

# WEB DEVELOPMENT WITH PHP

## COURSE #06



# LET'S RECAP

## COURSE 5 WAS ABOUT:

- ★ Superglobal variables
- ★ URL parameters and `$_GET`
- ★ Structuring your code
- ★ Functions

# TOPICS

COURSE #06

1. Superglobal variables, HTML forms & POST
2. Databases - MySQL [1/3]
3. Homework

# 1. SUPERGLOBALS VARIABLES

# SUPERGLOBAL VARIABLES

## WHAT ARE SUPERGLOBAL VARIABLES?

**Variables which are created automatically by PHP and hold various information regarding the current script execution.**

They are available anywhere in your code and you usually use them to grab the user input or to communicate between the server where the PHP code is stored and the visitor's browser.

*"From a programmer's point of view, the user is a peripheral that types when you issue a read request."*

*— P. Williams*

# HTML forms

HTML forms are used to collect user input.

# HTML FORMS

## ATTRIBUTES

- METHOD
- ACTION
- SUBMIT



# HTML Forms - METHOD

- The **Method** attribute is used to tell the browser how the form information should be sent.
- The two most popular methods you can use are GET and POST.

## Examples:

```
<form name="form1" method="get" action="">  
    <INPUT TYPE = "TEXT" VALUE ="username">  
    <INPUT TYPE = "Submit" Name = "Submit1" VALUE = "Login">  
</form>
```

```
<form name="form1" method="post" action="">  
    <INPUT TYPE = "TEXT" VALUE ="username">  
    <INPUT TYPE = "Submit" Name = "Submit1" VALUE = "Login">  
</form>
```

Try each one of them! What is the difference?

# HTML Forms - ACTION

- ➔ The Action attribute is crucial. It means, "Where do you want the form sent?".
- ➔ In PHP, a popular technique is to send the script to the same page that the form is on – send it to itself, in other words.

## Examples:

```
<form name="form1" method="get" action="course6.php">  
    <INPUT TYPE = "TEXT" VALUE ="username">  
    <INPUT TYPE = "Submit" Name = "Submit1" VALUE = "Login">  
</form>
```

```
<form name="form1" method="post" action="other-page.php">  
    <INPUT TYPE = "TEXT" VALUE ="username">  
    <INPUT TYPE = "Submit" Name = "Submit1" VALUE = "Login">  
</form>
```

Try each one of them! Where did it land?

# HTML Forms - SUBMIT

- ➔ The HTML Submit button is used to submit form data to the script mentioned in the ACTION attribute.

## Examples:

```
<form name="form1" method="get" action="">  
    <INPUT TYPE = "TEXT" VALUE ="username">  
  
</form>
```

```
<form name="form1" method="get" action="">  
    <INPUT TYPE = "TEXT" VALUE ="username">  
    <INPUT TYPE = "Submit" Name = "Submit1" VALUE = "Enter!">  
  
</form>
```

Try each one of them! What happened in the first case, but in the second?

# HTML Forms - RECAP

➔ In order to get a form working one would need to define:

- A METHOD
- AN ACTION
- A SUBMIT handler

➔ Let's build our first form!

# HTML Form - First Form

- Create a new file named ***form.php***
- Create the following page/form:

```
<html>
  <head>
    <title>A BASIC HTML FORM</title>
  </head>
  <body>
    <form name="form1" method="post" action="">
      <input type="text" name="myName" value="MyName">
      <input type="submit" name="submit1" value="Send">
    </form>
  </body>
</html>
```

- Tasks to accomplish:
  - ◆ Add a textarea named "myTextArea"
  - ◆ Add a select list named "mySelect" having 3 options:
    - Yes
    - No
    - Maybe
  - ◆ Find out how to add labels to inputs

# Let's review superglobals...

→ `$_GET`

An associative array of variables passed to the current script via the HTTP GET method.

→ `$_POST`

An associative array of variables passed to the current script via the HTTP POST method.

Marie



# Keep reviewing superglobals...

→ **\$\_GET** may be used for sending non-sensitive data.

Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL).

GET also has limits on the amount of information to send. The limitation is about 2000 characters.

However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

→ **\$\_POST** has to be used when dealing with sensitive data

Information sent from a form with the POST method is invisible to others (all names/values are embedded within the body of the HTTP request) and has no limits on the amount of information to send.

Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.

# Let's practice a bit

Open file *form.php* created earlier - in any browser

Notice the form. Let's complete and submit it!

What happend ?

Why ?



So what next ?



# Using Form Submissions in PHP

## Remember the definition of \$\_POST and \$\_GET ?

In order to get the values submitted through our form we will take advantage of these globals.

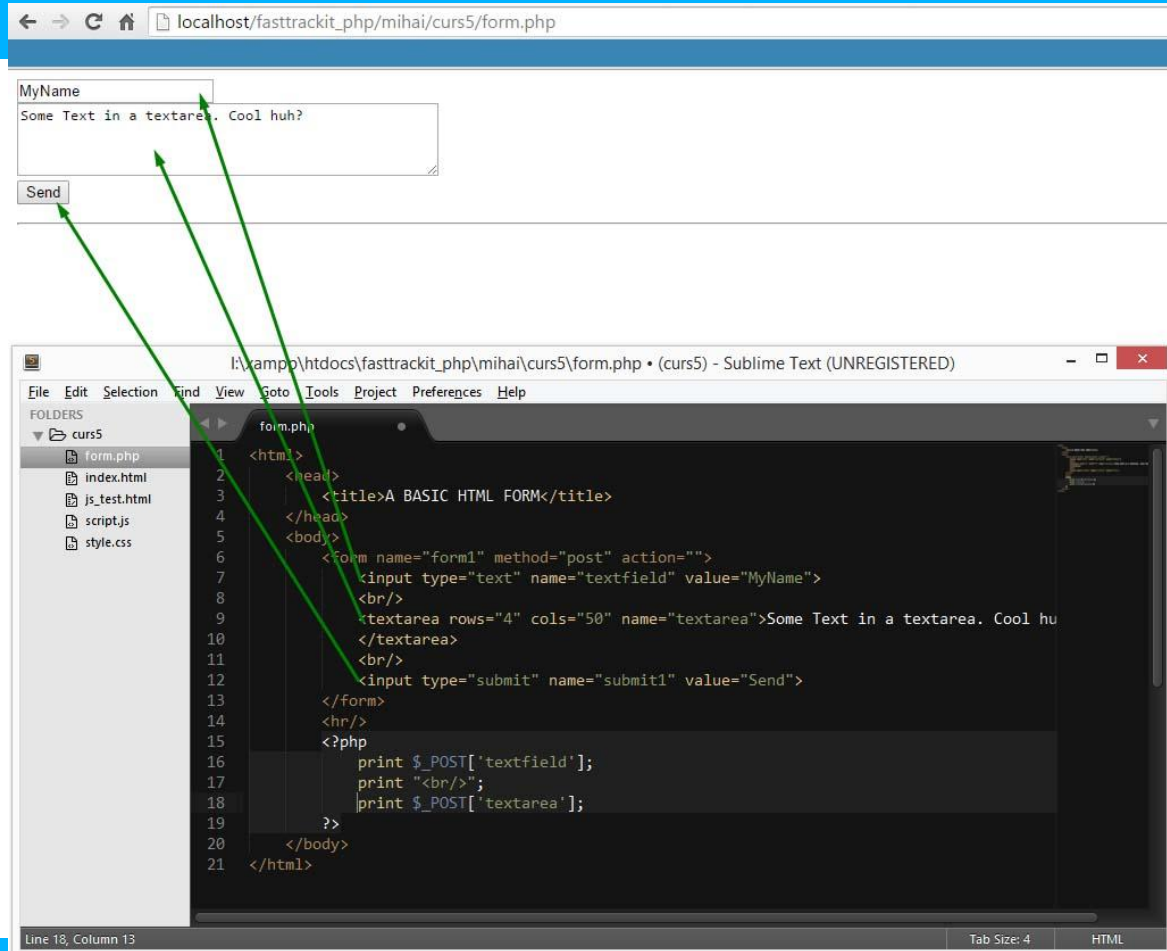
Our values will be available in:

`$_POST['myName']`

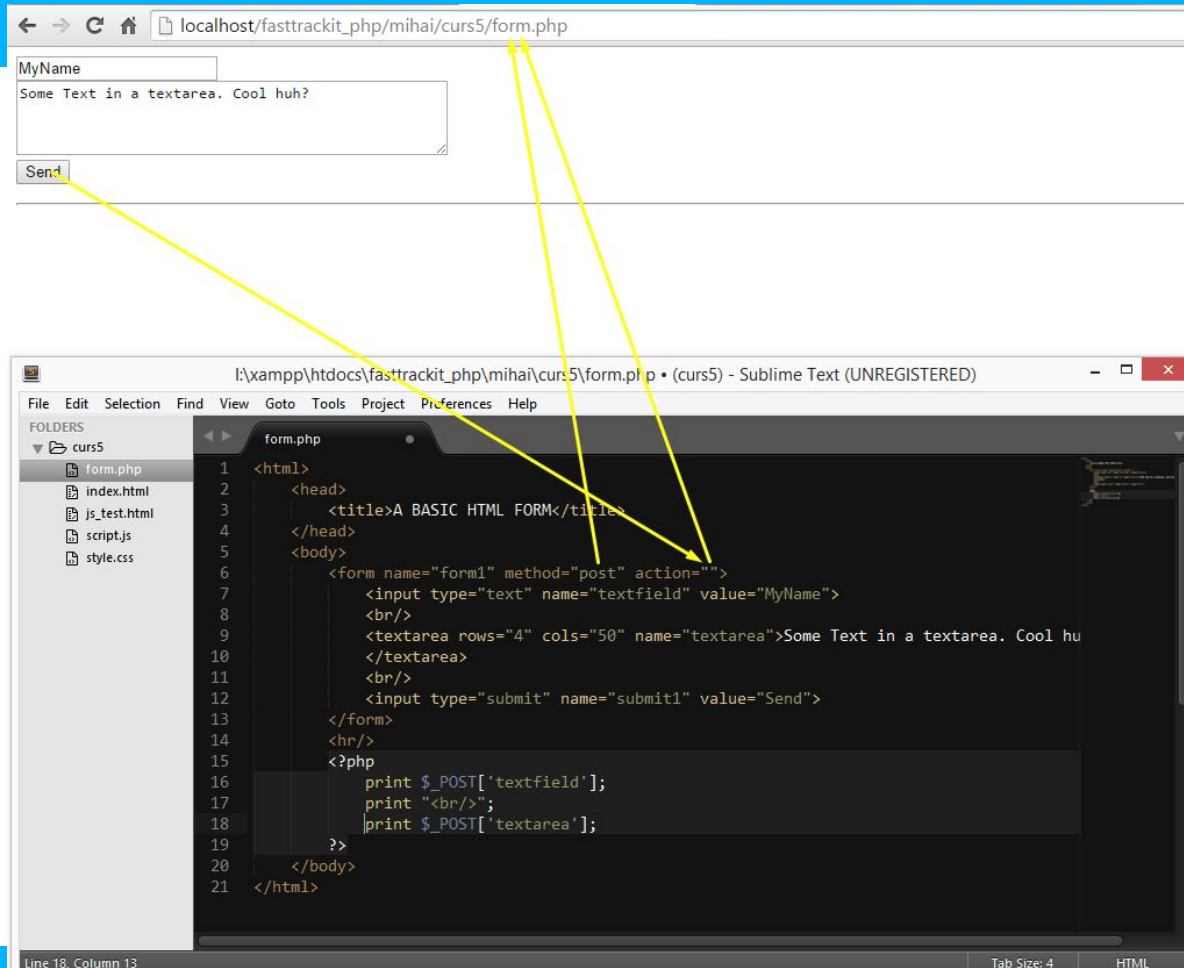
`$_POST['myTextArea']`

`$_POST['mySelect']`

# A picture is worth a thousand of words...



# A picture is worth a thousand of words...



# Yeey! IT WORKS !!!

The image shows a web browser window at the top and a code editor window at the bottom. The browser window displays the output of a PHP script: a form with a text input containing "MyName", a text area containing "Some Text in a textarea. Cool huh?", and a "Send" button. The code editor window shows the source code of the script, which uses the `<form>` tag to create the form and `<input>` tags for the text field and submit button. The PHP code at the bottom of the form tag prints the values of the text field and text area. Green arrows point from the browser's rendered form elements to their corresponding HTML tags in the code editor.

Browser window: localhost/fasttrackit\_php/mihai/curs5/form.php

Form output:

MyName  
Some Text in a textarea. Cool huh?  
Send

Code editor: I:\xampp\htdocs\fasttrackit\_php\mihai\curs5\form.php • (curs5) - Sublime Text (UNREGISTERED)

```
1 <html>
2 <head>
3 <title>A BASIC HTML FORM</title>
4 </head>
5 <body>
6 <form name="form1" method="post" action="">
7 <input type="text" name="textfield" value="MyName">
8 <br/>
9 <textarea rows="4" cols="50" name="textarea">Some Text in a textarea. Cool huh?
10 </textarea>
11 <br/>
12 <input type="submit" name="submit1" value="Send">
13 </form>
14 <hr/>
15 <?php
16 print $_POST['textfield'];
17 print "<br/>";
18 print $_POST['textarea'];
19 ?>
20 </body>
21 </html>
```

# YOUR TURN! :D

Use a PHP output command to display on the screen the following after form submission:

```
Hello my name is [YourName] and i just submitted:  
[YourText] with option [YourSelect]
```

# Different destinations for your Form Submissions

Maybe we would like to send the user input to another page, not the current one, right?

Let's say we need to output it on page ***output.php***

Create it and let's see how to send data to it.

What attribute of the form needs to be changed?  
Why?


Let's try the previous example, but with sending data to ***output.php***

Display on page **output.php** the following after form submitting page **form.php**:

Hello my name is  
[YourName] and i just  
submitted:  
[YourText]  
with option [YourSelect]

LET'S EXERCISE A BIT

# Creating a login form



The image shows a login form with a light gray header bar containing the text "Login to Web App". Below the header, there are two input fields: the first is labeled "Username or Email" and the second is labeled "Password". Below the "Password" field, there is a checkbox labeled "Remember me on this computer". To the right of the checkbox is a blue button with the text "Login".

After login, it should go on home.php and display :

**"Hello [username], your password [password] was correct."**

Also it has to display one of the following, based on user's choice:

**"Yeeey you want us to remember you"**

**"Oh no... you chose that we'll forget you :( "**



# Creating a customer survey form

The form will be located in **survey.php**

After “Send Survey”  
user should view all his answers in  
**answer.php** and have a button to  
“Take the survey again”

Name (optional)

Email (optional)

I BOUGHT...\*

☐ Breakfast Food  
☐ Baked Goods  
☐ Cake/Cupcakes  
☐ Beverage

It TASTED...\*

☐ Excellent  
☐ Good  
☐ Neutral  
☐ Bad  
☐ Horrible

The STYLE was...\*

☐ Excellent  
☐ Good  
☐ Neutral  
☐ Bad  
☐ Horrible

How satisfied are you with...\*

	Very Satisfied	Satisfied	Neutral	Unsatisfied	Very Unsatisfied
The purchase you made.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The service you received.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Our company overall.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

How likely are you to...\*

	Very Satisfied	Satisfied	Neutral	Unsatisfied	Very Unsatisfied
Buy from us again.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Recommend our products to others.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Recommend our shop to others.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

SEND SURVEY

And now...



## 4. DATABASES

# What is a database?

***"A way to store information in a structured way"***

Before computers, we had databases on paper.

As a computer solution, it appeared around 1960.

# Types of databases

## → Flat file

- you edit content manually, not by instructions

## → Relational

- each record has an unique identifier, which is used to create links to other records

## → Non-relational (NoSQL)

- no connection between records
- each record can have a different set of fields

# MySQL

→ **Second most popular database in the world, most popular choice for web development**

→ **Integral part of the LAMP infrastructure:**

**L**inux

**A**pache

**M**ySQL

**P**HP

# History of MySQL

- **1994 - created by Ulf Michael Widenius (a.k.a. Monty) in Sweden**
- **1998 - Windows version**
- **2008 - acquired by Sun (\$1 billion)**
- **2010 - acquired by Oracle**

# Storing information in a database

**We have a list of students, each student has a first name and a last name.**

**How would you do it in an Excel file?**

First name	Last name
John	Doe
Jane	Smith
Alice	Popescu



# Storing information in a database

**We also have a list of courses. Each course has a name and a trainer.**

Course name	Trainer
Introduction in QA	Mike Testovici
Web development with PHP	Sam Webby

# Storing information in a database

**How do we know what course is each student attending?**

**Let's add a new column in the Students sheet, called Course.**

First name	Last name	Course
..	..	

**What should we put in the Course column?**

# Storing information in a database

**Working with words is for humans... computers like numbers better:**

id_student	First name	Last name	Curs
1	John	Doe	2
2	Jane	Smith	2
3	Alice	Popescu	1

id_curs	Course name	Trainer
1	Introduction in QA	Mike Testovici
2	Web development with PHP	Sam Webby

# How to access a mysql database ?

## **phpMyAdmin**

phpMyAdmin is a web-based tool which allows us to work with MySQL databases.

It's written in ... you're right, PHP ... and it's open source.

Let's check if you already have it:

<http://localhost/phpmyadmin/>

Let's phpMyAdmin !

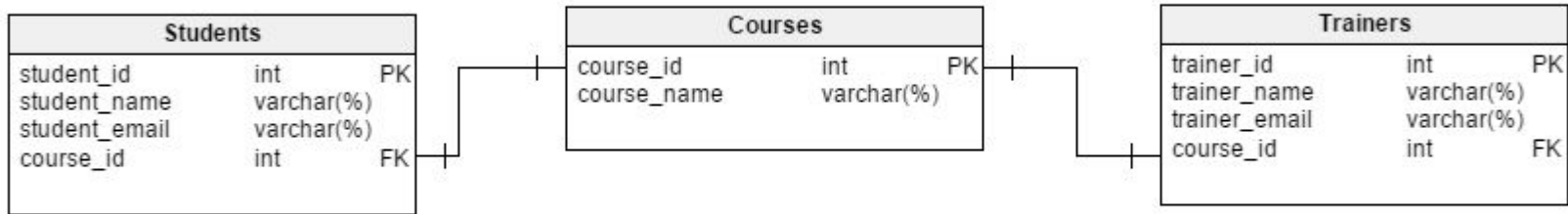


# Let's phpMyAdmin !

1. Create a database called fasttrackit
2. Create a table named students with the following columns
  - a. **student\_id** (numeric, not null, primary\_key)
  - b. **student\_name** (var\_char)
  - c. **student\_email** (var\_char)
  - d. **course\_id** (numeric, foreign\_key)
3. Create a table named courses
  - a. **course\_id** (numeric, not null, primary\_key)
  - b. **course\_name** (var\_char)
4. Create a table named trainers
  - a. **trainer\_id** (numeric, not null, primary\_key)
  - b. **trainer\_name** (var\_char)
  - c. **trainer\_email** (var\_char)
  - d. **course\_id** (numeric, foreign\_key)

# Let's phpMyAdmin !

Reading a database diagram ...



What is a PK ???

What is a FK ???

What is a **one-to-one** relation ?

What is a **one-to-many** relation ?

What is a **many-to-one** relation ?

What is a **many-to-many** relation ?

I don't know,

**Google it.**

Next time on fasttrackit course :)

Coming Up  
Next . . .

SELECT  
INSERT  
UPDATE  
DELETE

MANIPULATING  
DATABASE ENTRIES  
WITH SQL



# GROUP PROJECT

# What do we need



1. Group in teams of 2-3 persons
2. Assign a team responsible
  - a. will communicate the team needs and updates
  - b. will ensure team cohesion
  - c. can be a different person each week so you can get your turn
3. Create trello accounts: <https://trello.com/>
4. Add a picture of you to the trello account
5. Send a mail with your trello username at the group email address
6. We'll add you to the trello board and kickstart the group project!

