

Backend Development

HTTP Trip Story

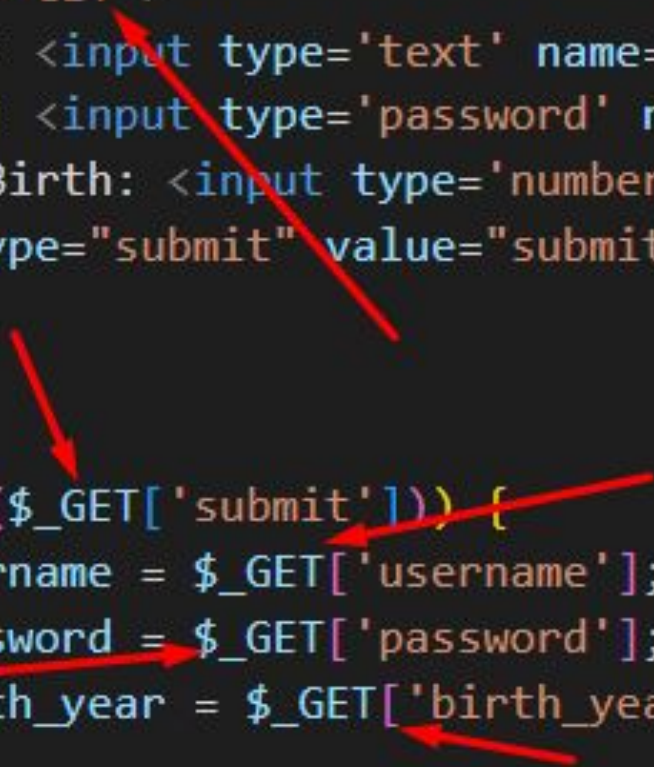


Learning Objectives

- HTTP Requests (How Internet works)
- PHP Superglobals
- GET & POST (usage & differences)
- Form Validation
- Sessions

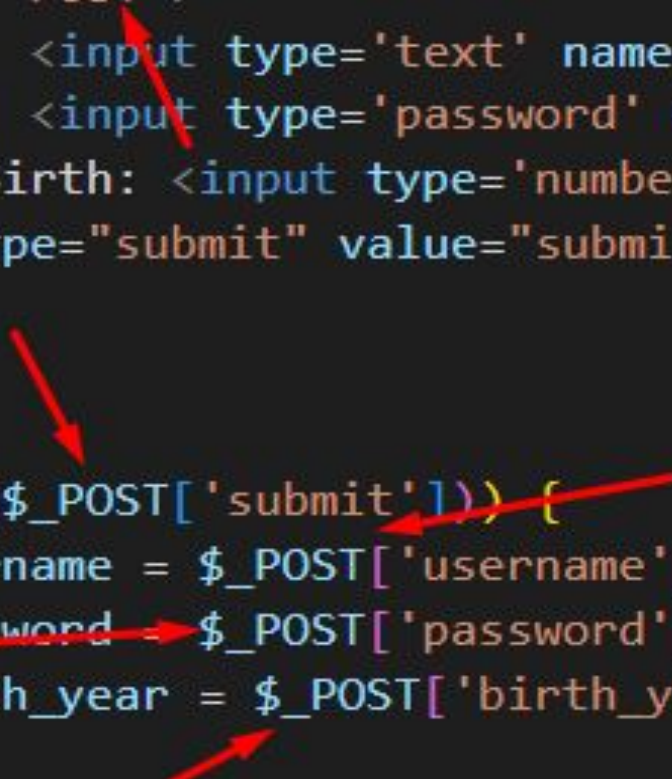
```
<form method="GET">
  Username: <input type='text' name='username' ><br>
  Password: <input type='password' name='password' ><br>
  Year of Birth: <input type='number' name='birth_year'><br>
  <input type="submit" value="submit" name='submit'>
</form>

<?php
  if(isset($_GET['submit'])) {
    $username = $_GET['username'];
    $password = $_GET['password'];
    $birth_year = $_GET['birth_year'];
  }
```



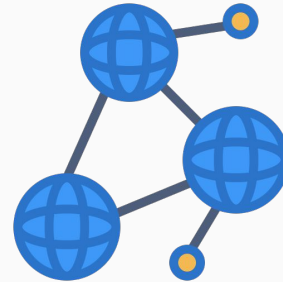
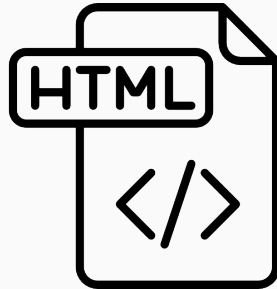
```
<form method="POST">
  Username: <input type='text' name='username' ><br>
  Password: <input type='password' name='password' ><br>
  Year of Birth: <input type='number' name='birth_year'><br>
  <input type="submit" value="submit" name='submit'>
</form>

<?php
  if(isset($_POST['submit'])) {
    $username = $_POST['username'];
    $password = $_POST['password'];
    $birth_year = $_POST['birth_year'];
  }
```



How Internet works?

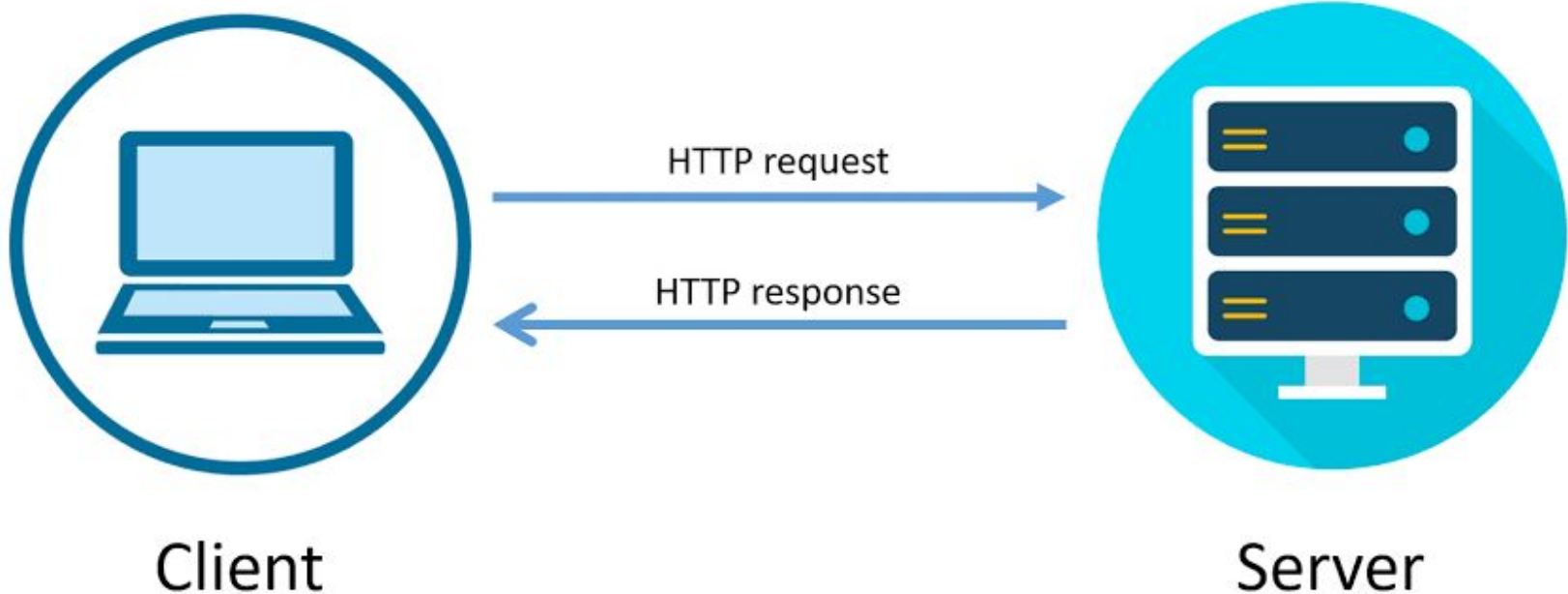
- What is HyperText Markup Language (HTML) ?
- What's the connection between Microsoft Word Office, HTML, Networks and Facebook login process ?









How Internet works?

- LinkedIn example
- Facebook example

How Internet works?



HTTP Request Methods

 GET	 POST	 PUT	 DELETE	 PATCH	 HEAD
retrieve data from server	add data to an existing file or resource	update(replace) an existing file or resource in server	delete data from server	update a resource partially (modify)	retrieve the resource's headers

- **CONNECT** is used to open a two-way socket connection to the remote server;
- **OPTIONS** is used to describe the communication options for specified resource;
- **TRACE** is designed for diagnostic purposes during the development.
- **HEAD** retrieves the resource's headers, without the resource itself.



Form Data, JSON Strings, Query Parameters, View States, etc

GET vs POST

- GET requests can be **cached**
- GET requests remain in the **browser history**
- GET requests can be **bookmarked**
- GET requests should **never** be used when dealing with **sensitive data**
- GET requests have **length restrictions** (2 KB)
- GET requests are only **used to request data** (not modify)

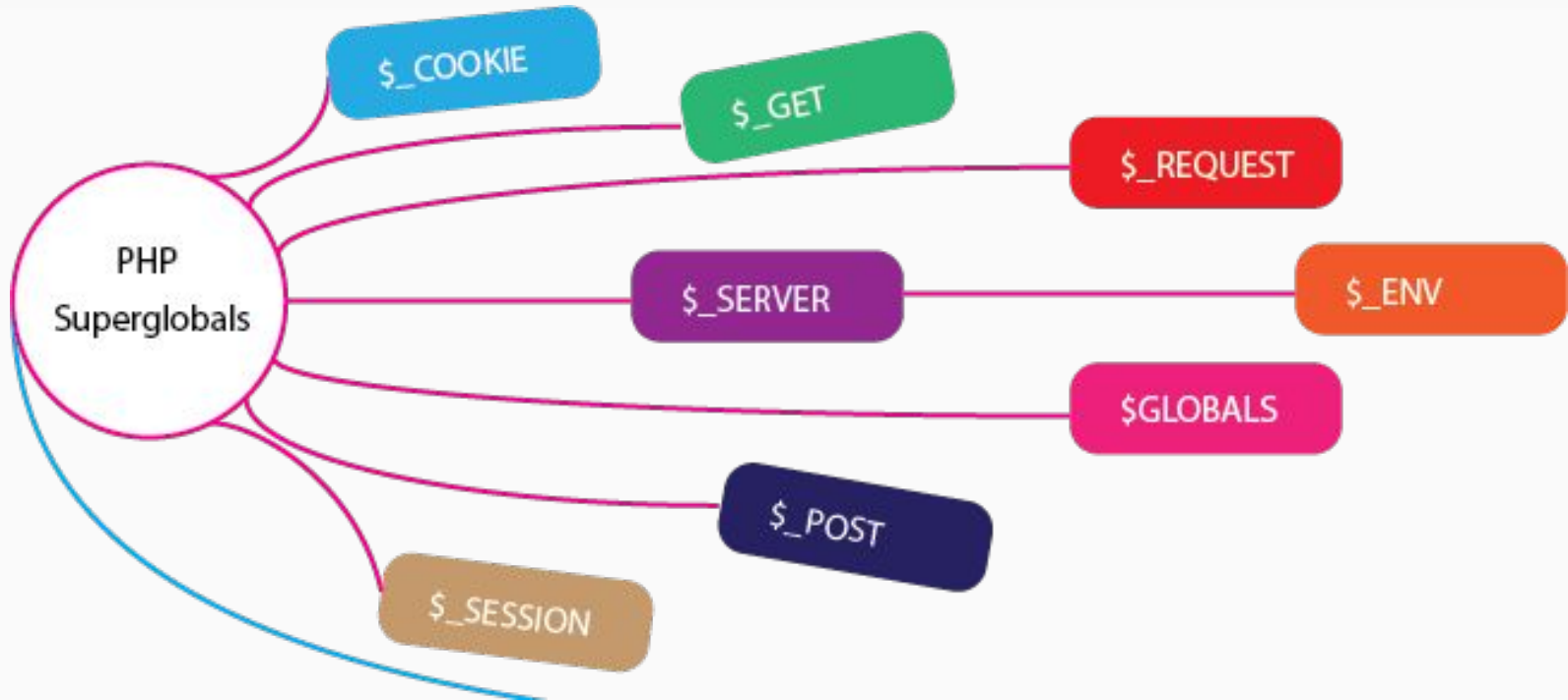
GET vs POST

- POST requests are **never cached**
- POST requests do not remain in the browser history
- POST requests **cannot be bookmarked**
- POST requests have **no restrictions on data length**

GET vs POST

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Superglobals



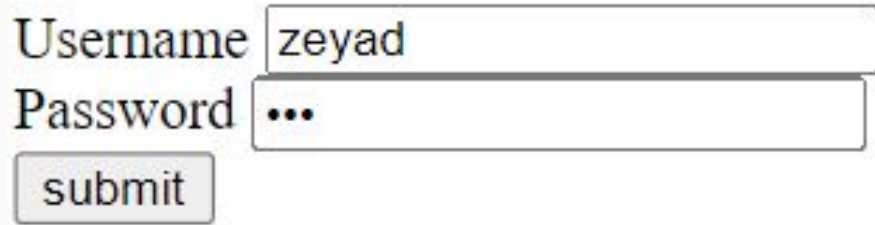
\$_GET

- \$_GET is super global array variable stores the values that come in the URL

```
<form method="GET" action="controller.php">
  <label for="username">Username</label>
  <input type="text" name='username'><br>

  <label for="password">Password</label>
  <input type="password" name='password'><br>

  <input type="submit" value="submit" name='loginform'>
</form>
```



```
<?php
    if(isset($_GET["loginform"])) {
        $username = $_GET["username"];
        $password = $_GET["password"];
        // process data ..
    }
?>
```

<http://localhost/session2/controller.php?username=zeyad&password=123&submit=submit>

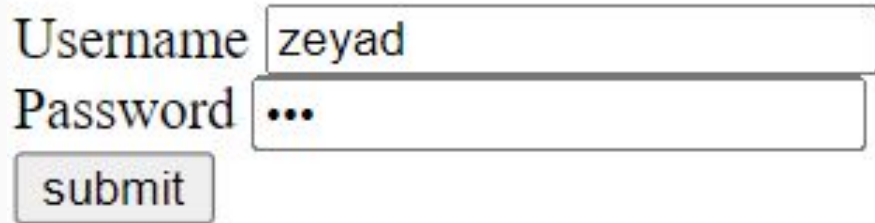
\$_POST

- \$_POST collects data -invisibly- from the submitted HTML form

```
<form method="POST" action="controller.php">
  <label for="username">Username</label>
  <input type="text" name='username'><br>

  <label for="password">Password</label>
  <input type="password" name='password'><br>

  <input type="submit" value="submit" name='loginform'>
</form>
```



Username zeyad

Password ...

submit

```
<?php
    if(isset($_GET["loginform"])) {
        $username = $_POST["username"];
        $password = $_POST["password"];
        // process data ..
    }
?>
```

<http://localhost/session2/controller.php>

- Contains information about headers, paths and script locations

```
'SERVER_SOFTWARE' => string 'Apache/2.4.46 (Win64) PHP/7.4.9' (length=31)
'SERVER_NAME' => string 'localhost' (length=9)
'SERVER_ADDR' => string '::1' (length=3)
'SERVER_PORT' => string '80' (length=2)
'REMOTE_ADDR' => string '::1' (length=3)
'DOCUMENT_ROOT' => string 'C:/wamp64/www' (length=13)
'REQUEST_SCHEME' => string 'http' (length=4)
'CONTEXT_PREFIX' => string '' (length=0)
'CONTEXT_DOCUMENT_ROOT' => string 'C:/wamp64/www' (length=13)
'SERVER_ADMIN' => string 'wampserver@wampserver.invalid' (length=29)
'SCRIPT_FILENAME' => string 'C:/wamp64/www/firstapp/script.php' (length=33)
'REMOTE_PORT' => string '57381' (length=5)
'GATEWAY_INTERFACE' => string 'CGI/1.1' (length=7)
'SERVER_PROTOCOL' => string 'HTTP/1.1' (length=8)
'REQUEST_METHOD' => string 'GET' (length=3)
'QUERY_STRING' => string '' (length=0)
'REQUEST_URI' => string '/firstapp/script.php' (length=20)
'SCRIPT_NAME' => string '/firstapp/script.php' (length=20)
'PHP_SELF' => string '/firstapp/script.php' (length=20)
```

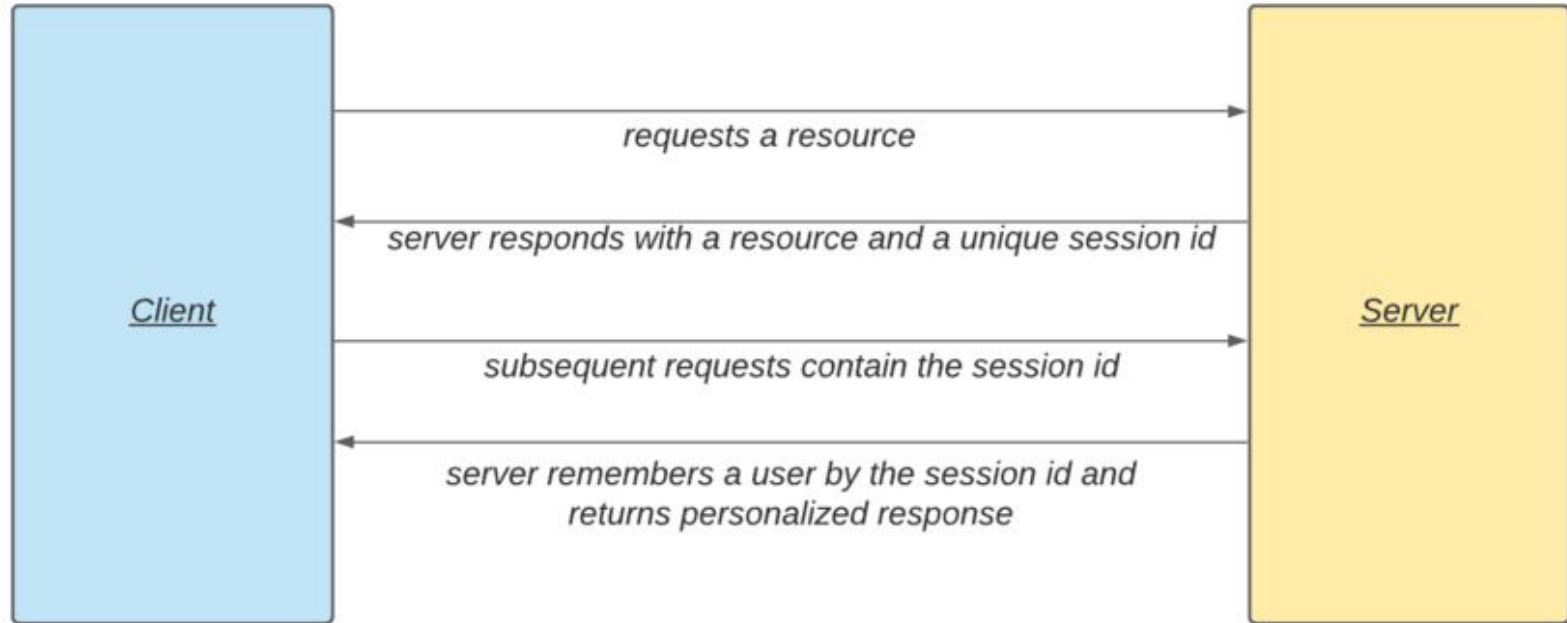

Task 1

- Given a user-registration form, receive the data & validate it

PHP Sessions

- A session is a way to **store information** (in variables) to be used **across multiple pages**.
- Session variables hold information about **one single user**, and are available to all pages in one application
- **\$_SESSION** is also an **associative array** that holds a set of keys and values.

PHP Sessions



PHP Session

- `session_start()` => starts a new session (or resume if one exists).
- `session_unset()` => clear session variables (data).
- `session_destroy()` => delete session.
- `session_id()` => get session id : string

Authentication using PHP Session

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Document</title>
</head>
<body>
  <form method="POST" action="controller.php">
    <label for="username">Username</label>
    <input type="text" name='username'><br>

    <label for="password">Password</label>
    <input type="password" name='password'><br>

    <input type="submit" value="submit" name='loginform'>
  </form>
</body>
</html>
```

Authentication using PHP Session

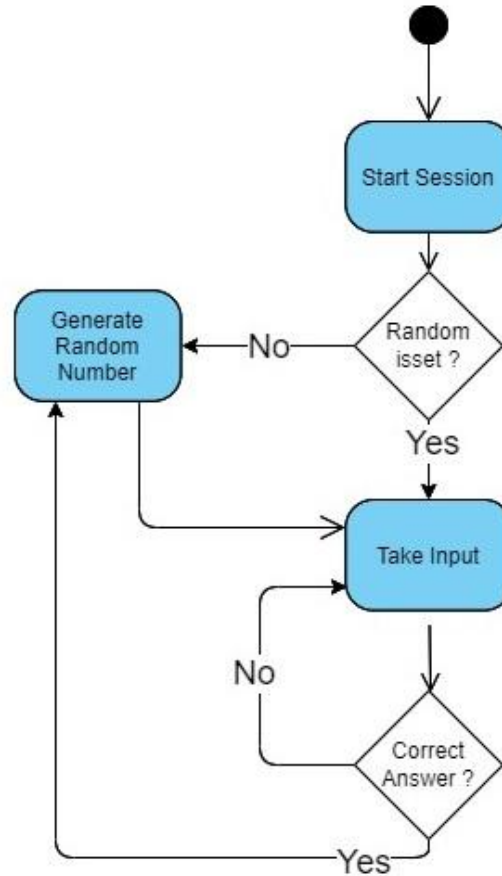
```
<?php
session_start();

    $all_users = [
        "ahmed" => [
            "username" => "ahmed",
            "password" => "12345",
            "major" => "CS"
        ],
        'manar' => [
            "username" => "manar",
            "password" => "54321",
            "major" => "IS"
        ]
    ];
```

Authentication using PHP Session

```
if (isset($_POST["loginform"])) {  
    $username = $_POST["username"];  
    $password = $_POST["password"];  
    if(array_key_exists($username, $all_users)) {  
        if($password == $all_users["$username"]["password"]) {  
            echo "Logged in Successfully";  
            $user = $all_users["$username"];  
            $_SESSION["user"] = $user;  
        }  
        else {  
            echo "Wrong password";  
        }  
    }  
    else {  
        echo "Username is not found, please create a new account";  
    }  
}
```

Task 2 (Implement 'Guess the Number' Game)



Task 2 (Implement 'Guess the Number' Game)

```
<form>
    <input type="number" name="answer">
    <br>
    <input type="submit" value="guess">
</form>
```

```
<?php
    session_start();
    function generateRandom() {
        $_SESSION["random_n"] = rand(1, 100);
    }
    if(!isset($_SESSION["random_n"])) {
        generateRandom();
    }
?>
```

Task 2 (Implement 'Guess the Number' Game)

```
<?php
    if(isset($_GET["answer"])) {
        $answer = $_GET["answer"];
        $correct_answer = $_SESSION["random_n"];
        if($answer == $correct_answer){
            echo "<h1>Your Answer is Correct: $correct_answer";
            echo ", Play Again</h1>";
            // unset($_SESSION["random_n"]); // another solution
            generateRandom();
        }
        else if($answer > $correct_answer)
            echo "<h1>Go Low</h1>";
        else
            echo "<h1>Go High</h1>";
    }
?>
```

Backend Project 1

- Given a front-end project [Blog] (a set of folders and html files), apply authentication (register, login) and allow user to create, read, update, delete posts.