Web Programming

Assignment No: II

(SET 1)

**Submitted by:**

Anitta Siby

S7 CSE - B

Roll no.: 18

# During the process of fetching a web page from a web server to a client browser, at what point does an embedded PHP script get executed? What are the two modes that the PHP processor operates in? Distinguish between implode and explode functions in PHP with suitable examples.

1. **Execution of Embedded PHP Script:**

When a web page containing embedded PHP scripts is requested by a client's browser, the PHP code is executed on the server before the page is sent to the client. The process involves the following steps:

1. **Client Request:**

A user, through their web browser, requests a specific web page by entering a URL or clicking on a link.

1. **Server Identification:**

The web server receives the request and identifies that the requested page contains PHP scripts.

1. **PHP Script Execution:**

The PHP processor on the server parses the PHP scripts, executes them, and generates output (which can include HTML, CSS, and other content).

1. **Content Generation:**

The server combines the output of the PHP scripts with any static content (HTML, CSS, etc.) to form the final web page.

1. **Response to Client:**

The server sends the generated web page as a response to the client's browser.

1. **Client Rendering:**

The client's browser receives the response and renders the web page, displaying the content generated by PHP along with any other static elements.

The client never sees the PHP code; they only receive the results of its execution.

1. **PHP Processor Modes:**

PHP operates in two modes:

1. **Command-Line Interface (CLI) Mode:**
   * In CLI mode, PHP scripts are executed directly from the command line.
   * This is useful for running scripts directly from the terminal or as part of batch processes.
   * This mode is often used for running scripts independently of a web server, such as for automation tasks or background processes.
2. **Web Server Module Mode:**
   * In web server module mode, PHP is configured as a module within a web server (e.g., Apache, Nginx).
   * It processes PHP scripts when they are requested by a client's browser and embeds the results into the web page before sending it to the client.
   * When a client requests a PHP-enabled page, the web server hands over the processing of PHP scripts to the PHP module.
   * The PHP module processes the scripts and generates dynamic content, which is then combined with static content and sent to the client's browser.
3. **implode() and explode() Functions:**

Both **implode()** and **explode()** are functions in PHP used for handling strings.

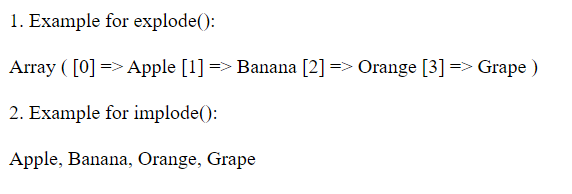
1. **explode():**
   * Syntax: **explode(separator, string, limit)**
   * The **explode()** function is used to split a string into an array based on a specified delimiter.
   * Example:

$str = "Apple, Banana, Orange, Grape";

$arr = explode(", ", $str);

print\_r($arr);

* Output:



**implode():**

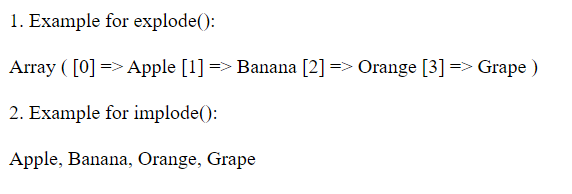
* Syntax: **implode(glue, pieces)**
* The **implode()** function (also known as **join()**) is used to join array elements into a string with a specified delimiter (glue or separator).
* It is the reverse of explode().
  + Example:

$arr = array("Apple", "Banana", "Orange", "Grape");

$str = implode(", ", $arr);

echo $str;

* Output:



# Write a PHP form handling program to perform the user registration of any student portal with a minimum of 5 different fields and insert the data into a MySQL table after establishing necessary connections with the DB.

**testform.html:-**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Student Registration Form</title>

</head>

<body style="text-align: center;">

    <h1>Student Registration Form</h1>

    <form action="testinsert.php" method="post">

        Name: <input type="text" name="inpname" required><br/><br/>

        Email: <input type="email" name="inpemail" required><br/><br/>

        Username: <input type="text" name="inpusername" required><br/><br/>

        Password: <input type="password" name="inppassword" required><br/><br/>

        Course: <input type="text" name="inpcourse" required><br/><br/>

        <input type="submit" name="submit" value="Register"/><br/>

    </form>

</body>

</html>

**testinsert.php:-**

<?php

    $dbname="testdb";

    $conn = mysqli\_connect("localhost","root","","testdb");

    if(!$conn) {

        die("Connection failed : " .mysqli\_connect\_error());

    }

    $name = $email = $username = $password = $course = "";

    if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

        $name = test\_input($\_POST["inpname"]);

        $email = test\_input($\_POST["inpemail"]);

        $username = test\_input($\_POST["inpusername"]);

        $password = password\_hash(test\_input($\_POST["inppassword"]), PASSWORD\_DEFAULT);

        $course = test\_input($\_POST["inpcourse"]);

        $sql = "INSERT INTO studentreg (name, email, username, password, course) VALUES ('$name', '$email', '$username', '$password', '$course')";

        $res = mysqli\_query($conn,$sql);

        if($res=== TRUE) {

            echo "<br/>Registration successful ! ";

        }

        else {

            echo "Error : ".mysqli\_error($conn);

        }

    }

    mysqli\_close($conn);

    function test\_input($data) {

        $data = trim($data);

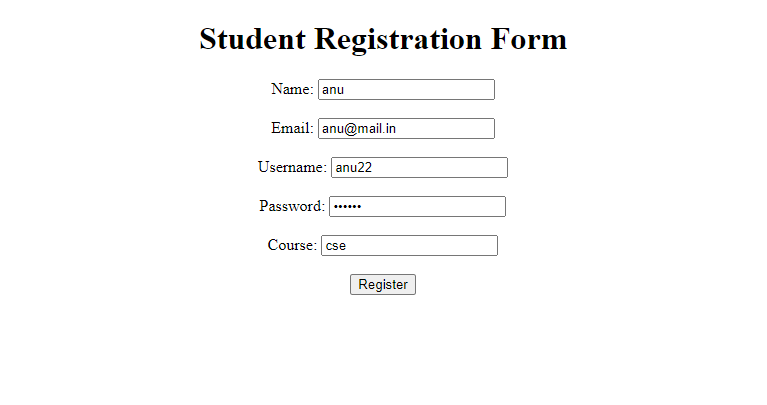
        $data = stripslashes($data);

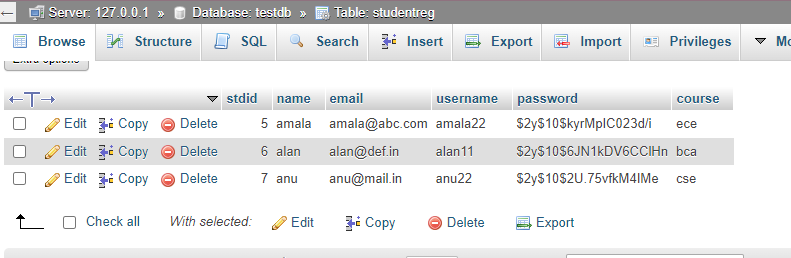
        $data = htmlspecialchars($data);

        return $data;

    }

?>





# Enumerate the data types in JSON. Illustrate the document definition of a ‘Employee document ‘using JSON Schema.

1. **Data Types in JSON:**

JSON (JavaScript Object Notation) is a lightweight data interchange format that supports several data types. The basic data types in JSON include:

1. **String:**

* A sequence of characters enclosed in double quotation marks.
* Eg: "java programming"

1. **Number:**

* A numeric value (integer or floating-point).
* Eg: 42

1. **Boolean:**

* Represents either **true** or **false**.
* Eg: true

1. **Array:**

* An ordered list of values.
* Eg: [1, 2, 3]

1. **Object:**

* An unordered collection of key-value pairs.
* Eg:

{

"key1": "value1",

"key2": "value2"

}

1. **Null:**

* Represents a null value.
* Eg: null

1. **‘Employee document ‘using JSON Schema**

JSON Schema allows us to annotate and validate JSON documents. It provides a way to describe the structure and constraints of JSON data. Below is an example JSON Schema definition for an 'Employee Document':

Example for Employee Document:-

{

    "id": 110,

    "name": "John Doe",

    "age": 28,

    "department": "Marketing",

    "salary": 50000,

    "active": true,

    "skills": ["English", "Content Creation", "Social Media Management"],

    "hobbies": {

        "indoor": ["Chess", "Reading"],

        "outdoor": ["Cycling", "Football"]

    }

}

JSON Scheme:-

{

    "$schema": "http://json-schema.org/draft-07/schema#",

    "title": "Record of Employee",

    "$id": "https://example.com/employee.schema.json",

    "description": "This document records the details of an employee",

    "type": "object",

    "properties": {

        "id": {

            "type": "integer",

            "description": "Employee ID"

        },

        "name": {

            "type": "string",

            "description": "Employee's full name"

        },

        "age": {

            "type": "integer",

            "minimum": 18,

            "description": "Employee's age"

        },

        "department": {

            "type": "string",

            "description": "Employee's department"

        },

        "salary": {

            "type": "number",

            "minimum": 0,

            "description": "Employee's salary"

        },

        "active": {

            "type": "boolean",

            "description": "Employee's active status"

        },

        "skills": {

            "type": "array",

            "items": {

                "type": "string"

            },

            "description": "list of employee's skills"

        },

        "hobbies": {

            "type": "object",

            "properties": {

                "indoor": {

                    "type": "array",

                    "items": {

                        "type": "string"

                    },

                    "description": "List of indoor hobbies"

                },

                "outdoor": {

                    "type": "array",

                    "items": {

                        "type": "string"

                    },

                    "description": "List of outdoor hobbies"

                }

            },

            "description": "Indoor and outdoor hobbies"

        }

    },

**"required": ["id", "name", "age", "department", "salary", "active", "skills", "hobbies"]**

}

* In this example, the JSON Schema defines an 'Employee' object with properties such as ID, name, age, department, salary, active status, skills, and hobbies.
* Here, “skills” property is an array of strings and “hobbies” is an array of objects.
* Constraints like data types, minimum age, minimum salary, and required fields are also specified.