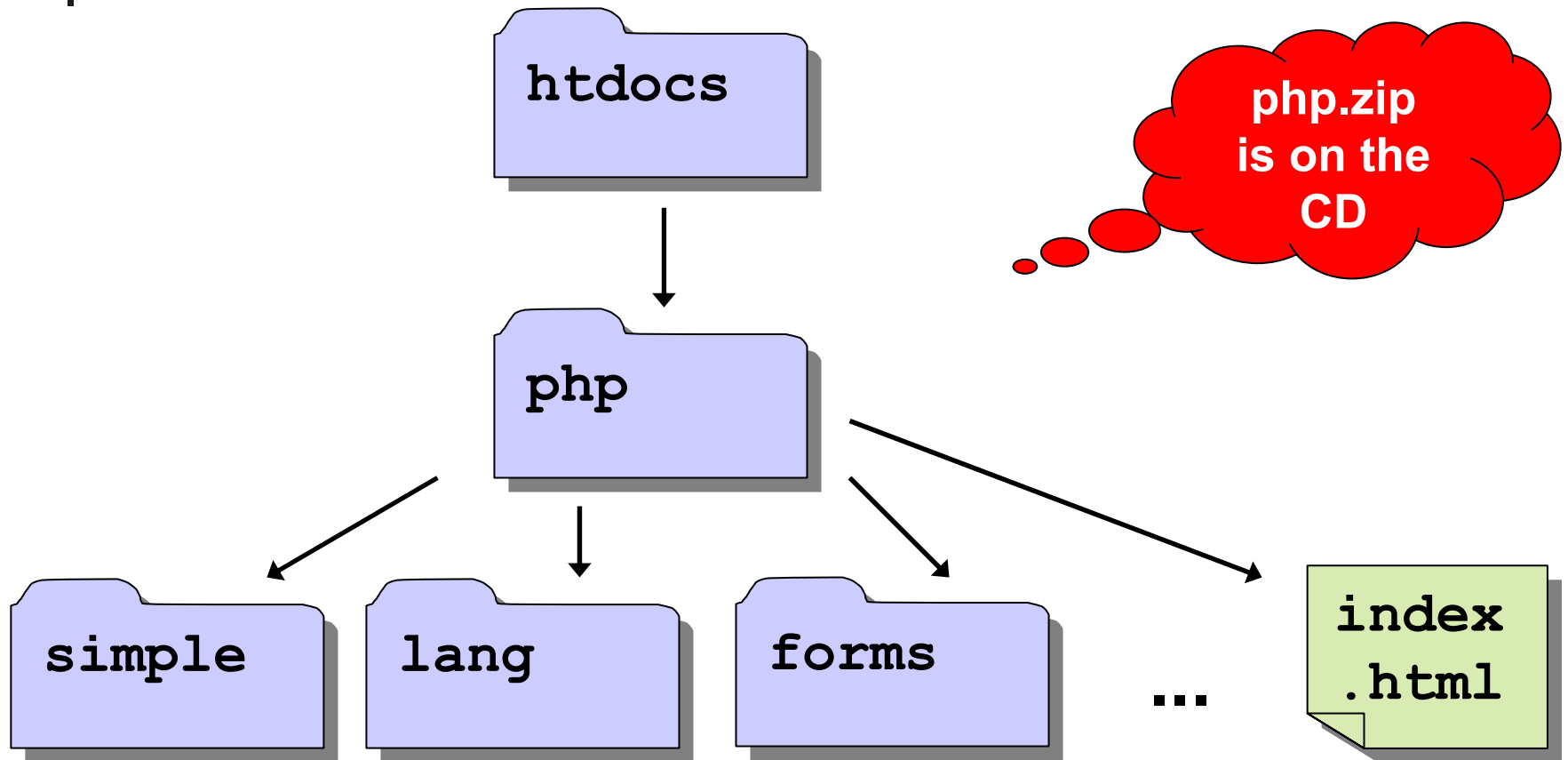
The server environment

External global variables

Form processing

Page counts

Web Directory Structure





Hello Script (HTML Format)

```
<html>
<head><title>Hello Script</title></head>
<body>
<?php echo "<h1>Hello PHP World!</h1>"; ?>
</body>
</html>
```

- Try <http://localhost/php/simple/hello.php>
- Try <http://localhost/php/index.html> or just <http://localhost/php/> (/ at end is necessary)



Hello Script (Plain Format)

```
<?php
header("Content-type: text/plain");
echo "<h1>Hello PHP World!</h1>"; ?>
?>
```

default type
is text/html

- Try http://localhost/php/simple/hello_plain.php
- Try <http://localhost/php/index.html> or just <http://localhost/php/> (/ at end is necessary)



Date Script

```
<html>
<head><title>Date Script</title></head>
<body>
<h1>Displaying the date and time</h1>
<?php
    echo "The date is ", date("jS F Y, h:i A"), "\n";
?>
</body>
</html>
```

see PHP docs
for more formats

- <http://localhost/php/simple/date.php>
 - [view script](#)



Display PHP Info

```
<html>
<head><title>PHP Info</title></head>
<body>
<?php phpinfo () ; ?>
</body>
</html>
```

- <http://localhost/php/simple/info.php>
 - [view script](#)



Server Environment

- A PHP script is provided with a superglobal associative array called `$_SERVER` of variables that describe the server environment:
- Example:
 - `$_SERVER['DOCUMENT_ROOT']`
 - this is a string giving the document root directory, for example
 - `c:\Apache\htdocs`



Global Environment

- **\$GLOBALS** is a superglobal associative array whose keys are the names of the global arrays that are available in PHP.
- The corresponding values are these arrays
- Example:
 - **\$GLOBALS [' _SERVER ']**
 - This is the **\$_SERVER** array
- You probably won't use **\$GLOBALS** much



Display \$_SERVER Variables

- Write a simple script that displays the `$_SERVER` array as an HTML table:
 - Write a function to display an associative array as an HTML table
 - Write a PHP script `server.php` that uses it to make an HTML page containing the array.
- <http://localhost/php/simple/server.php>
 - [view script](#)



server.php (1)

- A function to display an associative array:
put it in file `display_table.php`

```
<?php function display_table($a)
{   echo '<table border="1" cellpadding="2">', "\n";
    foreach ($a as $key => $value)
    {   echo "<tr>\n";
        echo "<td><b>$key<b></td>\n<td>$value</td>\n";
        echo "</tr>\n";
    }
    echo "</table>\n";
} ?>
```

[view source](#)



server.php (2)

- Use it in `server.php`:

```
<?php
    include ("display_table.php") ;
?>
<html>
<head><title>Server Variables</title></head>
<body>
<h1>Server Variables</h1>
<?php
    ksort($_SERVER) ;
    display_table($_SERVER) ;
?>
</body>
</html>
```



Partial output of server.php

Server Variables

COMSPEC	C:\WINDOWS\COMMAND.COM
DOCUMENT_ROOT	c:/apache/htdocs
GATEWAY_INTERFACE	CGI/1.1
HTTP_ACCEPT	*/*
HTTP_ACCEPT_ENCODING	gzip, deflate
HTTP_ACCEPT_LANGUAGE	en-us
HTTP_CONNECTION	Keep-Alive
HTTP_HOST	localhost



Much More



Display \$GLOBALS Variables

- Write a simple script that displays the associative arrays in \$GLOBALS as HTML tables
 - use the **display_tables** function in a script called `globals.php`
- <http://localhost/php/simple/globals.php>
 - [view source](#)



globals.php (1)

```
<?php
    include ("display_table.php") ;
?>
<html>
<head><title>Values in $GLOBALS array</title></head>
<body>
<h1>Values in $GLOBALS array</h1>
<?php
    // See next slide
?>
</body>
</html>
```



globals.php (2)

```
foreach ($GLOBALS as $key => $value)
{
    echo "<h2>$key</h2>";
    if (! empty($value))
    {
        display_table($value);
    }
    else
    {
        echo "$key is empty";
    }
}
```




Partial Output of globals.php

Values in \$GLOBALS array

HTTP_POST_VARS

HTTP_POST_VARS is empty

_POST

_POST is empty

HTTP_GET_VARS

HTTP_GET_VARS is empty



More



The GET Method

- A script can be invoked using a query string of encoded name-value pairs
- This can be done explicitly by including a query string at the end of the URL
- Example:
 - `?name1=val1&name2=val2...`
- Alternatively the query string can be provided by a form using the GET method



The Query String

- Try the following link
 - <http://localhost/php/simple/server.php?name=Fred>
- Look for the **QUERY_STRING** table row
- This shows that the query string is made available to the PHP script as
 - `$_SERVER['QUERY_STRING']`



\$_GET and \$_REQUEST arrays

- The name value pairs are made available to the script using the global **\$_GET** or **\$_REQUEST** arrays.
- Example:
 - if one of the name value pairs is **name=Fred** then **\$_REQUEST['name']** and **\$_GET['name']** both have the value 'fred'
- To see this try the following link and look for the **_GET** and **_REQUEST** tables
 - <http://localhost/php/simple/globals.php?name=Fred>



THE POST method

- There are limitations to the GET method:
 - long strings may exceed browser limitations
 - Name value pairs are visible to anyone
 - However, URL's can be bookmarked
- POST method doesn't have these limitations.
- Forms can specify either GET or POST
- In either case the name value pairs are always available in the `$_REQUEST` array



GET and POST summary

- PHP scripts interact with the client using the GET and POST methods and the superglobal associative arrays:
 - **`$_GET`** contains any name-value pairs sent using the GET method.
 - **`$_POST`** contains any name-value pairs sent using the POST method.
 - **`$_REQUEST`** contains all name-value pairs.
 - **`$_COOKIE`** contains cookie name-value pairs



Determining the method

- A script can determine whether it was invoked using the GET or PUT method
- Use `$_SERVER['REQUEST_METHOD']`
- It will have the value 'GET' or 'PUT' so a test can be made

```
if ( $_SERVER['REQUEST_METHOD'] === 'GET' )  
{  
    ...  
}
```



Get method example

```
<html>
<head><title>The GET method</title></head>
<body><h1>The GET method</h1>
<?php
    $request = $_SERVER['REQUEST_METHOD'];
    $query = $_SERVER['QUERY_STRING'];
    echo "The request method is $request<br>";
    if ($request === 'GET')
    {   echo "The query string is $query<br/>";
    }
?></body></html>
```

- <http://localhost/php/simple/get.php?a=b>
 - [view source](#)



Form processing

- Browsers communicate with server-side scripts using HTML forms
- When you fill out a form you are specifying a number of name-value pairs.
- When you submit the form the name-value pairs are sent to the server-side script
- In the case of PHP these values are available to the script in the arrays `$_GET`, `$_POST`, `$_REQUEST`



Older versions of PHP

- Older versions of PHP (before 4.1) directly assigned form variable names to PHP variables of the same name
- This could be exploited (security hole)
- Now we should always use the superglobal arrays like `$_GET`, `$POST`, `$_REQUEST`
- This is the default with `register_globals` turned off in `php.ini` (this is now the default)



Form processing using GET

- The form has the following structure

```
<form action="doform.php" method="GET">  
.....  
</form>
```

- When the form is submitted the script called **doform.php** will be called using the GET method



Link processing using GET

- A link of the form

```
<a href="script.php?name=fred&age=24">..</a>
```

can be processed using the GET method and the name value-pairs can be accessed using

- `$_GET['name']` which has the value `'fred'`
- `$_GET['age']` which has the value `24`



Form processing using POST

- The form has the following structure

```
<form action="doform.php" method="POST">  
  
.....  
  
</form>
```

- When it is submitted the script called **doform.php** will be called using the POST method



Input field example

- `<input type="text" name="firstname" value="fred" />`
- The value is omitted if there is no default value
- When the form containing this element is submitted the value is available as
 - `$_REQUEST['firstname']`



form1_get.html

```
<html>
<head><title>Processing form with GET</title></head>
<body>
<h1>Processing form with GET</title></h1>
<form action="doform1.php" method="GET">
First name: <input type="text" name="firstname"><br>
Last name: <input type="text" name="lastname">
<p><input type="submit" name="button"
        value="Submit Name"></p>
</form>
</body>
</html>
```

http://localhost/php/forms/form1_get.html



form1_post.html

```
<html>
<head><title>Processing form with POST</title></head>
<body>
<h1>Processing form with POST</title></h1>
<form action="doform1.php" method="POST">
First name: <input type="text" name="firstname"><br>
Last name: <input type="text" name="lastname">
<p><input type="submit" name="button"
        value="Submit Name"></p>
</form>
</body>
</html>
```

http://localhost/php/forms/form1_post.html



doform1.php

```
<?php
    // get the submitted form values
    $first_name = $_REQUEST['firstname'];
    $last_name = $_REQUEST['lastname'];
?>

<html>
<head><title>Form Results</title></head>
<body>
<h1>Form Results</title></h1>
<?php echo "Hello $first_name $last_name\n"; ?>
</body>
</html>
```

[view source \(forms/doform1.php\)](#)



Try it directly with a URL

- The doform1.php script can be requested using a URL instead of a form
- For example, try
- <http://localhost/php/forms/doform1.php?firstname=Fred&lastname=Jones>



Form Generation with PHP

- In this example the form is generated by a static HTML page.
- A PHP script can display both the HTML page containing the form and the HTML page in response to the form submission
- It knows which page to display by detecting whether the submit button has been clicked or not:



Form processing logic

- If the input element for the button is

```
<input type="submit" name="button"  
value = "Submit Name" />
```

- then the PHP script logic has the form

```
if (isset($_REQUEST['button']))  
{  
    display_output_page();  
}  
else  
{  
    display_form_page();  
}
```



display_form_page()

```
<?php
function display_form_page ()
{   $self = $_SERVER['PHP_SELF'];
    ?><html>
    <head><title>Forms, version 2</title></head>
    <body><h1>Forms, version 2</title></h1>
    <form action="<?php echo $self ?>" method="POST">
    First name: <input type="text" name="firstname"><br>
    Last name: <input type="text" name="lastname">
    <p><input type="submit" name="button"
        value="Submit Name"></p>
    </form>
    </body></html><?php
}
```



doform2.php

```
<?php
if (isset($_REQUEST['button']))
{   display_output_page();
}
else
{   display_form_page();
}

<?php display_form_page goes here ?>
<?php display_output_page goes here ?>

?>
```

[view source forms/doform2.php](#)

<http://localhost/php/forms/doform2.php>



Using PHP_SELF

- Locate the script using
 - `$self = $_SERVER['PHP_SELF'];`
- This gives the script location relative to the document root. For example
 - `/php/forms/doform2.php`
- To see this view the html before submitting the form.



Using Multiple Buttons (1)

- Give the buttons different names:

```
<input type="submit" name="button1"  
      value = "Submit Name 1" />  
<input type="submit" name="button2"  
      value = "Submit Name 2" />
```

- Check which one was clicked

```
if ( isset($_REQUEST['button1']) ||  
      isset($_REQUEST['button2'])  
{ display_output_page();  
} else  
{ display_form_page();  
}
```




Using Multiple Buttons (2)

```
$button1 = isset($_REQUEST['button1'])  
           ? $_REQUEST['button1'] : '';  
$button2 = isset($_REQUEST['button2'])  
           ? $_REQUEST['button2'] : '';
```

```
if ($button1 != '')  
{ // process button1 click event here  
}  
if ($button2 != '')  
{ // process button2 click event here  
}
```

[view script forms/doform3.php](#)

<http://localhost/php/forms/doform3.php>



Error Checking

- Names contain only letters, hyphens, and ' '

```
<?php function validate_form()
{   $first_name = trim($_REQUEST['firstname']);
    $last_name = trim($_REQUEST['lastname']);
    $error = '';
    $reg_exp = '/^[a-zA-Z\-\ ']+$/' ;
    if (! preg_match($reg_exp, $first_name))
    {   $error .= "First name is invalid ...";
    }
    if (! preg_match($reg_exp, $last_name))
    {   $error .= "Last name is invalid ...";
    }
    return $error;
} ?>
```



Sticky input fields

- If there are errors in a form it needs to be sent to the user to be corrected so we can supply the values that the user entered as defaults.

```
First name: <input type="text" name="firstname"
            value="<?php echo $first_name ?>">
Last name: <input type="text" name="lastname"
           value="<?php echo $last_name ?>">
```

[view script forms/doform4.php](#)

<http://localhost/php/forms/doform4.php>



Checkbox group (1)

- Use array notation [] in the form to indicate a group of checkboxes

Which wines do you like?

Shiraz <input type="checkbox" name="wines[]" value="shiraz">

Merlot <input type="checkbox" name="wines[]" value="merlot">

Chianti <input type="checkbox" name="wines[]" value="chianti">



Checkbox group (2)

- Determining which boxes are checked

```
if (! empty($wines))  
{  
    echo "Wines selected were ";  
    foreach ($wines as $wine)  
    {  
        echo "$wine ";  
    }  
}  
else  
    echo "No wines were selected";
```

[view script forms/doform5.php](#)

<http://localhost/php/forms/doform5.php>



Drop down list

- Use select and option. The selected attribute indicates which choice has been made. If no choice is made the first one is selected

Select your favourite wine from the list:

```
<select name="wine">  
  <option selected="selected">Shiraz</option>  
  <option>Merlot</option>  
  <option>Chianti</option>  
  <option>Other</option>  
</select>
```

[view script forms/doform6.php](#)

<http://localhost/php/forms/doform6.php>



Sticky Checkboxes (1)

- Write a function:

```
<?php
function check($group, $val)
{
    if (is_array($group) and in_array($group,$val))
    {
        echo 'checked = "checked" ';
    }
}
?>
```



Sticky Checkboxes (2)

- Use the check function as follows

```
Shiraz <input type="checkbox" name="wines[]"  
      value="shiraz" <?php check($wines,"shiraz")?>>  
Merlot <input type="checkbox" name="wines[]"  
      value="merlot" <?php check($wines,"merlot")?>>  
Chianti <input type="checkbox" name="wines[]"  
      value="chianti" <?php check($wines,"chianti")?>>
```

[view script forms/doform7.php](#)

<http://localhost/php/forms/doform7.php>



Sticky Radio Buttons (1)

- Write a function:

```
<?php
function check($group, $val)
{
    if ($group === $val)
    {
        echo 'checked = "checked" ';
    }
}
?>
```



Sticky Radio Buttons (2)

- Use the check function as follows

```
Shiraz <input type="radio" name="wine"  
      value="shiraz" <?php check($wine,"shiraz") ?>>  
Merlot <input type="radio" name="wine"  
      value="merlot" <?php check($wine,"merlot") ?>>  
Chianti <input type="radio" name="wine"  
      value="chianti" <?php check($wine,"chianti") ?>>
```

[view script forms/doform8.php](#)

<http://localhost/php/forms/doform8.php>



Sticky Drop Down List (1)

- Write a function:

```
<?php
function check($group, $val)
{
    if ($group === $val)
    {
        echo 'selected = "selected"';
    }
}
?>
```



Sticky Drop Down List (2)

- Use the check function as follows

```
<select name="wine">
<option <?php check($wine, "Shiraz") ?>>Shiraz</option>
<option <?php check($wine, "Merlot") ?>>Merlot</option>
<option <?php check($wine, "Chianti") ?>>Chianti</option>
<option <?php check($wine, "Other") ?>>Other</option>
</select>
```

[view script forms/doform9.php](#)

<http://localhost/php/forms/doform9.php>



Generic form script (1)

```
<?php // A simple script to display parameters ?>
<html>
<body><h1>Form post and get parameters</h1>
<?php
    if (empty($_REQUEST))
        echo "No parameters were sent to this script";
    while(list($key, $value) = each($_REQUEST))
    {   if (is_array($value))
        $value = implode(",", $value);
        echo "$key = $value<br>";
    }
?></body></html>
```

[view script forms/general.php](#)

<http://localhost/php/forms/general.php>



Generic form script (2)

- Try the script with the following query string
 - ?name=fred&age=24&wines[]=shiraz&wines[]=merlot&wines[]=chianti
- [http://localhost/php/forms/general.php?name=fred&age=25&wines\[\]=shiraz&wines\[\]=merlot&wines\[\]=chianti](http://localhost/php/forms/general.php?name=fred&age=25&wines[]=shiraz&wines[]=merlot&wines[]=chianti)



Testing Perl Regex

Testing Perl style regular expressions

NOTE:

Regular expression should include the Perl style delimiters
optionally followed by an `i` for a case insensitive match.

Regular expression	<input type="text"/>
Target string	<input type="text"/>

Perform test

[view script forms/regexp.php](#)

<http://localhost/php/forms/regexp.php>



regexp.php (1)

```
<html>
<head>
<title>Testing Perl style regular expressions</title>

<style type="text/css">
.mono { font-family: "Courier New", monospace;
        font-weight: bold;
        font-size: medium;
        background-color: #CCCCCC
      }
</style>
</head>
<body>
```




regexp.php (2)

```
<h1>Testing Perl style regular expressions</h1>
```

```
<b>NOTE:</b><br />
```

```
Regular expression should include the Perl style  
delimiters<br />
```

```
optionally followed by an <code>i</code> for a case  
insensitive match.
```

```
<?php
```

```
    $self = $_SERVER['PHP_SELF'];
```

```
    $regexp = isset($_REQUEST['regexp']) ?
```

```
        $_REQUEST['regexp'] : '';
```

```
    $target = isset($_REQUEST['target']) ?
```

```
        $_REQUEST['target'] : '';
```

```
?>
```



regexp.php (3)

```
<form action="<?php echo $self ?>" method="POST">
<table border="1" cellpadding="5">
<tr>
  <td>Regular expression</td>
  <td><input class="mono" type="text" name="regexp"
    value="<?php echo $regexp ?>" size="25"/></td>
</tr>
<tr>
  <td>Target string</td>
  <td><input class="mono" type="text" name="target"
    value="<?php echo $target ?>" size="25"/></td>
</tr>
</table>
<p><input type="submit" name="test"
  value="Perform test" /></p>
</form>
```



regexp.php (4)

```
<?php // check for a match
    if ( isset($_REQUEST['test']) ) // button clicked
    {
        if ( preg_match($regexp, $target) )
        {
            echo "<p><b>It's a match</b></p>";
        }
        else
        {
            echo "<p><b>It's not a match</b></p>";
        }
    }
?>
</body>
</html>
```

[view script forms/regexp.php](view_script_forms/regexp.php)



Temp conversion

Fahrenheit to Celsius Conversion

Fahrenheit temperature:

Convert to Celsius

input
page

Fahrenheit to Celsius Conversion

115.00F is 46.11C [Another conversion](#)

output
page



Distinguishing GET and POST

- Distinguish between GET and POST methods to decide whether to display a form or use the results

```
$request_method = $_SERVER['REQUEST_METHOD'];  
if ($request_method == 'GET')  
{  
    // display form to get fahrenheit temperature  
}  
elseif ($request_method == 'POST')  
{  
    // use submitted temperature to convert to  
    // celsius and display the result  
}
```



temp1.php (1)

```
<html><head>
<title>Fahrenheit to Celsius Conversion</title>
<body>
<h1>Fahrenheit to Celsius Conversion</h1>
<?php
    $request_method = $_SERVER['REQUEST_METHOD'];
    $self = $_SERVER['PHP_SELF'];
    if ($request_method == 'GET')
    { ?>
        <form action="<?php echo $self ?>" method="POST">
        Fahrenheit Temperature:
        <input type="text" name="fahrenheit" />
        <p><input type="submit" name="convert"
            value="Convert to Celsius" /></p>
        </form>
```

temp1.php (2)

```
<?php
}
elseif ($request_method == 'POST')
{
    // Form was submitted so do calculations

    $fahr = $_POST['fahrenheit']; // or $_REQUEST
    $celsius = ($fahr - 32) * (5.0/9.0);
    printf("%.2fF is %.2fC", $fahr, $celsius);
    echo " <a href=\"\$self\">Another conversion</a>";
}
?>
</body></html>
```

[view script forms/temp1.php](view_script_forms/temp1.php)

<http://localhost/php/forms/temp1.php>



Another approach

- Another way to decide whether to display the form or do the calculations and display the results: use button name. It will be sent to the PHP script only if it is clicked.

```
if (isset($_REQUEST['convert'])) // button clicked
{
    // do the conversion and display results
}
else
{
    // display the form
}
```




temp2.php (1)

```
<html><head>
<title>Fahrenheit to Celsius Conversion</title>
</head><body>
<h1>Fahrenheit to Celsius Conversion</h1>
<?php
    $self = $_SERVER['PHP_SELF'];
    if (isset($_REQUEST['convert']))
    {
        $fahr = $_REQUEST['fahrenheit'];
        $celsius = ($fahr - 32) * (5.0/9.0);
        printf("%.2fF is %.2fC", $fahr, $celsius);
        echo " <a href=\"\$self\">Another conversion</a>"
    }
}
```

temp2.php (2)

```
else
{
?>
<form action="<?php echo $self ?>" method="POST">
Fahrenheit Temperature:
<input type="text" name="fahrenheit" />
<p><input type="submit" name="convert"
    value="Convert to Celsius" /></p>
</form>

<?php
}
?>
</body></html>
```

[view script forms/temp2.php](view_script_forms/temp2.php)

<http://localhost/php/forms/temp2.php>



Another approach

- Another way is to use an HTML page with some dynamic parts. The output is displayed on same page as the form.



temp3.php (1)

```
<?php $self = $_SERVER['PHP_SELF'];  
      $fahr = isset($_REQUEST['fahrenheit']) ?  
      $_REQUEST['fahrenheit'] : ''; ?>  
<html><head>  
<title>Fahrenheit to Celsius Conversion</title>  
</head><body>  
<h1>Fahrenheit to Celsius Conversion</h1>  
<form action="<?php echo $self ?>" method="POST">  
Fahrenheit temperature:  
<input type="text" name="fahrenheit"  
      value="<?php echo $fahr ?>" />  
<p><input type="submit" name="test"  
      value="Convert to Celsius" /></p>  
</form><p><?php result($fahr) ?></p>  
</body></html>
```



temp3.php (2)

```
<?php

// function to insert result into HTML page

function result($fahr)
{
    if (! ($fahr === ' '))
    {
        $fahr = $_REQUEST['fahrenheit']);
        $celsius = ($fahr - 32) * (5.0/9.0);
        printf("%.2fF is %.2fC", $fahr, $celsius)
    }
}

?>
```