

CENG420
Server side
Lecture 2

PHP

Handling user input



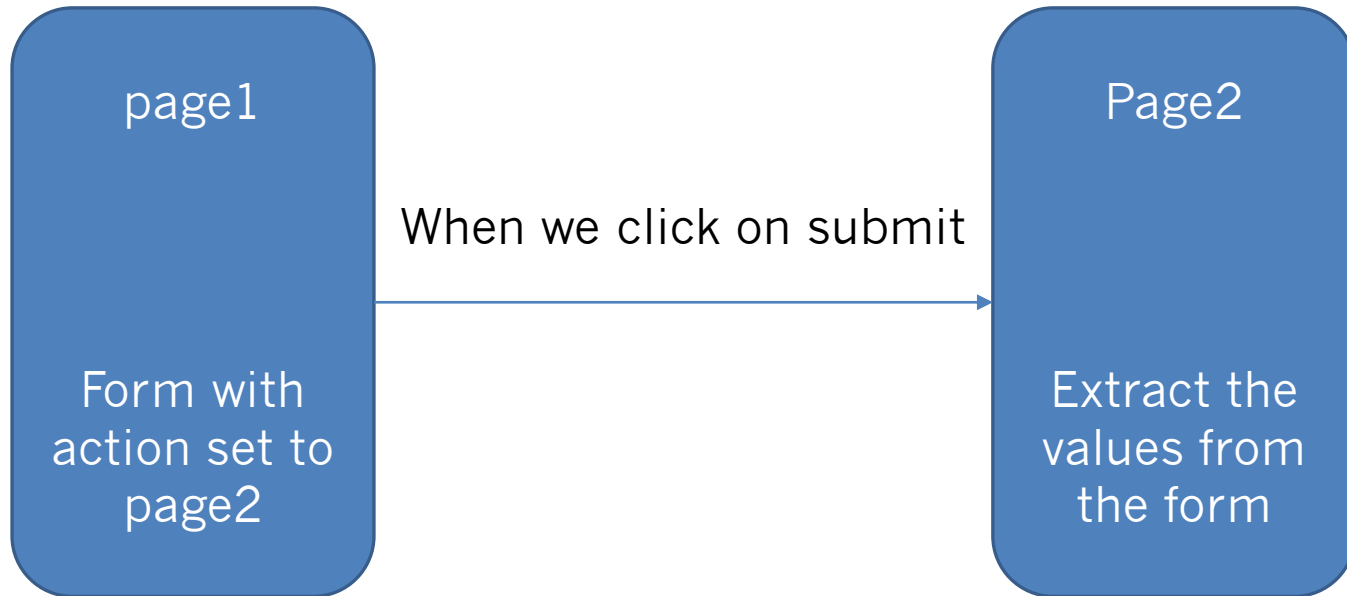
Why?

- The majority of websites have forms
- Users use forms to input values.
- Whether, you're filling an application online or you're commenting on facebook or posting to a forum.
- When the form is submitted, those values should be found somewhere and can be accessed!
- Hints: `$_POST`, `$_GET`, `$_FILES`,...

What really happens?

```
<form action="somepage.php" method="GET">
<input type="text" name="fname"/>
<input type="submit" value="next"/>
</form>
```

- Assume that the above html code is inside ***index.php*** or ***index.html*** page
- When the submit button is clicked, this form is sent to ***somepage.php*** (which is on the server)
- The code inside `somepage.php` should be able to extract the values that the user has input in the form. In our case, we have 1 input only (fname)



- Page 2 might contain html code or might not!
- It means, page 2 might be sent back to the client to see it or might do background jobs (like saving to DB) and then redirecting to another page that will be sent from the server back to client.

PHP helps you!

- It creates associative arrays ready for you to use when you submit a form.
- Access those arrays and read those values.
- **The index of those arrays are the “name” fields in the form**
- Some of them are editable → assign values instead of reading (covered later!)

PHP associative arrays are called AutoGlobals

Array	Description
<code>\$_COOKIE</code>	An array of values passed to the current script as HTTP cookies
<code>\$_ENV</code>	An array of environment information
<code>\$_FILES</code>	An array of information about uploaded files
<code>\$_GET</code>	An array of values from a form submitted with the “get” method
<code>\$_POST</code>	An array of values from a form submitted with the “post” method
<code>\$_REQUEST</code>	An array of all the elements in the <code>\$_COOKIE</code> , <code>\$_GET</code> , and <code>\$_POST</code> arrays
<code>\$_SERVER</code>	An array of information about the Web server that served the current script
<code>\$_SESSION</code>	An array of session variables that are available to the current script
<code>\$GLOBALS</code>	An array of references to all variables that are defined with global scope

How to extract them?

```
<form action="somepage.php" method="GET">  
<input type="text" name="fname"/>  
<input type="submit" value="next"/>  
</form>
```

- If the method of the form is GET, then the values input by the user are saved into the **\$_GET** array
- If the method was POST, the values are saved into the **\$_POST** array
- In our case, we can say `$n= $_GET['fname'];`

In brief

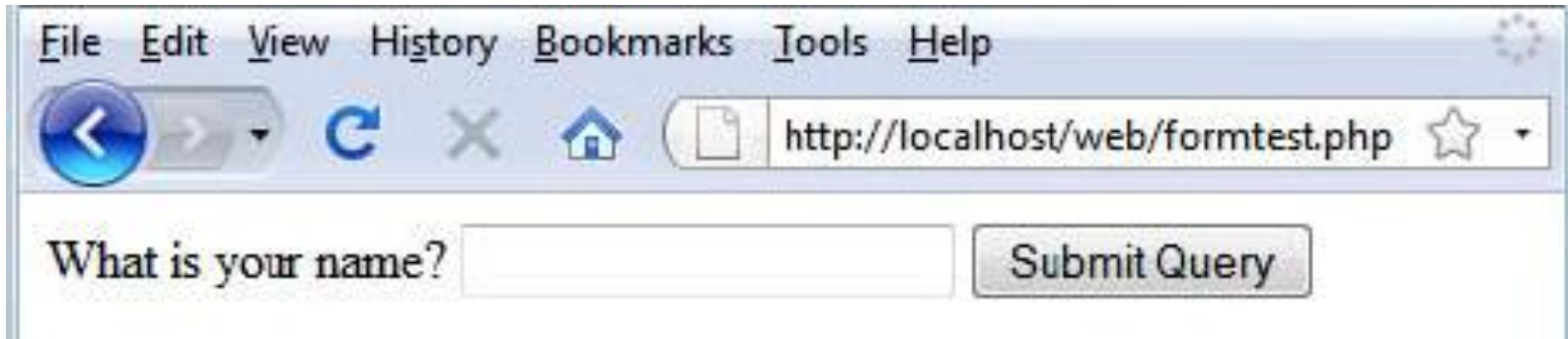
- Sometimes, we need to pass values from a page to another but it isn't any element of the form!
- Use “**hidden**” types and assign values to them:

```
<input type="hidden" name="delete" value="yes" />
```


The function isset()

- Can be used to test if the user has inputted anything that led to filling the `$_POST` or `$_GET` or `$_FILES`,.....
- EXAMPLE:
- `if(isset($_POST['delete']) && isset($_POST['fname']))`

A simple example, revisited!



```
echo <<<_END
```

```
<html> <head> <title>Form Test</title> </head>  
<body>
```

```
<form method="post" action="formtest2.php">
```

```
What is your name?
```

```
<input type="text" name="your_name" />
```

```
<input type="submit" value="Submit Query" />
```

```
</form> </body> </html>_END;
```



What if we want to show the name the user enters on the same page, and keep the form?

- Add a small PHP code to read this value from the `$_POST` array.
- Of course, we need to check if this value is set (i.e. if user enters anything) to make sure it's not first time we navigate to page or if we click on submit without entering anything.
- Solution next.

Solution

```
<?php // formtest2.php

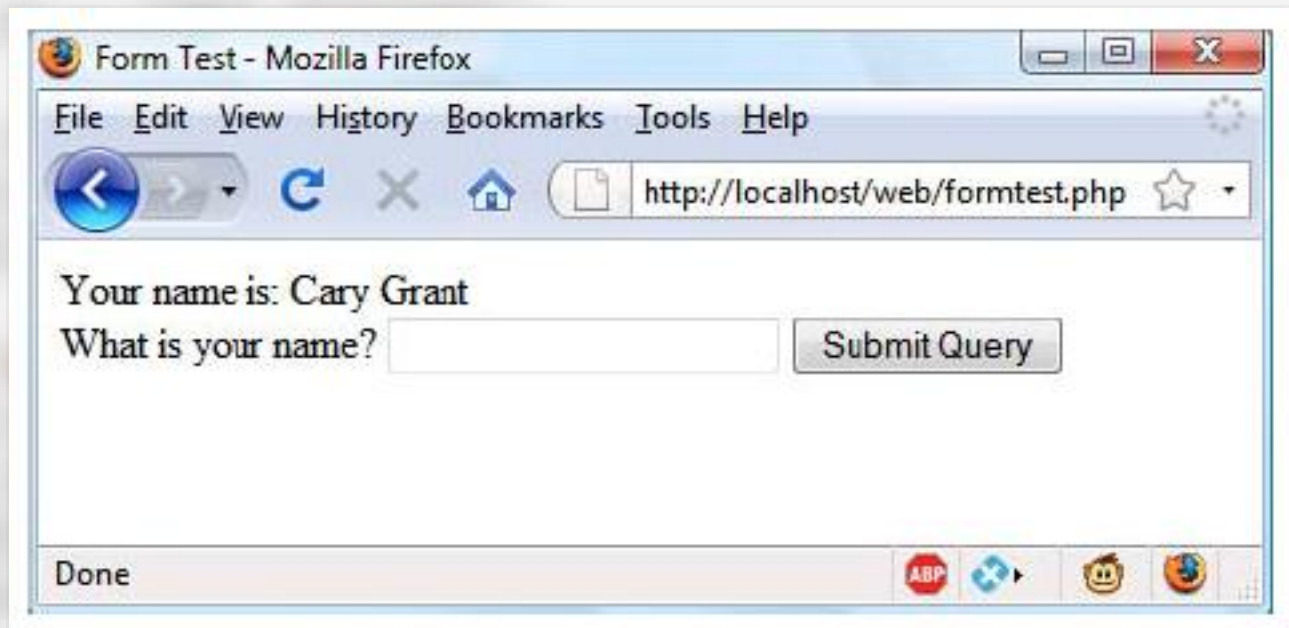
if (isset($_POST['your_name']))
    $n = $_POST['your_name'];
else
    $n = "(Not entered)";

echo <<<_END
<html> <head> <title>Form Test</title> </head>
<body>

Your name is: $n <br/>
<form method="post" action="formtest2.php">
What is your name?
<input type="text" name="your_name" />
<input type="submit" value="Submit Query" />

</form> </body> </html> _END; ?>
```

Output if the user enters “Cary Grant” then clicks on submit





1

Input Types

Text Boxes

```
<input type="text" name="name"  
size="size" maxlength="length"  
value="value" />
```

Text Area

```
<textarea name="name" cols="width"  
rows="height" wrap="type">  
This is some default text.  
</textarea>
```

TextArea (cont)

Table 11-1. The wrap types available in a textarea input

Type	Action
off	Text does not wrap and lines appear exactly as the user types them.
soft	Text wraps but is sent to the server as one long string without carriage returns and line feeds.
hard	Text wraps and is sent to the server in wrapped format with soft returns and line feeds.

Checkboxes

- This is what we are used to:

Vanilla

```
<input type="checkbox" name="ice" value="Vanilla"/>
```

Chocolate

```
<input type="checkbox" name="ice" value="Chocolate"/>
```

Strawberry

```
<input type="checkbox" name="ice" value="Strawberry"/>
```

What if the user click on several checkboxes?

- He/she is allowed of course to check several checkboxes.
- If you want to limit him/her to 1 choice, use radio buttons (old stuff!)
- What if we access `$_POST['ice']` now? (or `$_GET['ice']` if the method was GET?)

Vanilla ☐ Chocolate ☒ Strawberry ☒

Slight modification to the html code: add []

Vanilla

```
<input type="checkbox" name="ice[]" value="Vanilla" />
```

Chocolate

```
<input type="checkbox" name="ice[]" value="Chocolate" />
```

Strawberry

```
<input type="checkbox" name="ice[]" value="Strawberry" />
```



Now, we can do this:

```
$ice = $_POST['ice']
```

- What do you think the variable `$ice` hold now?
- (note that `$ice` could also be `$x`, `$y`,... we dont have to name it the same as the name of the checkboxes!!!)
- `$ice` now is an array! We can access `$ice[0]` and `$ice[1]` for example if the user has clicked on 2 boxes.

The possible values of the variable \$ice

One value submitted	Two values submitted	Three values submitted
<code>\$ice[0] => Vanilla</code>	<code>\$ice[0] => Vanilla</code>	<code>\$ice[0] => Vanilla</code>
<code>\$ice[0] => Chocolate</code>	<code>\$ice[1] => Chocolate</code>	<code>\$ice[1] => Chocolate</code>
<code>\$ice[0] => Strawberry</code>		<code>\$ice[2] => Strawberry</code>
	<code>\$ice[0] => Vanilla</code>	
	<code>\$ice[1] => Strawberry</code>	
	<code>\$ice[0] => Chocolate</code>	
	<code>\$ice[1] => Strawberry</code>	

Submit button

- 2 options: either an input with type=submit (covered before)
- Or an image:

```
<input type="image"  
name="submit" src="image.gif"  
>
```



2

Sanitizing inputs

Caution

- The first thing to remember is that regardless of what constraints you have placed in an HTML form to limit the types and sizes of inputs:
 - ***it is a trivial matter for a hacker to use their browser's View Source feature to extract the form and modify it to provide malicious input to your website.***

That's why

- We need to **sanitize** the inputs from the users:
- Inputs might contain escape characters, html tags, javascript commands, slashes that might interfere with the website functionality or might reveal some database tables!

Some useful functions

```
<?php
$str = "Is your name O'reilly?";

// Outputs: Is your name O'reilly?
echo stripslashes($str);
?>
```

```
<?php
$str = "A 'quote' is <b>bold</b>";

// Outputs: A 'quote' is &lt;b&gt;bold&lt;/b&gt;
echo htmlentities($str);
```

```
<?php
$text = '<p>Test paragraph.</p><!-- Comment --> <a href="#fragment">Other text</a>';
echo strip_tags($text);
// outputs: Test paragraph. Other text
```

solution

```
function sanitizeString($var)
{
    $var = stripslashes($var);
    $var = htmlentities($var);
    $var = strip_tags($var);
    return $var;
}
```

```
function sanitizeMySQL($var)
{
    $var = mysqli_real_escape_string($var);
    $var = sanitizeString($var);
    return $var;
}
?>
```

Now we can do this:


```
$variable = sanitizeString($_POST['user_input']);
```



3

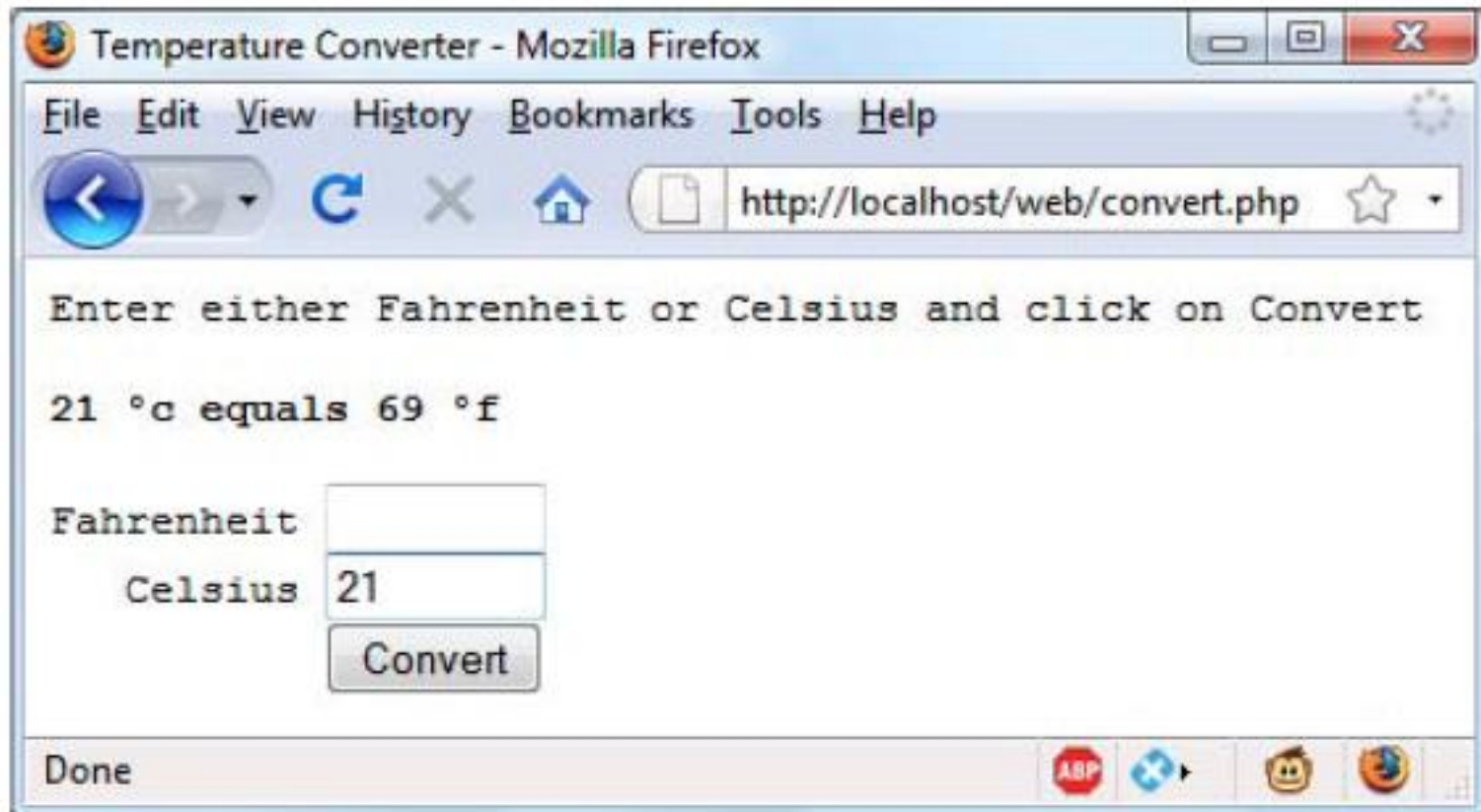
Example

```
<?php // convert.php
$f = $c = "";
if (isset($_POST['f'])) $f = sanitizeString($_POST['f']);
if (isset($_POST['c'])) $c = sanitizeString($_POST['c']);
if ($f != '')
{
    $c = intval((5 / 9) * ($f - 32));
    $out = "$f °F equals $c °C";
}
elseif($c != '')
{
    $f = intval((9 / 5) * $c + 32);
    $out = "$c °C equals $f °F";
}
else $out = "";
```



```
echo <<<_END
<html><head><title>Temperature Converter</title>
</head><body><pre>
Enter either Fahrenheit or Celsius and click on Convert
<b>$out</b>
<form method="post" action="convert.php">
Fahrenheit <input type="text" name="f" size="7" />
Celsius <input type="text" name="c" size="7" />
<input type="submit" value="Convert" />
</form></pre></body></html>
_END;
```

```
function sanitizeString($var)
{
$var = stripslashes($var);
$var = htmlentities($var);
$var = strip_tags($var);
return $var;
}
?>
```



Some tips and tricks (1)

- When you are embedding php within html, it's a common **MISTAKE** to write this

```
<p> the value is <?php $v ?> </p>
```

- This will not print the variable \$v as you did not ask php to print it. it should be

```
<p> the value is <?php echo $v; ?> </p>
```

- A **quick** way to do this also:

```
<p> the value is <?=$v?> </p>
```


Some tips and tricks (2)

- When your php script receives the form, we usually read the form values using `$_GET[]` and `$_POST[]`
- There's a quick way to get those values using their "name" fields in the form
- just do this:
`EXTRACT($_POST); // or`
`EXTRACT($_GET)`
- then you have all the values in variables having the same name as the "name" fields in html.

Example on extract

- Assume the form looks like this

```
<form action='second.php' method='GET'>  
  value 1 <input type='text' name='v1'/>  
  value 2 <input type='text' name='v2'/>  
  <input type='submit' value='go'/>  
</form>
```

value 1 value 2

Example on extract (cont.)

value 1 value 2

- in the page second.php

```
<?php
```

```
extract($_GET);  
echo "Value 1 is $v1 and value 2 is $v2";
```

```
?>
```



directly use the names as variables

GET Trick

- sometimes, we want to send a value or some values to another page without having a form. We can do this using a link and **manually doing** a GET request.

```
<a href='somepage.php?x=2&y=3'> go </a>
```

Now, in somepage.php, we can extract those values using `$_GET['x']` and `$_GET['y']`

Lab session/ example

- Write an index page containing a form that has 3 input fields and a submit button.
- When we click on submit, we are navigated to another page containing a select list whose options are the values entered by the user in page 1.