

PHP form handling

CIS 1152 Adv Web Design

Steve Ruegsegger

Modified with Permission by Peter Chapin

Overview

Goal: Use PHP to process form output from sticky inputs

Objectives:

1. HTTP & PHP
2. Form elements
3. Form submission & vars
4. Form handling UI templates
5. Sticky forms!
6. Form/process 'states'
7. HTML Security

static vs dynamic

- This is a really *critical* lecture!
- What you now know:
 - HTML & CSS (last semester)
 - PHP development environment, PHP blocks and variables
 - PHP loops & logic
 - PHP functions
- All those had "**static**" data.
- We must add *dynamic* data. i.e., data read during execution of the script.

1. HTTP & PHP

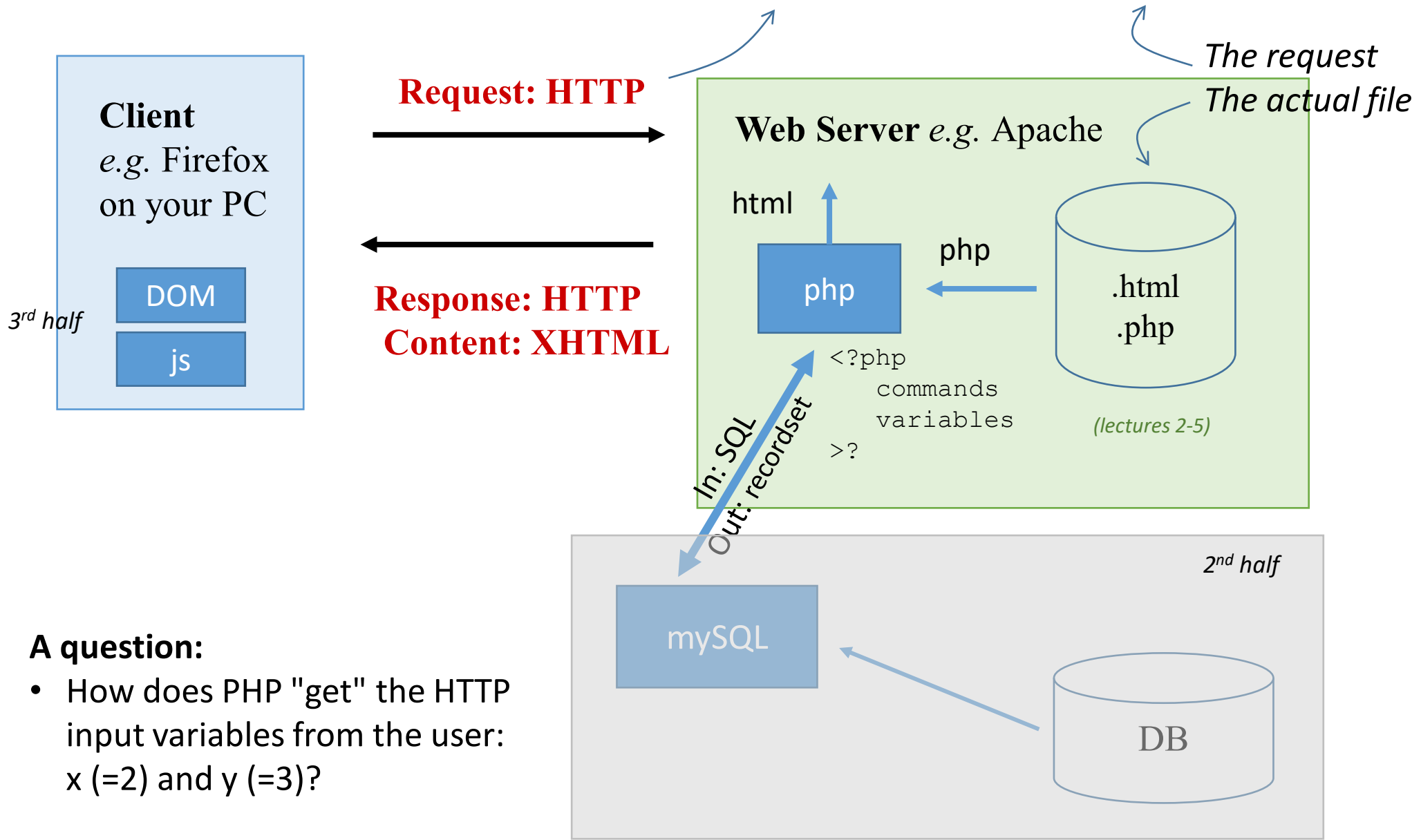
Interaction Overview, Big picture, Variable "handoff"

Superglobal Arrays

`$_GET`: from URL to PHP

Recall: 3-tier web system

http://www.vtc.edu/project.php?x=2&y=3



A question:

- How does PHP "get" the HTTP input variables from the user: x (=2) and y (=3)?

PHP Superglobal arrays

- Superglobals means *available everywhere*
- These are a PHP thing. PHP just creates these **every time!**
- These are the interface between PHP and HTTP
- Notice the "\$_" prefix
 - \$_GET, \$_POST, \$_REQUEST → when using forms
 - \$_SERVER
 - \$_ENV
 - \$_COOKIE
 - \$_SESSION

HTTP to PHP variable hand-off

- `$_GET` is a *superarray* with input vars from HTTP request

localhost/lectures/105_get_superarray.php

\$_GET PHP superarray

This is in `$_GET`:

```
Array
(
    [x] => 3
    [foo] => bar
)
```

```
array(2) {
    ["x"]=>
    string(1) "3"
    ["foo"]=>
    string(3) "bar"
}
```

Remember `print_r()` ???

```
6 <h1>$_GET PHP superarray</h1>
7 This is in $_GET: <br>
8 <pre>
9 <?php print_r($_GET) ?>
10 <hr>
11 <?php var_dump($_GET) ?>
12 </pre>
13
```

106_get_superarray.php

2. HTML form elements

Things in UI forms
and how to make them sticky

HTML forms element review

- I assume you know this... right?
 - `<form>`
 - `<input type="..." name="..."`
 - `<select> </select>`
- There are lots of places for you to review.
Last semester's course?
- Here's the w3schools web site: <http://bit.ly/2BH3yQz>

HTML form elements to know

- `<form action=xxxxxxx method=yyy></form>`
 - "action" is the next URL to "go to" when the user clicks "submit"
 - There are two *methods*. (We'll look at them later.)
 - There are two strategies for *action* management. We'll look at them later.
- `<input type="text" ...>`
- `<input type="radio" ...>`
- `<input type="checkbox" ...>`
- `<input type="submit" ...>`
- `<select><option value=zzzz>...</select>`
- `<textarea>...</textarea>`
- `<button>`

Think key-value pairs

- Each HTML form element has a **key-value pair**
- The **key** is the *name=key*
- The **value** is *value=value*



```
<input type="text"   name='address'   value="$add">
<input type="radio" name='brand'   value='nike'>
<input type="checkbox" name="giftwrap">
<select name='shoesize'><option value='9.5'>
<input type="submit" name='submenu' value='Verify'>
```

value

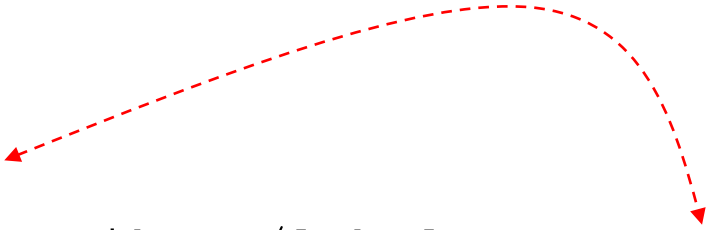
`$_GET['shoesize'] = '9.5';`

key/name

value

Preference: Label for radio and checkbox

- Use a label to be able to click on the "word" (label) next to radio or checkbox to change the checkbox.
- The id= is a DOM (html/javascript) key.



```
<label for="id01">Nike:</label>  
<input type=checkbox name="nike" id="id01">
```

Note:

- "for" and "id" are not the same as "name" and "value"
- Only "name" and "value" are the key/value pairs for \$_GET
- "for" and "id" are for client-side javascript

Default values to form elements

- Know how to give each html web element type a "current" or "default" value in the form.
 - text → value=
 - radio, checkbox → "checked"
 - option → "selected"
- **Later**, we'll use PHP to fill in the proper default or "state" (previous) value.

```

28 <form method="post" action="form2.php">
29
30 customer name: <input type="text" value="Bob"><br>
31 shoe brand:
32     <input type="radio" name="brand" value="adidas"> Adidas &nbsp; &n
33     <input type="radio" name="brand" value="nike" checked> Nike &nbsp;
34     <input type="radio" name="brand" value="puma"> Puma
35 <br>
36 Select from available sizes:
37 <select name=size>
38     <option value="6">6</option>
39     <option value="7.5" selected>7.5</option>
40     <option value="8">8</option>
41     <option value="9">9</option>
42     <option value="9.5">9.5</option>
43 </select>
44 <br><br>
45 Select additional options (check all that apply): <br>
46 <input type="checkbox" name="options1" value="gift"> Gift Tag
47 <input type="checkbox" name="options2" value="lace"> Extra Laces
48 <input type="checkbox" name="options3" value="email" checked> Email
49 <br> <br>
50 <input type=submit>
51 </form>
52 <em>Hit F5 to reset this form</em>
53 </body>
54 </html>

```

*Build this with
PHP arrays*

Steve's shoe shop

customer name:

shoe brand: Adidas Nike Puma

Select from available sizes:

Select additional options (check all that apply):

Gift Tag Extra Laces Email me other discounts

[106_shoe_shop_1_simple.php](#)

Using PHP arrays to build form elements

The same template as building <table>'s from arrays

1. Init the form element (line 22 below)
2. loop over array (line 23)
3. echo the form element for each array (line 24)
4. close the form element (line 26)

my example:

```
19 <?php
20     foreach ($brands as $v) {
21         print("
22 &nbsp; &nbsp; <input type=radio name=brand value='$v' id='id$v'>
23 <label for='id$v'> $v </label>
24 ");
25     }
26 ?>
```

book example:

```
20
21 // Make the months pull-down menu:
22 echo '<select name="month">';
23 foreach ($months as $key =>
24     $value) {
25     echo "<option value=\"\$key\">
26         $value</option>\n";
27 }
```

3. Form submission

Details and Variables

Form submission

- There are 2 ways to send data from a form
 - POST
 - GET
- GET – the values are in the URL.
 - Messy – super long URLs
 - Not great for cut/paste.
 - Values stored in browser history
- POST – form values are sent in "hidden" buffer
 - Variables not available for cut/paste in URL
 - The values are in an environment array
 - Probably preferred method.

Form method

- Probably use **method="post"** most often
- The method= and action= are in <form> tag
- The default "action=" is the same, current script.

```
27
28 <form method="post" action="form2.php">
29
30 customer name: <input type="text" value="Bob"><br>
31 shoe brand:
32     <input type="radio" name="brand" value="adidas"> Adidas &nbsp; &nbsp; &nbsp;
33     <input type="radio" name="brand" value="nike" checked> Nike &nbsp; &nbsp; &nbsp;
```

PHP variable hash array

- Remember that key/value pair concept?
 - name=\$key
 - value=\$value
- Access to the php vars from form associative superglobal array
 - \$_GET
 - \$_POST
 - \$_REQUEST
- Use function print_r() or var_dump() to see all values.
- Run: l06_shoe_shop_1.php

Steve's shoe shop 1

Simple form. Values don't stick in form. PHP throws error if form not filled out correctly.

customer name:
shoe brand: (click label) Adidas Nike Puma

Select from available sizes:

Select additional options (check all that apply):

Gift Tag Extra Laces Email me other discounts

Array

```
(  
    [customer] => xxx  
    [brand] => Adidas  
    [size] => 6  
    [submit] => Go For It  
)
```

Variable exist

- Recall that PHP will **fatal error** if you ask for a 'key' that doesn't exist.
 - *e.g.* you ask for `$_REQUEST['name']`, but it's not filled out → ERROR
- Several functions can deal with this:
 1. **isset(\$var)** – will return TRUE if a value, (including 0, FALSE, "", but not NULL)
 2. **empty(\$var)** – will return TRUE for 0, FALSE, "", and NULL
 3. **array_key_exists('key', \$array)** – T or F if key exists (no error)

```
if (isset($_REQUEST('checkbox1'))) {  
    $gifttag = $_REQUEST('checkbox1');  
}  
else { $gifttag = 'none'; }
```

Notes from prof:

- `empty()` is used for type=text form fields since the element always exists
- All other form types use `isset()`, since the form element might not exist
- I should start using `array_key_exists()` more

Another way to check if key exists in array

- New operator for PHP 7 or later.
- The **null coalescing operator**: ??
- Similar to the ternary operator "? :", but ?? has no conditional.
- ?? has a built-in conditional is "! empty()"
- syntax:

```
$var = $non_null_value ?? null_value
```



- Another way to think of ?? ...
It's like an **"or"** function, where the first non-null wins.

```
$tag = $_POST['gifttag'] ?? FALSE ;
```

Examples...

- These 3 lines do the *same thing*...
- They check if the key "tag" exists in the super array `$_POST` , and avoid PHP Warning
- Lines 58 & 59 uses **ternary operator**
- Line 60 uses the **null coalescing operator** (same as 58)

```
57  
58  $tag = isset($_POST['gifttag']) ? $_POST['gifttag'] : 'none'; # ternary  
59  $tag = isset($_POST['gifttag']) ? "yes" : 'no'; # ternary  
60  $tag = $_POST['gifttag'] ?? 'none'; # null coalescing  
61
```

4. Form handling templates

Forms → User Interface (UI)

Model, View, Controller

You have a lot of choices

User experience

- There are 3 things you *must* do with every form:
 - Why do I say *must*? This is the human *expectation*...
1. **Get** the data from the user in a *nice form*
 2. **Present** it back to the user for *verification* (and allow *editing*)
 3. **Process** the verified data... permanently store it where it can cause another desired action (purchase, service, product, info)
- Ok... How **to do** those 3 things?
 - Well, we have *options*...
 - I'm going to present the options to you. You pick your favorite!

Model-View-Controller

- Here is a programming paradigm.. *A way to think of those 3 things*
- Model-View-Controller, MVC
- Each is a code function
- View is HTML, Model & Controller are PHP

- **Model**

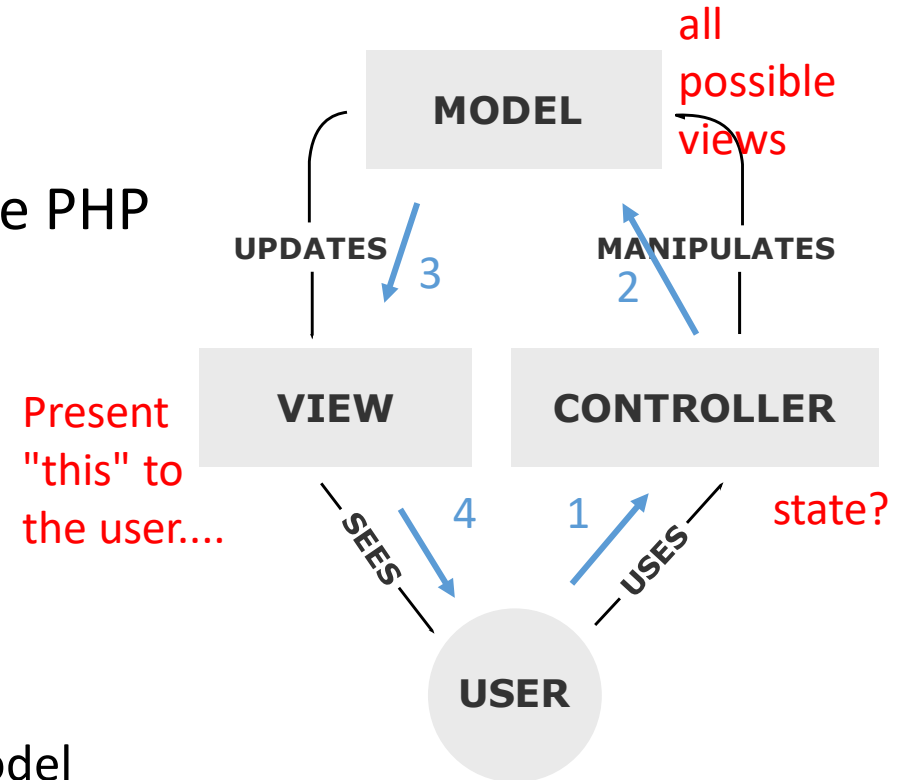
- What the form can "do".
- The collection of all info for all 3 states

- **View**

- The HTML output the user sees.
- Gets *appropriate* "state" variables from Model

- **Controller**

- Knows "where" the user is in the flow of the app
- Knows the app "state"
- Organizes the Model code, runs the *appropriate* part of the Model code.



Presentation Strategies

1. The data input form (UI) and the Results are both on the *same page at the same time*
 - The user sees *both*
 - When the user changes the *upper* form, then the *lower* results change when submitted
2. The data inputs, verification and results are on *different views / pages*
 - The user sees *one at time*
 - User Experience: 1) input data, 2) verify data, 3) save or write or process

Guess my favorite color

I guess:

Submit

Woo Hoo! You are correct. blue it is.

form

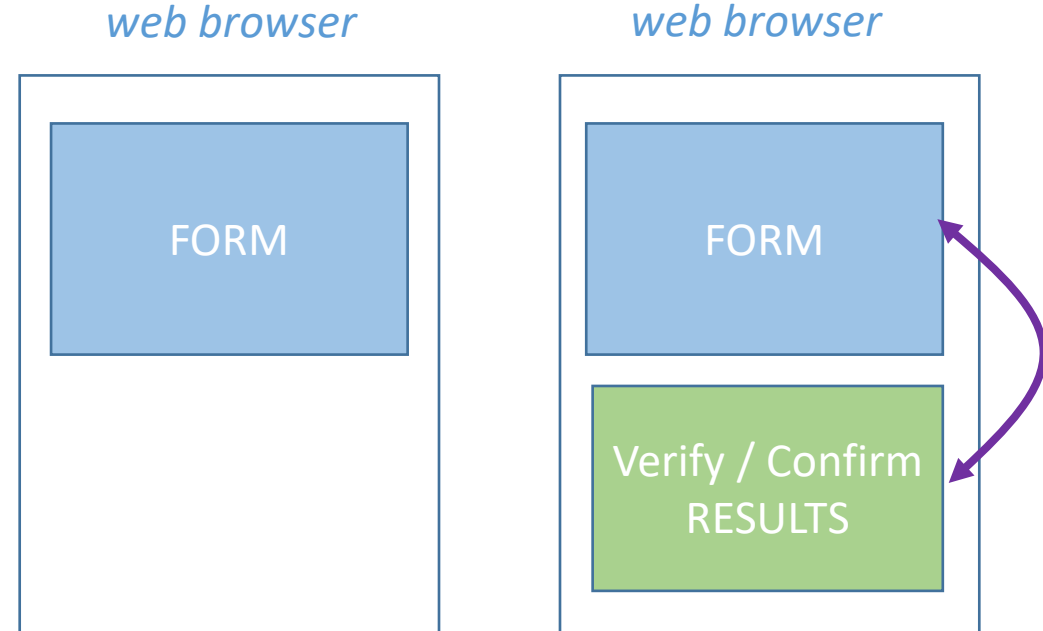
Response

[l06_mvc_guess_color_1.php](#)

Presentation Strategies

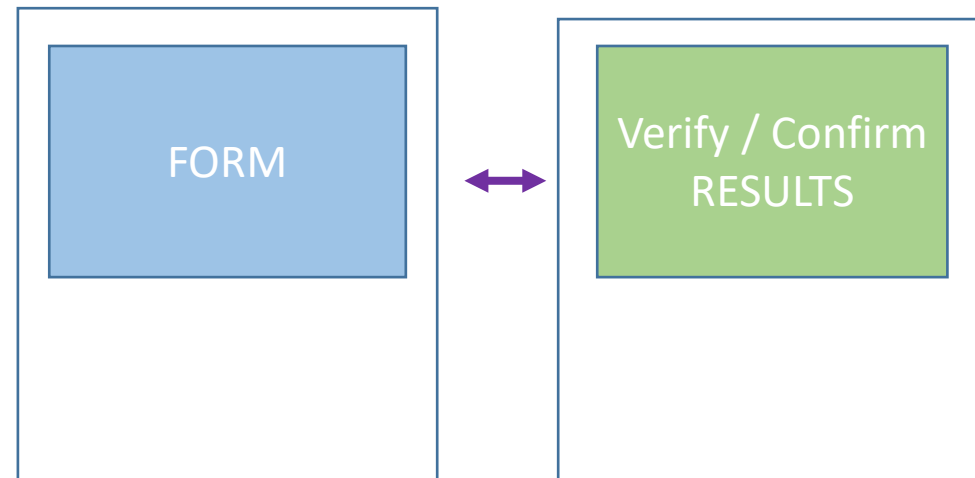
Method 1

- Put *results* under *form*
- UI feels like 1 page
- User changes element in "top" form and updates "bottom" results
- Note: All PHP code in 1 .php script



Method 2

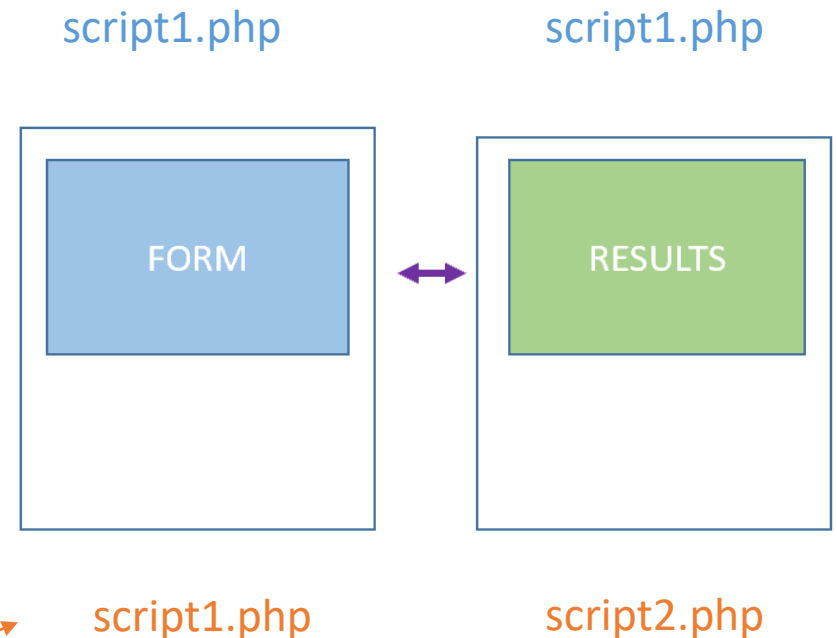
- Results on separate page from input form
- UI feels like multiple pages
- User goes "back" to edit form, and re-submits to new results page
- Note: The PHP either be in 1 script or *multiple* .php scripts.



Same or different .php script files

1. You could have all PHP code, UI form and Results, in **1 script file**.

- The `<form action=...>` points to the very same file.
- An if-then checks to see if this is the first time to the file or after the "submit"
- (+) all variables are in 1 file. The form has 'sticky' values from same vars as UI form.
- (+) any pre-processing can be the same code
- (+) only maintaining 1 file
- (-) the UI form is mostly HTML, so there a lot of PHP `print()`-ing of HTML code.



2. Or you could the UI form code in a **different file** from the Results code, and thus there are **2 script files**.

The first file's `<form action=...>` points to the 2nd PHP form processing file

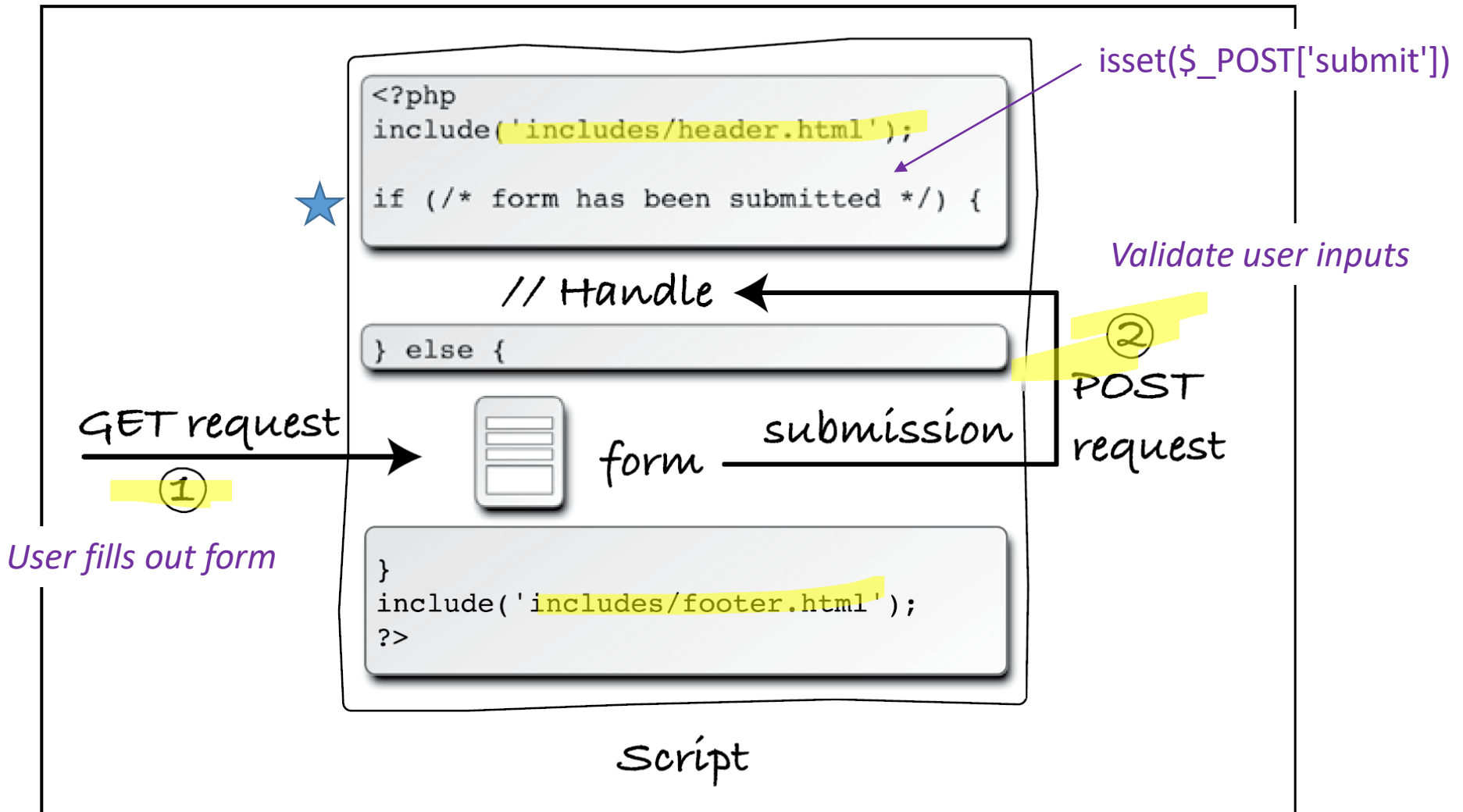
(+) Most helpful if the form has "no" PHP and is all HTML

(-) I now have 2 files to maintain and edit. The 2 files *have* to know about each other.

(-) two headers/footers – they could deviate

"Same script" Template from Ullman book

pg 85




A The interactions between the user and this PHP script on the server involves the user making two requests of this script.

"Same script" template from prof

- Line 76 uses `$_SERVER['REQUEST_METHOD']` to determine if the user has submitted the form or not.
- There are other methods than this one...

```
74
75 # Has the user submitted or not
76 if ($_SERVER['REQUEST_METHOD'] == 'POST'){
77     # the user has just submitted
78     display_UI_vars();
79     math_table();
80 }
81 else {
82     # NO submission
83     show_UIform();
84 }
85
```

`isset($_POST['submit'])`
`array_key_exists($_POST,'submit')`
`$state = $_POST['submit'] ?? '';`



l06_mvc_guess_color_[12ab].php

"single HTML view"

Guess my favorite color

I guess:

Huh? red? Incorrect. Try again.

- 1 script
- Use if-then-else to know what state & appropriate answer/presentation

"separate HTML views"

Guess my favorite color

What is my favorite color?

I guess:

Your answer to my favorite color

Huh? red? Incorrect. Try again.

Play again:

- 1 script or 2 scripts
- 1 script: use if-then-else to know what state
- 2 scripts: separate files for each state

MVC example

Model and Controller

- All PHP
- Sends **\$guess** and **\$message** to View (called the context)
- Knows if the user has guessed or not
- Silent (no output)

View

- All HTML
- gets **\$guess** and **\$message** (the context) from Model
- Puts them into proper HTML spot
- What the user sees

```
105_mvc_guess_color.php — C:\Users\steve\Box Sync\VTC\Adv We...
File Edit View Selection Find Packages Help

105_mvc_guess_color.php
1 <?php
2 $guess = $_POST['guess'] ?? "";
3 $favoriate = 'blue';
4
5 $message = "";
6 if (empty($guess)) {
7     $message = "What is my favoriate color?"; }
8 else if ($guess == $favoriate){
9     $message = "Woo Hoo! You are correct. ";}
10 else if ($guess == "purple"){
11     $message = "I'm not a child any more. ";}
12     else if ($guess == "yellow"){
13         $message = "Nope. Too bright. ";}
14     else { $message = "Incorrect. Try again. "; }
15 ?>
16
17 <!doctype html>
18 <html>
19 <head></head>
20 <body>
21 <h1> Guess my favoriate color</h1>
22 <p> <?= $message ?> </p>
23
24 <form method=POST action="105_mvc_guess_color.php">
25 I guess:
26 <input type=input name=gues value=<?= $guess ?> ><br><br>
27 <input type=submit value='Submit' name=submit>
28 </form>
29 </body>
30 </html>
31
```

validate entry values & highlight any errors

- It's good to check for valid responses and give that feedback to the user
- Missing values in UI text form

```
if (empty($_REQUEST('name'))) {  
    echo "<font color='red'>Error. Name is required</font><br>";  
    error_count = error_count + 1;  
}
```

- Use regex to ensure proper strings from user

```
if (!preg_match('\d{3}-\d{3}-\d{4}', $_REQUEST['phonenum'])) {  
    echo "<font color='red'>Error. Not a proper phone number</font><br>";  
    error_count = error_count + 1;  
}
```


5. Sticky forms via PHP

Have a *previous* value in every type of HTML input element from PHP script

"sticky" *a.k.a.* Persistent Form Data

Persisting Form Data

- When we submit forms and there is an error, we just expect that the data will remain in the form when the page is redisplayed.
- The application needs to make sure to put the previous values back into the form.

Our expectation!

```
www.wa4e.com/code/forms/form3.php

Guessing game...

Input Guess 

Submit

$_POST:
Array
(
)

$_GET:
Array
```

```
www.wa4e.com/code/forms/form4.php

Guessing game...

Input Guess 20

Submit

$_POST:
Array
(
    [guess] => 20
)

$_GET:
Array
```

"This input value '20' does not automatically show up in this form. Us developers have to make this happen!"

Sticky text box

- Get previous value, if exists into PHP \$var
 - Use ternary or null coalesce (PHP7+) function

```
17  # If the user has submitted, then get the user value
18  # or a default
19
20  # $customer = isset($_POST['customer']) ? $_POST['customer'] : 'ima default'; # ternary
21  $customer = $_POST['customer'] ?? 'ima default'; # null coalesce
```

- In the HTML UI form, use that PHP \$var as value=""

```
42  <form method="post" action="l05_shoe_shop_2_sticky.php">
43
44  customer name: <input type="text" name="customer" value='<?=$customer ?>'><br>
45  shoe brand: (<em>click label</em>)
```

These are the same

Note: cool PHP short cut!

```
<?=$oldguess ?>
<?php echo($oldguess); ?>
```



sticky **checkbox** → array of "checked"

- I use an \$array to store the options of a set of checkboxes
- Remember, *multiple* array values can be 'checked'.

```
41 # checkbox
42 $c2[0]=(isset($_POST['options1'])) ? 'checked' : '';
43 $c2[1]=(isset($_POST['options2'])) ? 'checked' : '';
44 $c2[2]=(isset($_POST['options3'])) ? 'checked' : '';
45
```

PHP-land

HTML-land

```
65 Select additional options (check all that apply): <br>
66 <input type="checkbox" name="options1" value="gift" <?php echo "$c2[0]";?>> Gift Tag
67 <input type="checkbox" name="options2" value="lace" <?php echo "$c2[1]";?>> Extra Laces
68 <input type="checkbox" name="options3" value="email" <?php echo "$c2[2]";?>> Email me other di
69 <br> <br>
```

User's browser

Select additional options (check all that apply):
 Gift Tag Extra Laces Email me other discounts

Sticky `select` pulldown method → "select"

- I also use an `$array` to store the options of a set of sections
- But only one will be 'selected'.

```

34 $size = isset($_POST['size']) ? $_POST['size'] : '';
35 $s1[0]=($size=='6') ? 'selected' : '';
> 36 $s1[1]=($size=='7.5') ? 'selected' : '';
37 $s1[2]=($size=='8') ? 'selected' : '';
38 $s1[3]=($size=='9') ? 'selected' : '';
39 $s1[4]=($size=='9.5') ? 'selected' : '';

```

ternary operators: \$size in Boolean T/F test

```

57 <br>
58 Select from available sizes:
59 <select name=size>
60   <option value="6"   <?=$s1[0] ?> >6 </option>
61   <option value="7.5" <?=$s1[1] ?> >7.5</option>
62   <option value="8"   <?=$s1[2] ?> >8 </option>
63   <option value="9"   <?=$s1[3] ?> >9 </option>
64   <option value="9.5" <?=$s1[4] ?> >9.5</option>
65 </select>

```

Select from available sizes: 8 ▼

Select additional options (check all that apply)

Gift Tag Extra Laces Email

Submit

Array

```

(
    [customer] => joe blow
    [size] => 8
    [submit] => Submit
)

```

Sticky **radio** buttons → array of "checked"

- Radios are like checkboxes above, but only 1 will be 'checked'.
- Notice how I built all the N radio buttons with a foreach loop. Cool!

```
23     $brand = $_POST['brand'] ?? ''; #
24     $c1[0]=($brand=='Adidas') ? 'checked' : '';
25     $c1[1]=($brand=='Nike') ? 'checked' : '';
26     $c1[2]=($brand=='Puma') ? 'checked' : '';
```

```
45     shoe brand: (<em>click label</em>)
46     <?php
47         $n=0;
48         foreach ($brands as $v) {
49             print("
50             &nbsp; &nbsp; <input type=radio name=brand value='$v' id='id$v' $c1[$n]>
51             <label for='id$v'> $v </label>
52             ");
53             $n++;
54         }
55     ?>
```

customer name:

shoe brand: Adidas Nike Puma

Select from available sizes: ▼

6. Form 'states'

Knowing where you are in the process

What is a 'state'

- A 'state' is the current process step in a flow
- Tic-tac-toe
 - The 'state' is the current board values and who's turn it is.
- Math Table
 - 3 variables: row, col, function
- Restaurant
 - Table number, num orders, all the plate & drink orders,
- 'state' – all the information needed 'up to now' to get me here and needed to finish this thing

Use a variable for the state

- Here, the \$state is the submit button values

Steve's shoe shop

customer name:

shoe brand: Adidas Nike Puma

Select from available sizes: ▼

Select additional options (check all that apply):

Gift Tag Extra Laces Email me other discounts

DEBUG

All variables from \$_POST:

POST: key=submit , value=Return

```
27
28     # 3 states:
29     #     null = fill out form
30     #     Verify = verify form
31     #     Submit = proces form
32     $state = $_POST['submit'] ?? '';
33
34     $customer = $_POST['customer'] ?? "";
35     $brand = $_POST['brand'] ?? "";
36     $size = $_POST['size'] ?? "";
37
38     if ($state == 'Verify') {
39
```

We often need to 'save' the state

- How does our Web app save the state?
- Three primary methods
 1. HIDDEN
 2. Cookies
 3. Databases
- We'll cover *all* of these in this course!
- This week: **HIDDENs**

hidens

- An HTML form element
- Doesn't show anything on the screen
- But the **key/value** shows up in the `$_POST` value just like anything else.
- I have had a few to dozens of these in an app
- Remember, use `^U` to view HTML to see them

```
<input type=hidden name=KEY value='VALUE'>
```

```
<input type=hidden name=state value='verify'>
```

```
<input type=hidden name=state value=1>
```

One way to "automate" hidden values

- For the verify state, put all the values in hidden values so they can be passed:
 - *Forward* to the processing state
 - Or *back* to the re-edit state

```
19 function printHiddens() {
20     foreach ( $_POST as $k => $v ) {
21         if ( $k == 'submit' ) { continue; } # not this one
22         print "<input type=hidden name=$k value='$v'>\n";
23     }
24     print "\n\n";
25 }
26
27
```

foreach() loop to put ALL variables into a HIDDEN

```
5
6 <h1> Steve's shoe shop </h1>
7
8 <!-- all states see the same form -->
9 <form method='post' action='l5_shoe_shop_states.php'>
10
11
12 <input type=hidden name=customer value='Tim ' >
13 <input type=hidden name=brand value='puma'>
14 <input type=hidden name=size value='8'>
15
16
17 <br><br>This order is for customer: <b>Tim </b><br>
18     who wants a pair of <b>puma</b> shoes
19     of size <b>8</b>.<br>
```

7. HTML Security

HTML Injection protection via htmlentities()

What is HTML Injection?

- A bad guy knows your HTML
- He can see your code in his client (^U) and he knows you have a sticky text form...

```
43 <form method="post" action="l05_shoe_shop_2_sticky.php">
44
45 customer name: <input type="text" name="customer" value="<?=$customer?>"><br>
46 shoe brand: (<em>click label</em>)
```

- So, he enters in a 'name' of:

"> bad-guy code... "<

Steve's shoe shop 2

Values now stick in form. Form checked for being valid.

customer name: "> Bad guy code here <

shoe brand: (*click label*) Adidas Nike Puma

Select from available sizes:

What does HTML produce?

- *His code ends your code block*, then he can start to write whatever free HTML or PHP code he wants!

Values now stick in form. Form checked for being valid.

customer name: **Bad guy code here** <"">

shoe brand: (*click label*) Adidas Nike Puma

Select from available sizes: 8 ▾

Select additional options (check all that apply):

Gift Tag Extra Laces Email me other discounts

Array
(
 [customer] => "> Bad guy code here <"
 [size] => 8
 [submit] => Submit
)

His code could steal your personal information, credit cards, passwords, etc.

```
11  
12  
13 <form method="post" action="l05_shoe_shop_2_sticky.php">  
14  
15 customer name: <input type="text" name="customer" value=""> <b>Bad guy code here </b> <""><br>  
16 shoe brand: (<em>click label</em>)  
17
```

Q: How to protect?

- A: Always convert user text to 'escaped' characters
- One function: **htmlentities(string)**
- That function converts non-alphabet (non-A to Z) special characters to their HTML code.
- So the bad guy doesn't end your quote " or your HTML tag >

php code

```
20  # $customer = isset($_POST['customer']) ? htmlentities($_POST['customer']) : 'ima default';
21  $customer = htmlentities($_POST['customer']) ?? 'ima default'; # null coalesce
```



HTML output

```
12
13 <form method="post" action="l05_shoe_shop_2_sticky.php">
14
15 customer name: <input type="text" name="customer" value='&quot;&gt; &lt;b&gt;Bad guy code here
&lt;/b&gt; &lt;&quot;';><br>
16 shoe brand: (<em>click label</em>)
```

* Note: htmlentities() escapes *double* quotes but not *single* quotes.

Summary

1. Understand POST and GET process and global hash
2. Know how to make values 'sticky' in a form
3. Understand UI templates for handling forms
4. Understand process 'states'
5. Be wary of HTML Injection as a Security precaution