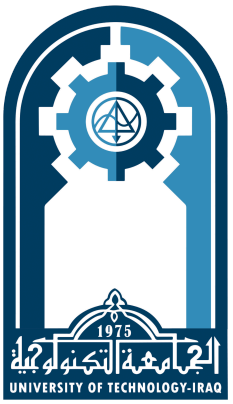

WEB LAB

Name : Zumuruda ahmed mohammed

Fourth stage - network - morning

Discussion

1- What are the differences between GET & POST methods?

GET method:

- GET requests are typically used to request data from a server.
- Data is sent via the URL in the form of query parameters.
- Parameters are appended to the URL after a question mark (`?`) **and are separated by ampersands (`&`).**
- **GET requests can be bookmarked and cached, as they are visible in the URL.**
- **GET requests have a limitation on the amount of data that can be sent (URL length limitations), which is usually around 2048 characters, depending on the browser and server configuration.**
- **GET requests should not be used for sensitive data (such as passwords) because the data is visible in the URL.**

POST method:

- POST requests are used to submit data to a server, typically to modify or update server-side resources.
- Data is sent in the request body, not in the URL, making it more secure compared to GET requests.
- POST requests can send larger amounts of data compared to GET requests, as the data is not limited by URL length constraints.
- POST requests are not bookmarkable and generally not cached.
- POST requests are preferred for submitting sensitive data, such as passwords, as the data is not visible in the URL.

2- Write a PHP script to create a form for entering three names and their emails and then print the names with the emails on the web page as sorting ascending. (Use GET method in your form).

```php
<!DOCTYPE html>
<html>
<head>
    <title>Sort Names and Emails</title>
</head>
<body>

<?php
// Define an array to store names and emails
$contacts = [];

// Check if the form is submitted
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    // Get data from the form
    $name1 = $_GET['name1'];
    $email1 = $_GET['email1'];
    $name2 = $_GET['name2'];
    $email2 = $_GET['email2'];
    $name3 = $_GET['name3'];
    $email3 = $_GET['email3'];

    // Add names and emails to the contacts array
    $contacts[$name1] = $email1;
    $contacts[$name2] = $email2;
    $contacts[$name3] = $email3;

    // Sort the contacts array by keys (names) in ascending order
    ksort($contacts);
}
?>

<h2>Enter Names and Emails</h2>
<form method="get" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
    Name 1: <input type="text" name="name1"><br>
    Email 1: <input type="text" name="email1"><br>
    Name 2: <input type="text" name="name2"><br>
    Email 2: <input type="text" name="email2"><br>
    Name 3: <input type="text" name="name3"><br>
    Email 3: <input type="text" name="email3"><br>
    <input type="submit" value="Submit">
</form>

<?php
// Check if contacts array is not empty
if (!empty($contacts)) {
```

```
    // Display sorted names and emails
    echo "<h2>Sorted Names and Emails:</h2>";
    echo "<ul>";
    foreach ($contacts as $name => $email) {
        echo "<li>$name - $email</li>";
    }
    echo "</ul>";
}
?>

</body>
</html>
```

3- There are many other methods besides those mentioned in this lecture that could be used to avoid hacking operations. Mention some of them and give an example of one of them.

There are several methods and best practices to mitigate hacking operations and enhance the security of web applications

1. Input Validation and Sanitization:

- Validate and sanitize user input to ensure it meets expected formats and doesn't contain malicious content.
- Example:

$username = $_POST['username'];
$sanitized_username = filter_var($username, FILTER_SANITIZE_STRING);
// Use $sanitized_username in your application

2. Parameterized Queries (Prepared Statements):

- Use parameterized queries or prepared statements when interacting with databases to prevent SQL injection attacks.
- Example using PDO:

```
$stmt = $pdo->prepare('SELECT * FROM users WHERE username = :username');
$stmt->execute(['username' => $username]);
```

3. Cross-Site Scripting (XSS) Prevention:

- Escape user-generated content when displaying it in HTML to prevent XSS attacks.
- Example using htmlspecialchars():

```
echo htmlspecialchars($user_input);
```