ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

PHP

Sistemas de Informação 1

Credits and Disclaimer

- This material/slides are adapted from:
  - SINF1 2022/23 slides produced by Constantino Martins
  - ARQSI 2014/15 slides
  - Books
  - Web sites:
    - https://www.php.net/;
    - https://www.w3schools.com/
    - ...

ISEP

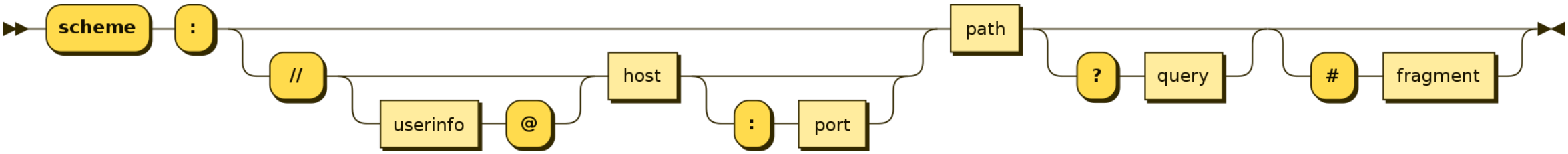
Web Concepts

Concepts

- **Web page or web document**
  - An HTML document (discussed later in PL classes).
- **Home page**
  - The entry page for a website.
- **Website/ Web Application**
  - A collection of one or more significantly linked web pages that describe a body of information.
- **Web server**
  - Computer software and underlying hardware that accepts requests via HTTP/HTTPS.
- **HTTP**
  - The network protocol created to distribute web content. HTTPS is the secure variant of HTTP.

Uniform Resource Location

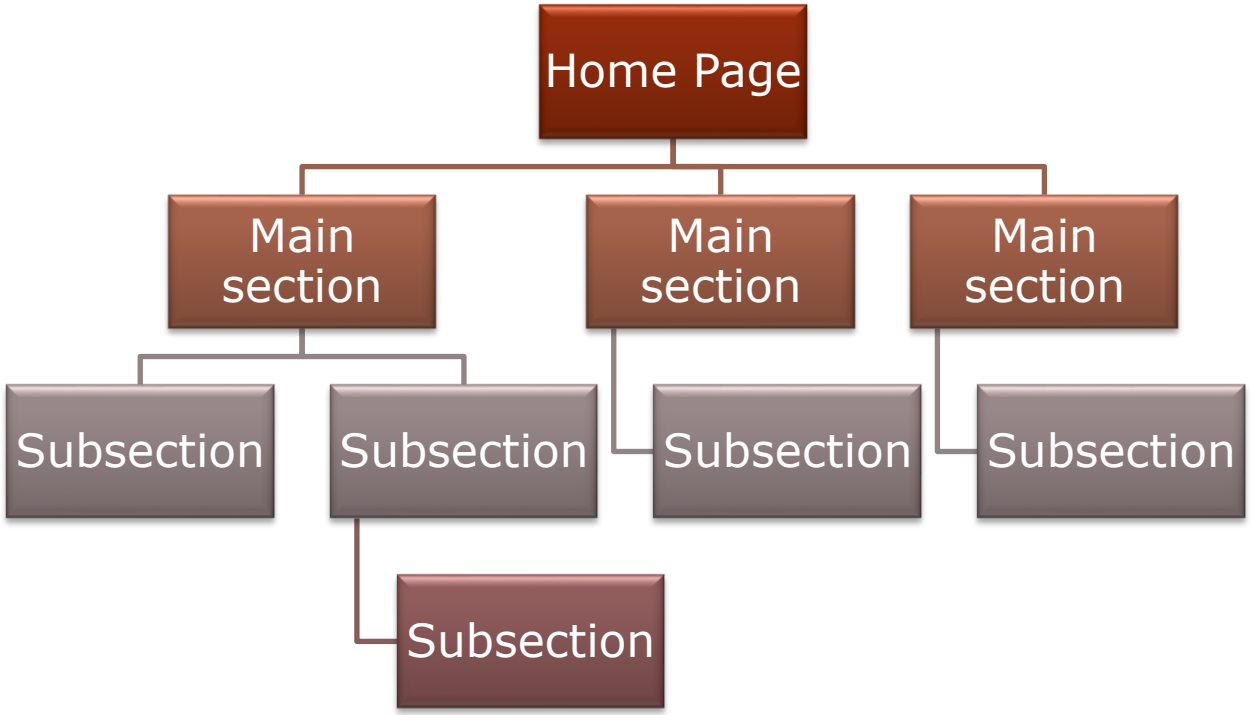
- A Uniform Resource Locator (URL), colloquially termed a web address, is a reference to a web resource that specifies its location on a network and a mechanism for retrieving it.
- URLs occur most commonly to reference web pages (*http*) but are also used for file transfer (*ftp*), email (*mailto*), database access (*JDBC*), ...
- URL syntax:




```
protocol://address/path/filename
```

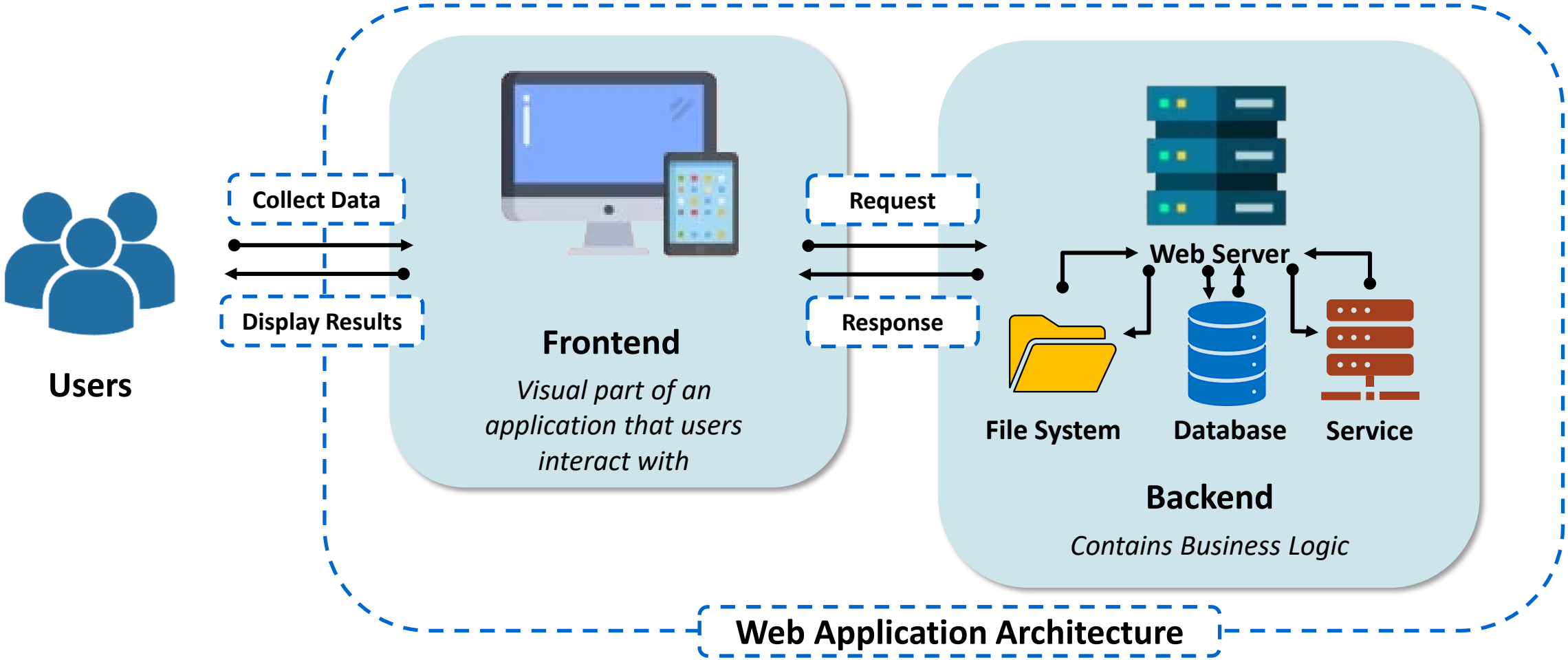
Website vs. Web Application

- A website is **informational**
  - A website is also known as a web presence;
  - A website is a collection of web pages and related content that is identified by a common domain name and published on a web server.



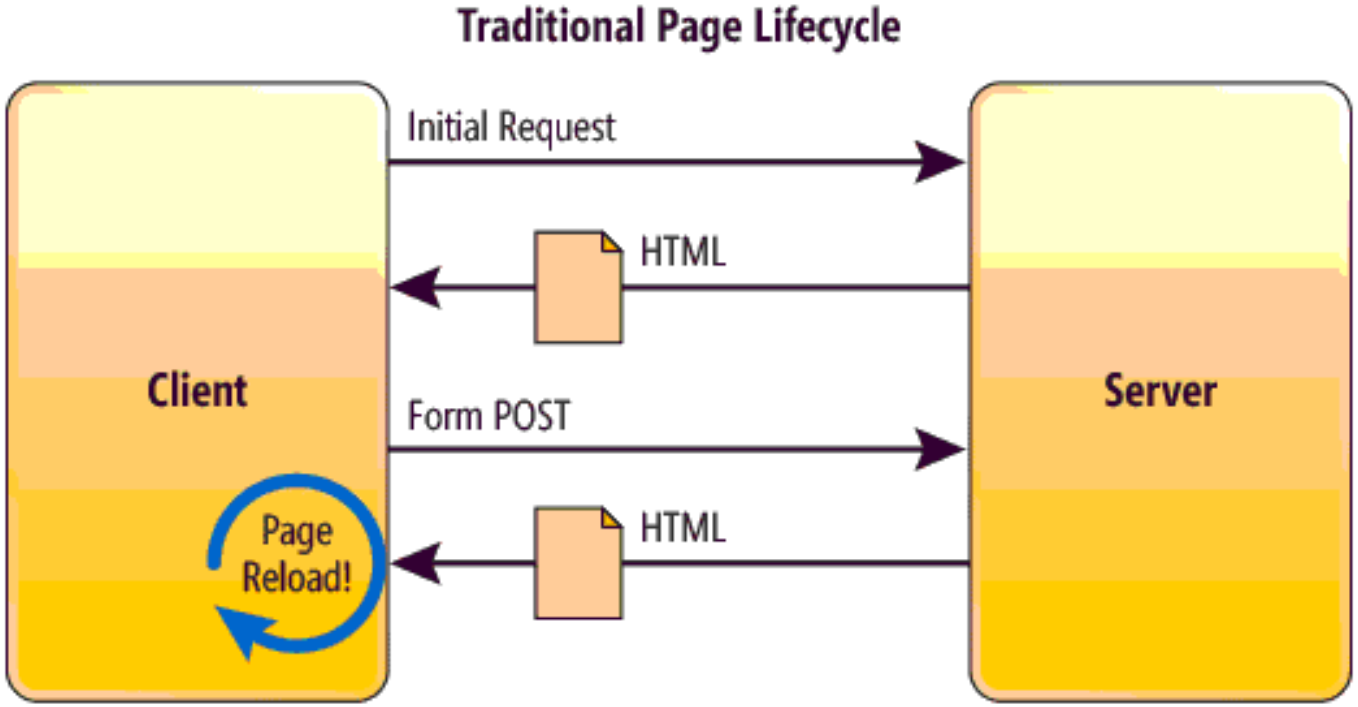
Website vs. Web Application

- A web application is **interactive**



Types of Web Applications

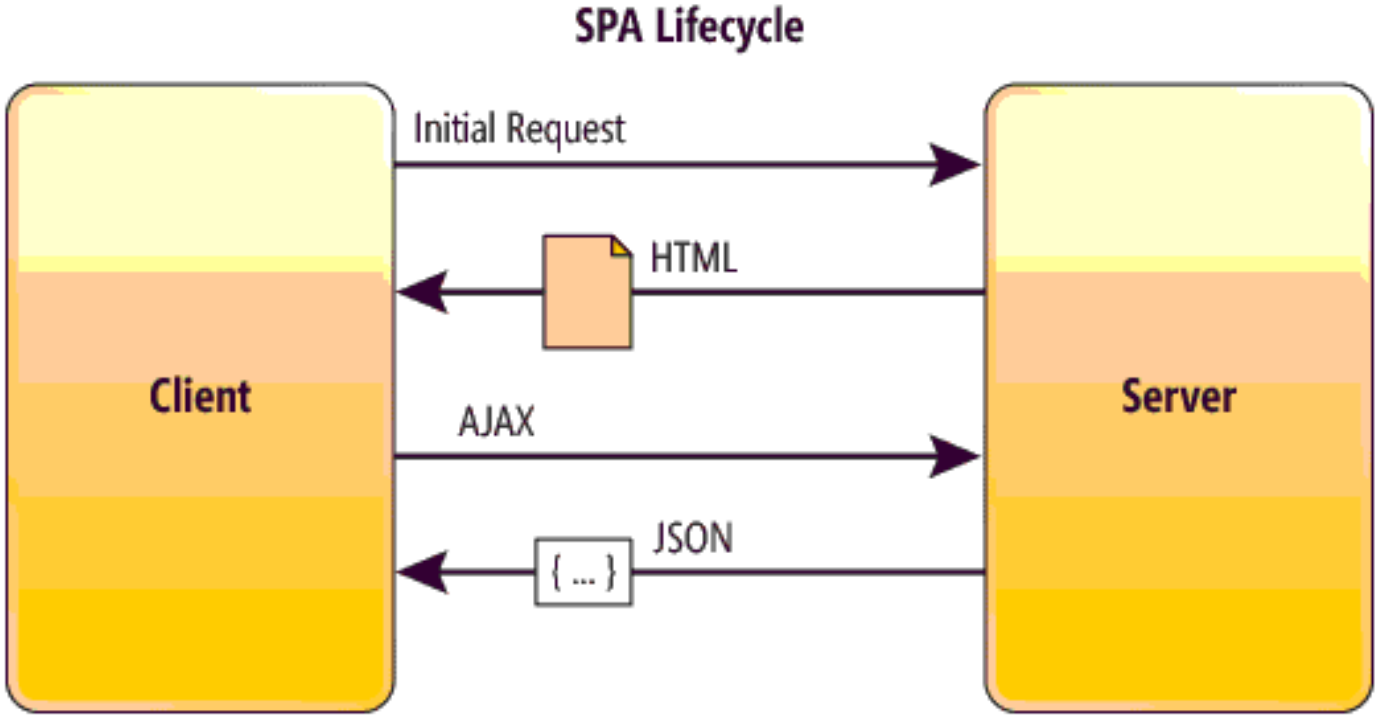
- **Multi Page Application (MPA)**
  - Is the traditional web application architecture that (re)loads the entire web page whenever user interacts with the web application.



cf: https://moz.com/blog/optimizing-angularjs-single-page-applications-googlebot-crawlers

Types of Web Applications

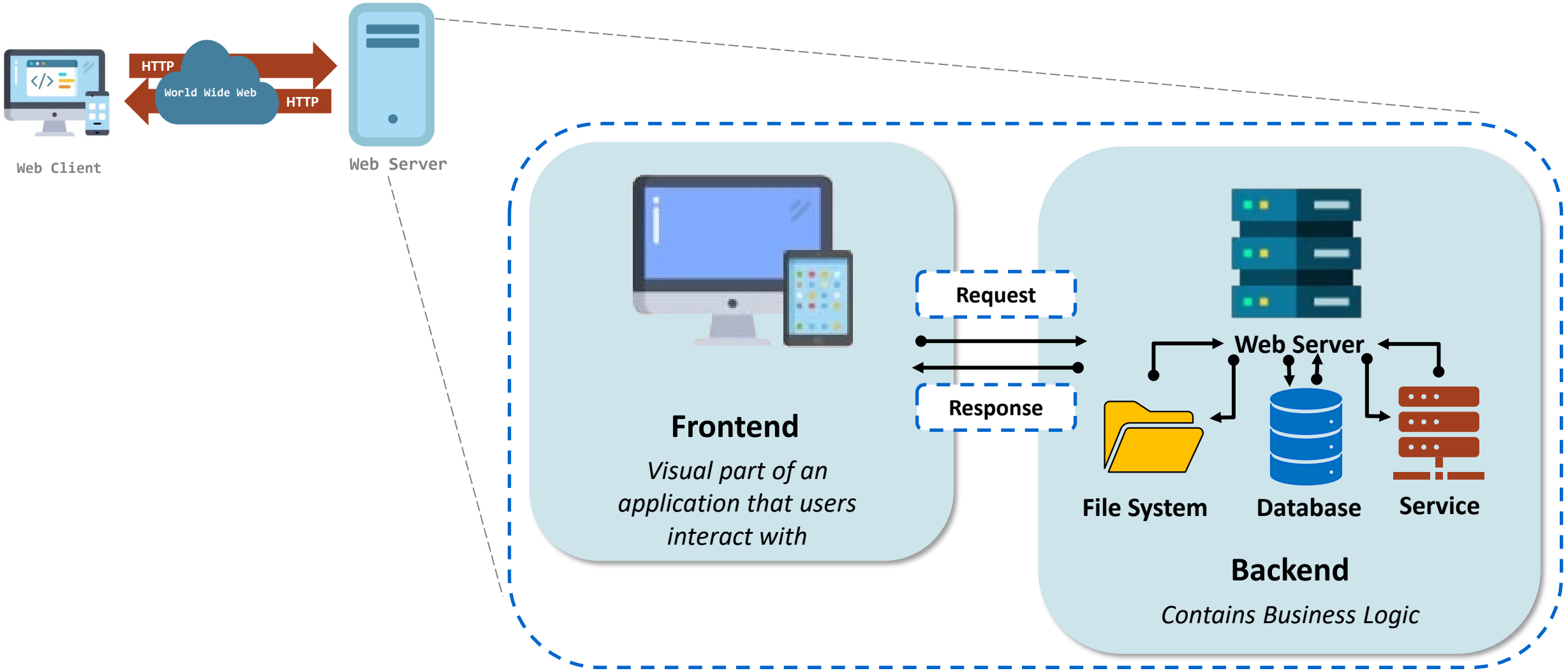
- **Single Page Application (SPA)**
  - Is a web application that re-render its content in response to user interactions without making a request to the server to fetch new web page.



cf: https://moz.com/blog/optimizing-angularjs-single-page-applications-googlebot-crawlers

World Wide Web Architecture

World Wide Web Architecture



HTTP Request Methods

- The HTTP specification includes a collection of methods that are used to interact with server-side resources. There are commonly referred to as **HTTP request methods** or **HTTP verbs** and are intended to cover all possible types of interaction with resources

HTTP GET & HTTP POST

- The GET and POST methods are the two most common HTTP request methods.
- They are used to retrieve or send data to a server.
- They are an integral part of the client-server model that enables the communication between a client and a server through the WWW.

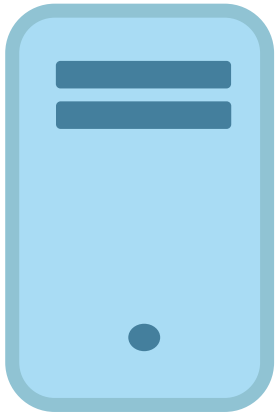
The GET Method

- GET is used to request data from a specified resource. It can retrieve any visible data to a client, such as HTML documents, images, and videos.

Web Client

https://myserver.com/doit.php?name=mary&phone=911234567

HTTP

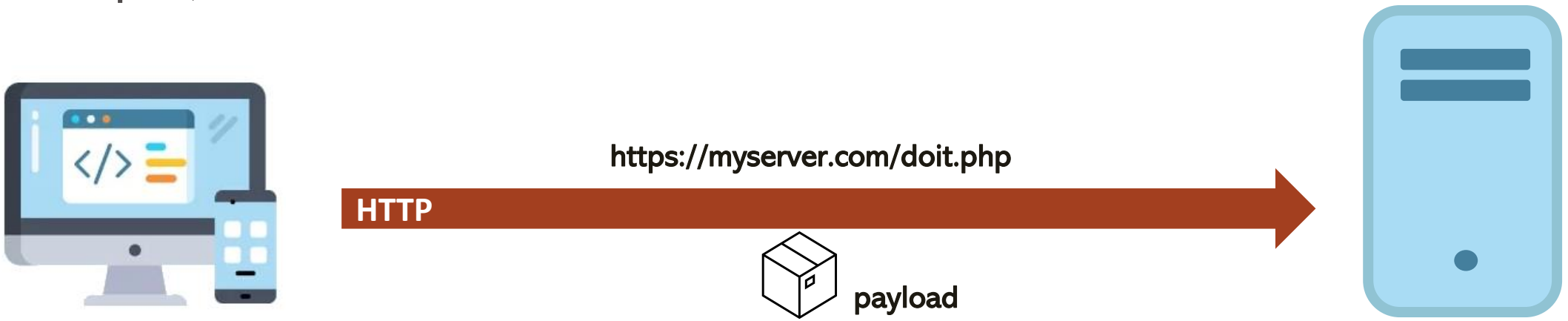


Web Server

- To send a GET request, a client needs to specify the resource it wants to retrieve. The request is then sent to the server, which processes the request and sends the requested data back to the client.

The POST Method

- The POST sends data to a server to create or update a resource. For example, it is often used to submit an HTML form to a server.



- To send a POST request, a client needs to specify the resource to which it wants to send data and the data itself. The request is then sent to the server, which processes the request and sends a response back to the client.

Differences Between GET and POST

1. Visibility

- When using GET, data parameters are included in the URL and visible to everyone.
- When using POST, data is not displayed in the URL but in the HTTP message body.

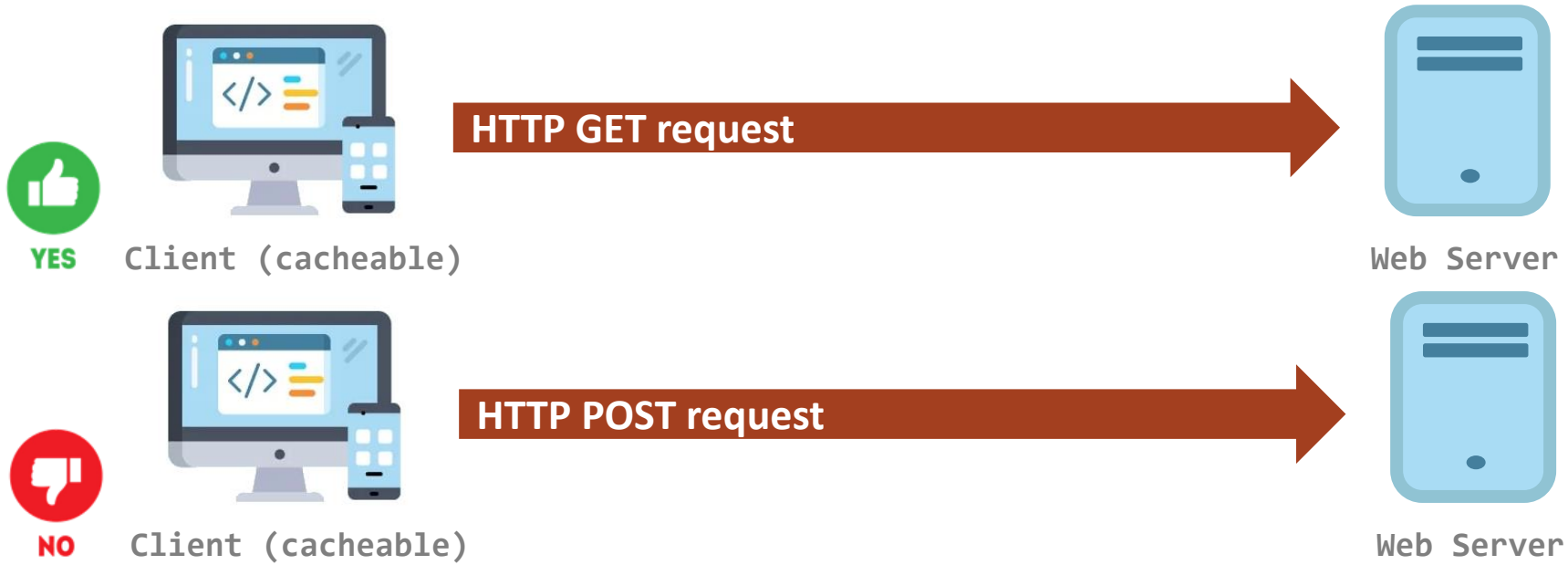
2. Security

- GET is less secure because the URL contains part of the data sent.
- POST is safer because the parameters are not stored in web server logs or the browser history.

Differences Between GET and POST

3. Cache

- GET requests can be cached and remain in the browser history, while POST requests cannot.
  - GET requests can be bookmarked, shared, and revisited, while POST requests cannot.



Differences Between GET and POST

4. Server State

- GET requests are intended to retrieve data from a server and do not modify the server's state.
- POST requests are used to send data to the server for processing and may modify the server's state.

5. Amount of Data Transmitted

- GET method is limited to a maximum number of characters.
  - This is because the GET method sends data through the resource URL, which is limited in length.
- POST method has no limitation.

Differences Between GET and POST

6. Data Type

- GET method supports only string data types.
- POST method supports different data types such as string, numeric, binary, and so on.

7. Idempotent

- An idempotent method has an identical outcome if called multiple times.
- GET method is idempotent.
- POST method is not idempotent.

Differences Between GET and POST

	GET	POST
Safer		✕
Cacheable	✕	
Unlimited in length		✕
Idempotent (fast)	✕	

ISEP

Forms

PHP Form Handling

- The PHP superglobals **$_GET** and **$_POST** are used to collect form-data.
  - $_GET is used to access information from the QueryString (information after '?' in a URL;
  - $_POST contains an array of variables received via the HTTP POST method.
- **$_REQUEST** is a superglobal variable which contains submitted form data, and all cookie data.
  - $_REQUEST is an array containing data from $_GET, $_POST, and $_COOKIE.
  - The $_COOKIE superglobal is an associative array that contains a record of all the cookies values sent by the browser in the current request

PHP Form Handling – GET method

```
<html>
  <head><title>Example</title></head>
  <body>
    <form action="action.php" method="GET">
      Name: <input type="text" name="name"><br>
      Age: <input type="text" name="age"><br>
      <input type=submit>
    </form>
  </body>
</html>
```

```
<?php
   $name=$_GET['name'];
   if(!empty($nome)){
        echo "Name: $name <br>";
   }

   $age=$_GET['age'];
   if(!empty($age)){
        echo "Age: $age <br>";
   }
?>
```

PHP Form Handling – POST method

```
<html>
  <head><title>Example</title></head>
  <body>
    <form action="action.php" method="POST">
      Name: <input type="text" name="name"><br>
      Age: <input type="text" name="age"><br>
      <input type=submit>
    </form>
  </body>
</html>
```

```
<?php
   $name=$_POST['name'];
   if(!empty($nome)){
        echo "Name: $name <br>";
   }

   $age=$_POST['age'];
   if(!empty($age)){
        echo "Age: $age <br>";
   }
?>
```

PHP Form Handling – Method independent

```html
<html>
  <head><title>Example</title></head>
  <body>
    <form action="action.php" method="POST">
      Name: <input type="text" name="name"><br>
      Age: <input type="text" name="age"><br>
      <input type=submit>
    </form>
  </body>
</html>
```

```php
<?php
   $name=$_REQUEST['name'];
   if(!empty($nome)){
        echo "Name: $name <br>";
   }

   $age=$_REQUEST['age'];
   if(!empty($age)){
        echo "Age: $age <br>";
   }
?>
```

PHP Form Handling – Just one file

```php
<?php
   $name=htmlspecialchars($_POST['name']);
   $age= (int) ($_POST['age']);

   if(!empty($nome) && (!empty($age))){
?>
       Hello <?=$name?>.
       Your age is
<?php  echo $age;
   }
   else {
?>
       <form action="" method="POST">
          Name: <input type="text" name="name"><br/>
          Age: <input type="text" name="age"><br/>
          <input type="submit">
       </form>
<?php
   }
?>
```

Convert special characters to HTML entities.
Ensures that all special characters in HTML are encoded correctly so that people cannot inject HTML tags or JavaScript into the webpage

ISEP

Files and URLs

File Reading

- **`fopen`** — Opens file or URL
- Syntax: `fopen(filename, mode, include_path, context)`
  - `filename` - Required. Specifies the file or URL to open
  - `mode` - Required. Specifies the type of access you require to the file/stream.
    - "r" - Read only. Starts at the beginning of the file
    - "r+" - Read/Write. Starts at the beginning of the file
    - "w" - Write only. Opens and truncates the file; or creates a new file if it doesn't exist.
    - "a" - Write only. Opens and writes to the end of the file or creates a new file if it doesn't exist
  - `include_path` - Optional. Set this parameter to '1' if you want to search for the file in the include_path (in php.ini) as well
  - `Context-` Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream

File Reading

```php
<?php
   header('Content-Type: text/xml’);

   /* handle information coming from Google */
   $fpointer = fopen("http://www.rtp.pt/web/rss/gera_rss.php?tema=2","r");

   if($fpointer){
       echo "<h2>Service is temporarily unavailable</h2>";
   }
   else {
       $txt="";
       while ( !feof($fpointer) ){
                           $txt = $txt . fgets($fpointer);
       }
       echo $txt;
       fclose($fpointer);
   }
?>
```

ISEP

Include and Require Statements

PHP Include Files

- The **`include`** (or **`require`**) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.
  - Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.
- The **`include`** and **`require`** statements are identical, except upon failure:
  - `require` will produce a fatal error (`E_COMPILE_ERROR`) and stop the script;
  - `include` will only produce a warning (`E_WARNING`) and the script will continue.

PHP Include Files

- The **`include`** expression includes and evaluates the specified file.

vars.php

```php
<?php

   $color = 'green';
   $fruit = 'apple';

?>
```

test.php

```php
<?php

   echo "A $color $fruit";  // Output: A

   include 'vars.php';

   echo "A $color $fruit";  // Output: A green apple

?>
```

Example

menu.php

```php
<?php

echo '<a href="/default.asp">Home</a> -
      <a href="/html/default.asp">HTML Tutorial</a> -
      <a href="/css/default.asp">CSS Tutorial</a> -
      <a href="/js/default.asp">JavaScript Tutorial</a> -
      <a href="default.asp">PHP Tutorial</a>';
?>
```

app.php

```php
<html>
<body>

    <div class="menu">
       <?php include 'menu.php';?>
    </div>

    <h1>Welcome to my home page!</h1>
    <p>Some text.</p>
    <p>Some more text.</p>

</body>
</html>
```

ISEP

Environment

PHP Environment

- Environment variables are an excellent way to configure PHP applications because they keep the application's settings outside of the code.
  - By doing this, it's easier to prevent secure credentials from being exposed, maintain applications, and use applications across multiple environments.

- **`getenv()`** allows to retrieve an environment variable.
  - If the function is called without an argument, then it returns all available environment variables.
  - If an argument is passed, however, the value of an environment variable with that name is returned.

Example – Getting some System Variables

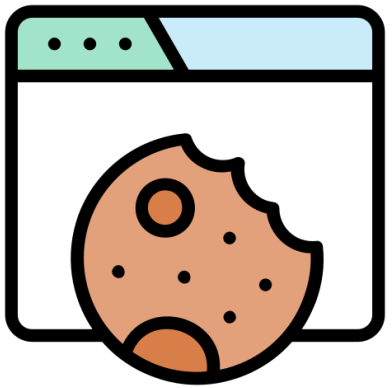
```php
<?php

        $REMOTE_ADDR = getenv("REMOTE_ADDR");
        $HTTP_REFERER = getenv("HTTP_REFERER");
        $DOCUMENT_URI = getenv("REQUEST_URI");
        $HTTP_USER_AGENT = getenv("HTTP_USER_AGENT");

        $REMOTE_HOST = gethostbyaddr($REMOTE_ADDR);
        $HTTP_USER_AGENT = str_replace("|", "", $HTTP_USER_AGENT);

        echo "$REMOTE_ADDR<br>";
        echo "$HTTP_USER_AGENT<br>";
        echo "$REMOTE_HOST<br>";

?>
```

ISEP

Sessions

Cookies

- A cookie is a small file that the server embeds on the user's computer
- Each time the same computer requests a page with a browser, it will send the cookie;
- Cookies are managed by browsers;
  - Most of them offer tools to view and edit these small files and allow you to define the lifecycle for a cookie.
- Each cookie has a cookie name that uniquely identifies it.



Sessions

- A session is a way to store information (in variables) to be used across multiple pages.
- Unlike cookies, session data is stored on the server.
- Session variables hold information about one single user and are available to all pages in one application.

How Sessions Work in PHP

- Starting a Session:
  - A session is started with the **`session_start()`** function.
  - This function first checks if a session is already started and if not, starts one.
  - It is typically the first thing in your PHP script.
- Storing Session Data:
  - After starting a session, you can store and access data in the **`$_SESSION`** superglobal array.
- Ending a Session:
  - A session can be ended by **`session_destroy()`**.
  - This function removes all session data stored in the **`$_SESSION`** array.

Example

```php
<?php

   session_start();
   if (isset($_REQUEST['username']) && isset($_REQUEST['password'])) {
      $username = $_ REQUEST['username'];
      $password = $_ REQUEST['password'];

      $isValid = ValidateUser($username, $password); //function to authenticate user in the DBMS

      if ($isValid == true) {
         $_SESSION['user'] = $username;
         header('Location: main.php');   //redirect to a page
      }
      else {
         echo 'Wrong Authentication';
         session_destroy();
      }
   }

?>
```