Introducció

- A les pàgines web actuals, els clients (també anomenats navegadors) no només obtenen un element HTML del servidor, sinó que també envien informació.

- La informació que envien pot ser:
  - El text de cerca que l'usuari ha escrit al motor de cerca
  - El contingut dels formularis
  - Aplicar filtres per restringir o ordenar els productes d'una botiga online segons criteris específics
  - Ordenar els resultats mostrats a la pàgina d'acord amb un criteri específic

- Per enviar certs tipus d'informació al servidor, el protocol HTTP proporciona diferents mètodes de petició.

- Els mètodes GET i POST són dues de les eines fonamentals que defineixen la comunicació entre el client i el servidor dins del protocol HTTP.

Protocol HTTP

- HTTP és un protocol sense estat (stateless) que regula l'intercanvi de dades entre el client (navegador) i el servidor. És el protocol de comunicació principal d'Internet, basat en un model client-servidor.

- Què vol dir **protocol sense estat**?
  - Cada sol·licitud realitzada pel client al servidor és independent de les altres, i el servidor no recorda cap informació sobre les peticions anteriors un cop es completa una petició.
  - El servidor no guarda cap "memòria" sobre l'estat de la connexió després de completar la resposta.

- Cada vegada que un client fa una sol·licitud a un servidor, aquesta es materialitza mitjançant un mètode HTTP que dicta el propòsit de la comunicació.

Estructura dels missatges HTTP

- Els missatges HTTP poden ser:
    - Peticions dels clients (request): Sol·licituds enviades al servidor.
    - Respostes dels servidors (response): Respostes del servidor al client.

Sol·licitud HTTP (Request)

- Estructura d’una petició

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html
```

- Línia de petició: GET /index.html HTTP/1.1 indica el mètode, la ruta i la versió d'HTTP.
- Capçaleres: Proporcionen metadades (Host, User-Agent, Accept, etc.).
- Cos (body): Utilitzat només amb alguns mètodes com POST.

Resposta HTTP (Response)

- Estructura d'una resposta

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 349
Date: Mon, 26 Nov 2024 12:00:00 GMT
```

- Línia d'estat: Indica el protocol, codi d'estat i descripció (200 OK).
- Capçaleres: Conté informació com el tipus de contingut, data, etc.
- Cos (body): El contingut de la resposta (HTML, JSON, etc.).

Procés de comunicació HTTP

Per què es fa servir HTTP?

- Quan es va crear, el protocol HTTP només servia per sol·licitar documents HTML a un servidor web.

- Actualment, en canvi, es fa servir amb gran varietat de fins:
  - Els navegadors usen HTTP per sol·licitar qualsevol tipus de fitxer de text o de vídeo.
  - Els programes d'aplicació utilitzen HTTP per carregar fitxers i actualitzacions.
  - L'API basada en REST és una solució que utilitza HTTP per controlar serveis web.
  - Les operacions d'accés a bases de dades a la web.

Capçaleres: Peticions i Respostes

- Les capçaleres HTTP són elements clau en les comunicacions client-servidor.
- Proporcionen metadades sobre la petició o la resposta HTTP, com el tipus de contingut, codificació, durada de la resposta, i molt més.
- Les capçaleres enviades pel client cap al servidor (peticions) inclouen informació com el tipus de contingut acceptat, el navegador utilitzat (User-Agent), i les cookies.
- Les capçaleres enviades pel servidor cap al client (respostes) inclouen informació com l'estat de la resposta, el tipus de contingut retornat (Content-Type), i la durada de la cache (Cache-Control).

Capçaleres: Peticions i Respostes

Exemple: Obtenir les capçaleres de resposta d'una URL

- La funció get_headers() en PHP permet obtenir les capçaleres d'una resposta HTTP des d'una URL.

```
<?php
$url = "https://www.example.com"; // URL de prova
$headers = get_headers($url, 1); // Obtenim només les capçaleres (1 inclou un
array associatiu)
echo "<h3>Capçaleres de la URL:</h3>";
foreach ($headers as $clau => $valor) {
        if (is_array($valor)) { // Si hi ha múltiples valors per una clau (ex: Set-Cookie)
       echo "<strong>$clau:</strong> " . implode(", ", $valor) . "<br>";
       } else {
       echo "<strong>$clau:</strong> $valor<br>";
   }
}
```

Resultats de les Peticions

Els codis d'estat HTTP indiquen el resultat d'una petició al servidor.

Formen part de les capçaleres de resposta i permeten al client saber si la petició s'ha completat correctament, si hi ha hagut un error…

Categories principals de codis d'estat:

- 2xx (Èxit): Indica que la petició s'ha completat amb èxit.
  - 200 OK: Petició completada correctament.
- 3xx (Redireccions): Indica que cal prendre accions addicionals per completar la petició.
  - 301 Moved Permanently: El recurs s'ha mogut de manera permanent.
- 4xx (Errors del client): Indica que hi ha hagut un problema amb la petició enviada pel client.
  - 404 Not Found: El recurs sol·licitat no existeix.
- 5xx (Errors del servidor): Indica que el servidor ha experimentat un error.
  - 500 Internal Server Error: Hi ha hagut un error general al servidor.

Funcions PHP gestió capçaleres HTTP

PHP inclou funcions per treballar amb capçaleres, tant per obtenir-les com per establir-les.

Obtenir capçaleres

La funció ***getallheaders()*** permet accedir a totes les capçaleres enviades en una petició HTTP:

```
<?php
$capçaleres = getallheaders(); // Obtenim les capçaleres
echo "<h3>Totes les capçaleres de la petició:</h3>";
foreach ($capçaleres as $clau => $valor) {
    echo "<strong>$clau:</strong> $valor<br>";
}
```

Establir capçaleres

La funció ***header()*** permet definir capçaleres personalitzades a les respostes HTTP

HTTP i els mètodes GET i POST

Els mètodes HTTP, tant GET i POST, són accions definides pel protocol HTTP per interactuar amb els recursos d'un servidor.

Aquests mètodes dicten el propòsit de la sol·licitud: obtenir informació, enviar dades, modificar recursos, etc.

GET i POST són els dos mètodes més utilitzats en aplicacions web, especialment per:

- Obtenir dades d'un servidor (GET).
- Enviar dades al servidor (POST).

Són casos d'ús típics d'aquests dos mètodes:

- GET: Cerca a Google, carregar pàgines, navegar entre seccions.
- POST: Formularis de registre, enviament de comentaris, pujada de fitxers.

Mètode GET

- Definició:
    - GET s'utilitza per passar informació des de la plana origen o obtenir informació o recursos d'un servidor en la plana destí.
    - Les dades enviades (si n'hi ha) s'inclouen a la URL com a paràmetres de consulta.
    - Exemple: Accedir a una pàgina web o buscar informació.
- Característiques:
    - Les dades són visibles a l'URL (ex.: example.com?page=1).
    - Limitació en la mida de les dades enviades.
    - No es recomana per dades sensibles (ex.: contrasenyes).

Passar informació amb el mètode GET

Amb el mètode GET, en la barra d'adreces del navegador, després de l'adreça que estem sol·licitant, podem veure el nom de les variables que rep la plana i el valor de cada variable.

Exemple 1:

En el següent exemple, es crida a exemple1.php i se li passa la variable var1 amb valor igual a 3 i la variable var2 amb valor igual a 25

http:\\localhost\exemple1.php?var1=3&var2=25

Exemple 2:

Es crida a exemple2.php i se li passa la variable var1 amb valor igual a "Aixo es un string"

http:\\localhost\plana.php?var1=aixo+es+un+string

Passar informació mètode GET

Per implementar amb PHP aquest pas d'informació amb el mètode GET, podem escollir 2 opcions:

- A través d'un enllaç (link) utilitzant l'etiqueta <a href> d'HTML

  Exemple 3: link.php

  ```
  <?php
  $var1="Aixo es un string";
  echo "<a href=\"plana.php?var1=$var1\"> Anar a plana.php </a>";
  ?>
  ```

- A través de formularis

Passar informació mètode GET

Paràmetres de la URL:

- El caràcter ? serveix per separar l'adreça web de les variables
- El caràcter = serveix per indicar el valor de la variable
- El caràcter & serveix per indicar que passem una altre variable
- Els caràcters + o % s'utilitzen per substituir els espais en blanc en variables de tipus string (cal utilitzar les funcions urlencode(...) i urldecode(...))

Observacions sobre el mètode GET:

- És pràctic passar les variables a través d'un enllaç (a href) sense haver d'utilitzar formularis.
- Mètode poc segur, ja que les dades estan a la vista de tothom. No és recomanable passar informació confidencial ni contrasenyes.

Rebre informació mètode GET

$_GET és un array associatiu que conté totes les variables que rep una plana php a través del mètode GET.

El nom de la variable que s'envia és l'índex de l'array associatiu $_GET i el seu valor està emmagatzemat en aquella casella de l'array.

Exemple 4: plana.php

```
<?php
$dada=$_GET[“var1”];
echo “La variable que rebem val: $dada”;
?>
```

Mètode POST

- Definició:
    - POST s'utilitza per enviar dades al servidor.
    - Les dades s'inclouen al cos de la petició, no a l'URL.
- Característiques:
    - Les dades no són visibles a l'URL (més segures que GET).
    - No hi ha límit teòric en la mida de les dades.
    - Ideal per dades sensibles (ex.: contrasenyes, formularis de registre).

Passar informació mètode POST

Per passar informació a través d'aquest mètode és obligatori utilitzar els formularis d'HTML

.Cada element del formulari s'envia en forma de variable.

Exemple 5: formulari_post.php

```
<form name="f1" action="planadesti.php" method=post>
NOM: <input name="nom" type="text">
COGNOM: <input name="cognom" type="text">
EMAIL: <input name="email" type="text">
<input type="submit" value="enviar">
</form>
```

Observacions:

- Enviem a planadesti.php, 3 variables: nom, cognom i email.
- Si canviem POST per GET en la propietat method, estarem enviant les dades via GET i per tant les veurem en la URL del navegador.

Rebre informació mètode POST

Cada element d'un formulari HTML d'una plana origen es pot recollir a la plana destí.

L'array $_POST és un array associatiu que conté totes les dades que s'envien a través del formulari pel mètode POST.

El nom de cada element del formulari que s'envia és un índex de l'array associatiu $_POST i el seu valor està emmagatzemat en aquella casella de l'array.

Exemple 6: planadesti.php

```
<?php
echo "NOM: ".$_POST["nom"];
echo "COGNOM: ".$_POST["cognom"];
echo "EMAIL: ".$_POST["email"];
?>
```

Observació: Rebem 3 variables pel mètode POST: nom, cognom i email

Funcions relacionades POST i GET

Validació de dades amb isset() i empty()

Aquestes funcions són essencials per comprovar si els camps han estat enviats i no estan buits.

```
if (isset($_POST['nom']) && !empty($_POST['nom'])) {
    echo "Hola, " . htmlspecialchars($_POST['nom']) . "!";
} else {
    echo "El camp 'nom' és obligatori.";
}
```

Filtrat i validació de dades amb filter_input()

Funcions relacionades POST i GET

Filtrat i validació de dades amb filter_input()

Permet validar i filtrar dades directament des de $_GET o $_POST.

```
$email = filter_input(INPUT_POST, 'email', FILTER_VALIDATE_EMAIL);
if ($email) {
    echo "El teu correu és: $email";
} else {
    echo "El correu no és vàlid.";
}
```

Funcions relacionades POST i GET

Validació avançada amb preg_match()

Permet validar dades amb expressions regulars, útil per formats específics (e.g., telèfons).

```
$telefon = $_POST['telefon'];
if (preg_match('/^\d{9}$/', $telefon)) {
    echo "Telèfon vàlid: $telefon";
} else {
    echo "Telèfon no vàlid.";
}
```