PHP Introduction

DWES UD2

3.- PHP Basic Features

Web Forms

Web forms are the tools that allow data entered by the user to be collected and processed on the server.

It is important to choose well the method (**method**) with which the form will be sent (**get** or **post**).

Remember that in the **action** attribute of the form you must indicate the script that will receive the data.

For a form field to be submitted it must have the **name** attribute

3.- PHP Basic Features

Web Forms

```
<form action="procesa.php" method="post">
    Nombre del alumno: <input type="text" name="nombre" id="nombre"><br>
    Apellidos del alumno: <input type="text" name="apellidos" id="apellidos"><br>

    Ciclo que cursa:<br>
    <input type="radio" name="ciclo" value="DAW"> Des. de ap. web
    <br>
    <input type="radio" name="ciclo" value="DAM"> Des. de ap. multiplataforma
    <br>  <br>

    <input type="submit" value="Enviar">
</form>
```

3.- PHP Basic Features

Web Forms

```
<form name="input" action="#" method="post">
    Nombre del alumno: <input type="text" name="nombre"><br>
    Ciclos que cursa:<br>

    <input type="checkbox" name="modulos[ ]" value="DWES">
    Desarrollo web en entorno servidor<br>
    <input type="checkbox" name="modulos[ ]" value="DWEC">
    Desarrollo web en entorno cliente<br>
    <br>

    <input type="submit" value="Enviar">
</form>
```

3.- PHP Basic Features

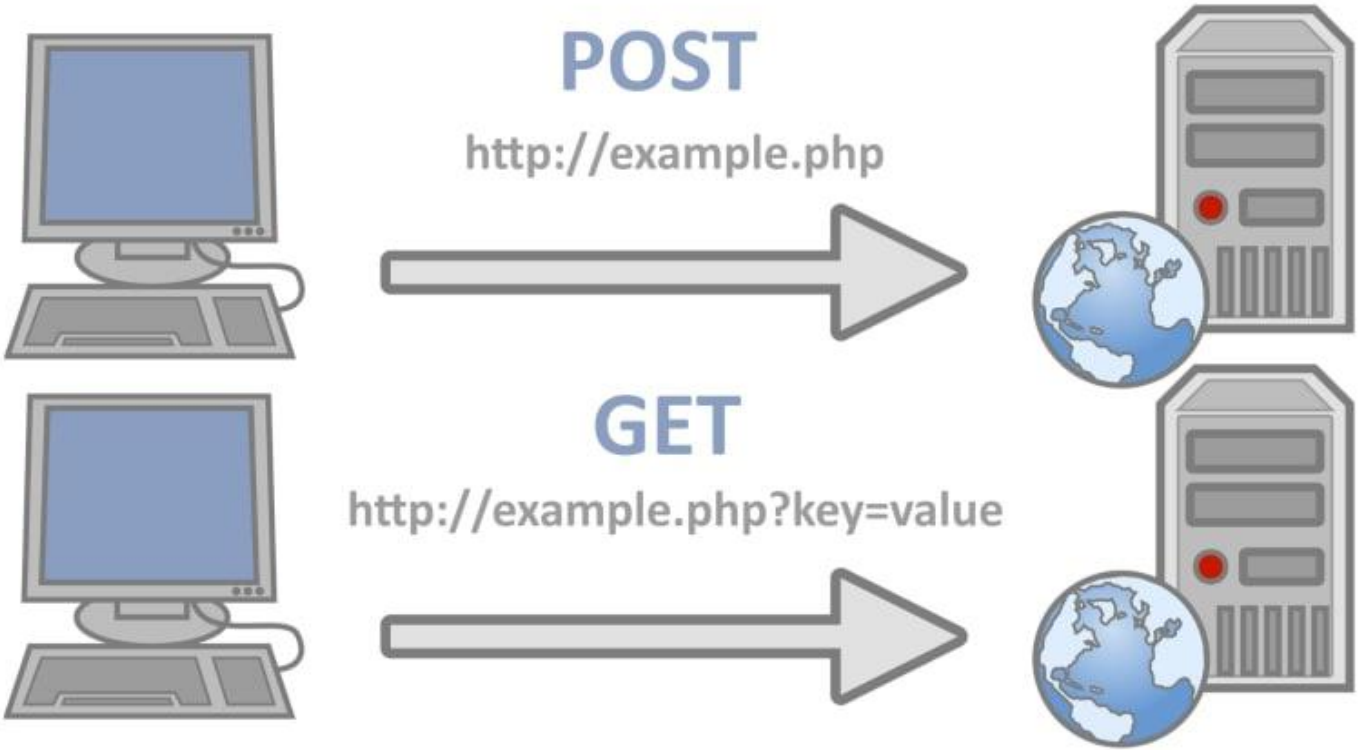
Web Forms

In the case that a checkbox can send several values as in the previous event, it must be indicated in the name attribute that it is an array.

```
<input type="checkbox" name="modulos[ ]" value="DWEC">
```

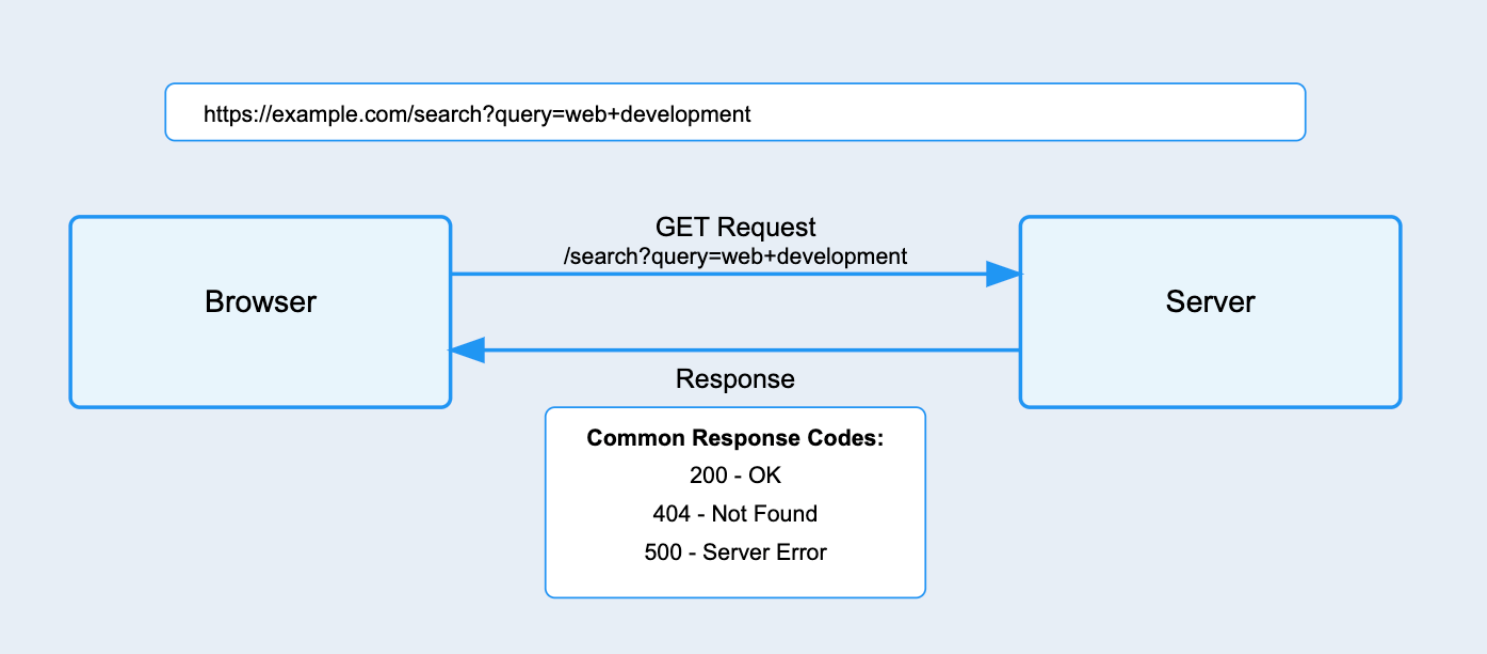
3.- PHP Basic Features

Web Forms- Get vs Post



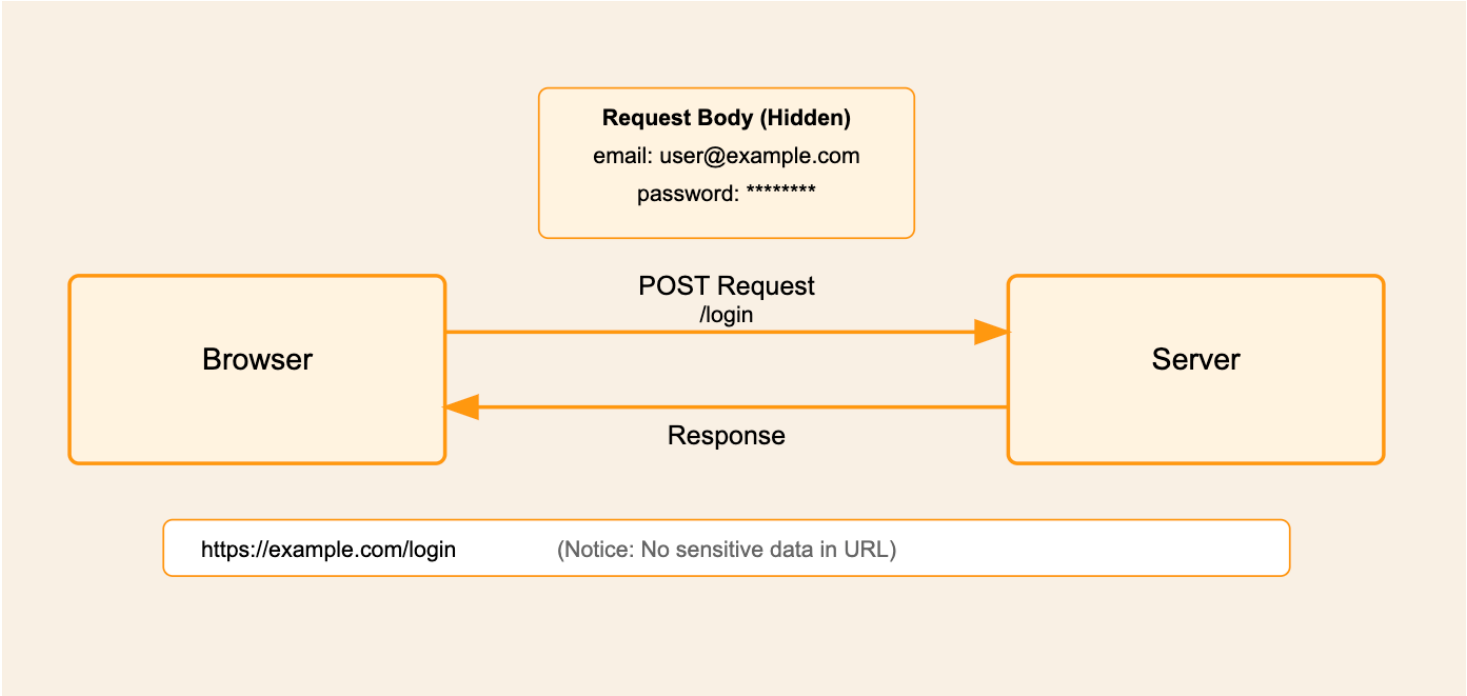
3.- PHP Basic Features

Web Forms – Get vs Post



3.- PHP Basic Features

Web Forms – Get vs Post



3.- PHP Basic Features

Web Forms – Get vs Post

Feature	GET	POST
Primary Use	Retrieving data	Submitting data
Data Location	URL parameters	Request body
Data Visibility	Visible in URL	Hidden in request body
Maximum Size	Limited (~2048 chars)	No practical limit
Caching	Can be cached	Not cached
Bookmarking	Supported	Not supported
Browser History	Saved	Not saved
Security	Less secure	More secure
Idempotency	Yes (same result)	No (may change)

3.- PHP Basic Features

Web Forms – Get vs Post

Security should always be your primary concern when choosing between GET and POST. While HTTPS encrypts all data in transit for both methods, POST requests offer an additional layer of privacy by keeping data out of URLs. This matters because:

- URLs often get logged in server logs, browser histories, and proxy servers
- URLs can be accidentally shared in screenshots or bookmarks
- API keys or tokens in URLs might get exposed through referrer headers
- Browser extensions can read and modify URL parameters

3.- PHP Basic Features

Use GET for:

- Search functionality
- Product listings
- Article pages
- User profiles
- Dashboard views

Use POST for:

- Login/Registration forms
- File uploads
- Comment submissions
- Creating new content
- Updating user settings

3.- PHP Basic Features

Take into account:

- Use GET when you want the request to be bookmarkable, shareable, and cached
- Use POST when dealing with sensitive data, file uploads, or creating/updating resources
- Consider the visibility, security, and size of your data when choosing between them

3.- PHP Basic Features

Web Forms- Get vs Post

If we process the following form with the Get method:

```
<form name="input" action="procesa_get.php" method="get">
    Nombre del alumno: <input type="text" name="nombre" id="nombre"><br>
    Apellidos del alumno: <input type="text" name="apellidos" id="apellidos"><br>
    Ciclo que cursa:<br>
    <input type="radio" name="ciclo" value="DAW"> Des. de ap. web
    <br>
    <input type="radio" name="ciclo" value="DAM"> Des. de ap. multiplataforma
    <br><br>
    <input type="submit" value="Enviar">
</form>
```

3.- PHP Basic Features

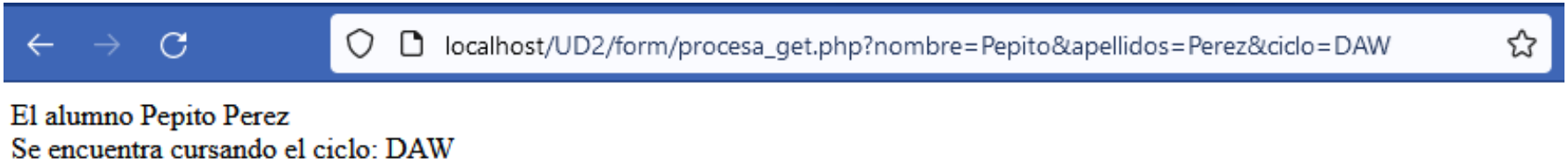
Web Forms

In the file **procesa_get.php**:

```
<?php
        echo 'El alumno ';
        echo $_GET['nombre'] .' '. $_GET['apellidos'];
        echo '<br>Se encuentra cursando el ciclo: ';
        echo $_GET['ciclo'];
?>
```

3.- PHP Basic Features

The result is as follows:



The information entered is shown on the screen.

The parameters that have been passed to the form are shown in the URL.

3.- PHP Basic Features

Web Forms- Get vs Post

If we process the same form but with the Post method:

```
<form name="input" action="procesa_post.php" method="post">
    Nombre del alumno: <input type="text" name="nombre" id="nombre"><br>
    Apellidos del alumno: <input type="text" name="apellidos" id="apellidos"><br>
    Ciclo que cursa:<br>
    <input type="radio" name="ciclo" value="DAW"> Des. de ap. web
    <br>
    <input type="radio" name="ciclo" value="DAM"> Des. de ap. multiplataforma
    <br><br>
    <input type="submit" value="Enviar">
</form>
```

3.- PHP Basic Features

Web Forms

In the file **procesa_post.php**:

```
<?php
        echo 'El alumno ';
        echo $_POST['nombre'] .' '. $_POST['apellidos'];
        echo '<br>Se encuentra cursando el ciclo: ';
        echo $_POST['ciclo'];
?>
```

3.- PHP Basic Features

The result is as follows:

The information entered is shown on the screen.

The parameters that have been passed to the form are NOT shown in the URL.

3.- PHP Basic Features

Web Forms

Sometimes it may be interesting for the variables sent to be an array:

```
<form name="input" action="#" method="post">
        Nombre: <input type="text" name="propio[nombre]"><br>
        Apellidos: <input type="text" name="propio[apellidos]"><br>
        Nombre: <input type="text" name="conyuge[nombre]"><br>
        Apellidos: <input type="text" name="conyuge[apellidos]"><br>
        <br>
        <input type="submit">
</form>
```

3.- PHP Basic Features

Web Forms

Data validation is very important and should be done in 3 places:

- **Browser:** by using the correct types in the **input** fields and the **required** attribute.
- **Client** (JavaScript): before they are sent in order to avoid overloading the server.
- **Server:** to avoid identity theft, for example someone creating their own form. This is done by using JavaScript and PHP together using an **input type="hidden**".

3.- PHP Basic Features

Web Forms

It is common for the same page that displays the form to be the one that processes the data it sends. **action="#"**

If all the data is correct, the appropriate action is taken.

If there are fields with errors, for example a person's name with numbers, the form is displayed again, filling out the fields with the values that were entered but indicating which fields contain errors. For this, the **value** attribute of the **input** and **checked** of the **check boxes** is used.

To do the latter, the **isset** function is used to know if variables arrive in the file from the method with which the form is sent.

3.- PHP Basic Features

Web Forms - Data Validation

location → If a specific field does not arrive, reload the page without data (possible impersonation)

If not, validate fields and depending on errors...

isset, is_numeric, strcmp()...

3.- PHP Basic Features

Regular expressions

preg_match($expr, $cadena): Searches “$string” for a match with the regular expression “$expr”. Returns 1 if it matches, 0 if it doesn't, or false if an error occurred.

It is not recommended to use regular expressions to validate email addresses. Instead it is recommended to use filter_var, which returns false if the applied filter fails, with the mail filter 'FILTER_VALIDATE_EMAIL’:

```
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo "Formato de correo no válido";
    }
```

Exercise

In the project you added a form in which you tested the different types of data and with which the user could send a query.

If you don't have them, add a **checkbox** type field and **input type="date"**.

Create a file called **consulta.php** in which the data received from the form will be displayed.

This document must have the same header and footer as the rest of the files.

Exercise

Web Forms - Data Validation

Create a file called **registro.php** in the project. This document must have the same header and footer as the rest of the files.

This document must have a registration form with the following fields: name, surname, username, password, email, date of birth, gender, acceptance of conditions, and acceptance of sending advertising.

The password field should be duplicated so that you can verify that it has been entered correctly.

The form will have to be sent to the same script (**registro.php**). If errors are found, the form with the data sent will be displayed and these errors will be indicated. If everything is correct, instead of displaying the form, a message will be displayed indicating that the registration has been completed correctly.

It will be necessary to validate that all fields are filled in except for the advertising field. That they contain the correct data type, that the passwords match, that the email contains a valid address (contains @ and domain - regular expressions).