El llenguatge PHP (III): Entorn web

El propòsit principal del llenguatge PHP és permetre construir aplicacions web. Per tant, la seva interacció amb elements HTML és bàsica. En aquest sentit, l'eina més habitual que permet la introducció de dades per part de l'usuari en un entorn web és el formulari HTML, així que molt sovint els nostres scripts PHP hauran de saber gestionar aquest tipus d'entrada de dades per tal de poder-les processar correctament.

Treball amb formularis HTML

Suposarem que tenim el següent formulari (exclusivament HTML), que consta de diversos elements (caixes de text, checkboxes, botons radio, comboboxes...) que envien les dades (en aquest cas, utilitzant el mètode GET perquè no s'indica cap mètode explícit) a una pàgina PHP anomenada "processar.php" (que desenvoluparem tot seguit)

> NOTA: Recordeu que per a què un formulari enviï les dades mitjançant POST només cal afegir l'atribut HTML *method="post"* a l'etiqueta *<form>*

```
<!DOCTYPE html><html><body>
<form action="processar.php">
Nom: <input type="text" name="nom" value="Jim"><br>
Contrasenya: <input type="password" name="contra" maxlength="10"><br>
Rang d'edat: <select name="edat">
        <option value="Sota 16">Sota 16</option>
        <option value="16-40" selected>De 16 a 40</option>
        <option value="41-99">Per sobre de 40</option>
        </select><br>
Biografia: <br><textarea name="bio" rows="10" cols="80">Escriu la teva biografia aquí</textarea><br>
<input type="radio" name="esport" value="Tenis" checked="checked">Tenis</input><br>
<input type="radio" name="esport" value="Basket">Basket</input><br>
<input type="radio" name="esport" value="Futbol">Futbol</input><br>
<input type="checkbox" name="idiomes[]" value="Anglès" checked="checked">Anglès</input><br>
<input type="checkbox" name="idiomes[]" value="Francès">Francès</input><br>
<input type="checkbox" name="idiomes[]" value="Alemany">Alemany</input><br>
<button type="submit" name="submit">Enviar</button>
</form>
</body></html>
```

> NOTA: Fixeu-vos com el nom dels diversos elements "radio" i "checkbox" és el mateix, respectivament, per establir que tots ells formen part del mateix conjunt d'opcions. Fixeu-vos també que, en el cas concret dels "checkbox", però, el seu nom finalitza amb "[]" perquè així després a la pàgina PHP podrem indicar que els valors que el componen -ja que poden ser varis- formen part d'un array (cosa que no passa amb els elements "radio", que només poden tenir un sol valor).

La idea de la pàgina "processar.php" (com la de qualsevol altra pàgina PHP, de fet), és que faci el processament/càlcul/filtratge que calgui de les dades obtingudes i, en acabar, que eventualment les guardi (bé en fitxer, bé en una base de dades, etc) o les reenviï a un altre destí. Però ara no ens entretindrem en fer cap d'aquestes tasques de processament (perquè podrien ser moltes i molt variades, depèn de cada cas particular) sinó que només ens centrarem en com implementar correctament l'entrada d'informació; és a dir, en saber com recull una pàgina PHP les dades introduïdes per l'usuari en el formulari HTML.

En el cas de què les dades escrites en un formulari s'envïin a la pàgina PHP fent servir el mètode GET (per tant, visibles a la URL en forma de "querystring"), cadascuna d'aquestes dades estarà llavors disponible dins del codi d'aquesta pàgina en forma d'element d'un array associatiu "superglobal" anomenat *$_GET["nomCampFormulari"]* En el cas que s'envïn, però, mitjançant el mètode POST, llavors cadascuna de les dades rebudes per la pàgina PHP estarà disponibles en forma d'element d'un altre array associatiu "superglobal" anomenat *$_POST["nomCampFormulari"]* Com a mostra d'exemple, la següent pàgina (que hem d'anomenar obligatòriament "processar.php" ja que és el que hem indicat a l'atribut *"action"* del formulari HTML) mostra per pantalla tot allò recollit del formulari d'exemple anterior (sempre que se serveixi des d'un servidor web!!).

```
<!DOCTYPE html><html><body><?php
echo "El teu nom és: {$_GET['nom']}<br>";
echo "La teva contrasenya és: {$_GET['contra']}<br>";
echo "La teva edat és: {$_GET['edat']}<br>";
echo "La teva biografia és: {$_GET['bio']}<br>";
echo "El teu esport preferit és: {$_GET['esport']}<br>";
if(empty($_GET['idiomes'])){
	echo "No has triat cap idioma";
} else {
	foreach ($_GET['idiomes'] as $unidioma) {
		echo "$unidioma<br>";
	}
}
?></body></html>
```

NOTA: Noteu com s'ha comprovat amb la funció *empty($nomVariable);* si concretament l'array *$_GET["idiomes"]* està buit -aquesta funció retorna *true* si el valor indicat com a paràmetre és així- (és a dir, que no s'ha triat cap idioma d'entre els disponibles als diferents "checkboxs" del formulari), per tal de mostrar el missatge corresponent. Això es podria haver fet, de fet, amb qualsevol altra dada del formulari per, per exemple, avisar a l'usuari de la obligatorietat d'introduir el valor en qüestió (situació en la qual llavors hauríem de definir la lògica de la nostra pàgina PHP, bé tornant l'usuari al formulari inicial, o a alguna altra pàgina d'error, etc)

NOTA: Noteu també com per recollir els possibles valors sel·leccionats dels diversos checkbox que formen el conjunt (l'array) "idiomes" hem simplement de recòrrer aquest array: els elements existents en ell es correspondran als valors dels checkboxs chequejats. En aquest sentit, una altra possibilitat per tractar els valors rebuts d'un array és "concatenar-los" en forma de cadena per tal de poder treballar-hi més còmodament (és a dir, si *$_GET["idiomes"]* conté per exemple tres valors: *$_GET["idiomes"][0]="Anglès"*, *$_GET["idiomes"][0]="Francès"* i *$_GET["idiomes"][0]="Alemany"*, transformar-los en la cadena "Anglès-Francès-Alemany", per exemple. Això es pot fer amb la funció *implode('x', $unarray);* , on "x" representa el caràcter que es farà servir per delimitar els diferents valors entre sí dins de la cadena (en aquest exemple, seria el guió ("-").

NOTA: D'altra banda, a l'hora de mostrar el text llarg ("la biografia"), podríem haver optat per substituir els possibles salts de línia "de terminal" que hi podrien haverper salts de línia HTML (és a dir, etiquetes *
*), per tal de mostrar aquest text en la pàgina web final d'una forma més entenedora. Això seria tan fàcil com, abans de mostrar el text a la pàgina, afegir la línia de codi següent: *$_GET["bio"]=str_replace("\n","
",$_GET["bio"]);*

NOTA: Noteu finalment com hem indicat el nom de la clau de cada element de l'array $_GET[] entre cometes simples per poder anar alternant amb les cometes dobles que encercla el missatge mostrat per l'echo

Si en provar l'exemple anterior, hi hagués alguna cosa que no funcionés i es volgués tornar a repetir la prova, recordeu sempre abans de netejar la memòria cau del navegador (és una opció que apareix d'entre unes quantes dins de la finestra de "Netejar historial") per tornar a efectivament realitzar una nova petició fresca, ja que si no correm el risc d'observar el resultat prèviament guardat de la versió anterior del codi.

Cas particular: pujada d'arxius

En el cas que un formulari ofereixi la possibilitat de pujar fitxer al servidor web, hem de disposar d'un codi PHP adient per tal de gestionar aquesta pujada, ja que en aquest cas no estem parlant de rebre via GET/POST simples valors de cadena sinó de rebre contingut binari amb un determinat format. Per aconseguir-ho cal fer el següent:

a) Assegurar-se que la directiva ***file_uploads***, present a l'arxiu "**/etc/php.ini**" (a Fedora) o bé "**/etc/php/*X.Y*/fpm/php.ini**" (a Ubuntu), valgui ***On***. Per defecte ja és així però per si de cas, va bé assegurar-se que l'intèrpret PHP permeti pujades d'arxius

NOTA: A més, convé tenir present altres directives relaciones que pot ser que necessitem modificar també (totes elles documentades aquí: https://www.php.net/manual/en/ini.core.php#ini.sect.file-uploads). Per exemple, ***upload_max_filesize nº*** (on "nº" representa el nº de bytes màxim de tamany permès que podran tenir els fitxers a pujar -es poden indicar també els sufixos "K", "M" o "G"- i ha de ser un valor sempre més petit que l'indicat a la directiva ***post_max_size nº***, la qual serveix per indicar el tamany màxim admès de les dades -en general- rebudes en una petició POST) o també **upload_tmp_dir "/ruta/absoluta"** (on "/ruta/absoluta" representa la ruta d'una carpeta on s'ubicaran temporalment els fitxers pujats pel formulari abans que el nostre codi PHP faci quelcom amb ells; sovint indicarem la carpeta "/tmp" -tot i que si no s'indica s'utilitzarà la carpeta temporal definida al sistema- però en tot cas, cal que sigui una carpeta on l'intèrpret PHP -o millor dit, l'usuari amb el què aquest s'executa- tingui permisos d'escriptura)

NOTA: Algunes directives que podem trobar a l'arxiu "php.ini" (les quals es troben totes llistades a https://www.php.net/manual/en/ini.list.php) poden modificar-se "al vol" en temps real dins del propi codi d'un script PHP mitjançant l'ús de la funció *ini_set("nomDirectiva", {valorDesitjat|"valorDesitjat"});* Per saber quines directives sí es poden i quines no, a la columna titulada "Changeable" de la llista anterior ha d'haver el valor "PHP_INI_USER" o "PHP_INI_ALL" (tal com s'explica a https://www.php.net/manual/en/configuration.changes.modes.php)

b) Dissenyar el formulari en qüestió amb l'element "input" que permet a l'usuari triar el fitxer a pujar. Un exemple podria ser el següent:

```
<!DOCTYPE html><html><body>
<form action="processar.php" method="post" enctype="multipart/form-data">
Tria fitxer a pujar: <input type="file" name="fitxer"><br>
<button type="submit" name="submit">Pujar</button>
</form>
<html><body>
```

NOTA: En aquest cas és important que el mètode utilitzat sigui POST perquè GET no permet la transferència de contingut binari. És per això que hem afegit l'atribut *"method=post"* a l'etiqueta *<form>*
NOTA: Un altre atribut que hem hagut d'indicar també en l'etiqueta *<form> és enctype="multipart/form-data"* Aquest atribut, que és obligatori quan es vol que el formulari pugui pujar fitxers; indica el valor de la capçalera "Content-Type" de la petició POST; concretament "**multipart/form-data**" és el format que ha de tenir aquesta petició si es vol que el contingut binari a pujar es transmeti correctament. L'altre valor permès per l'atribut *enctype* (i que és el valor per defecte, així que, per tant, no se sol indicar explícitament) és "**application/x-www-form-urlencoded**" el qual, com diu el seu nom, simplement consisteix simplement en una tira de caràcters, tot i que transformats al format URL-Encoded (és a dir, on els espais es converteixen en "+", determinats símbols especials es converteixen als seus valors hexadecimals corresponents, etc); aquest és el format amb el qual les dades introduïdes als formularis (excepte els fitxers) s'envien.

c) Desenvolupar el codi PHP que rebrà el fitxer, que l'haurà de gestionar d'alguna manera (normalment, emmagatzemant-lo en algun destí concret) i que, preferiblement, informi a l'usuari de si la pujada ha sigut exitosa o no. Concretament, per poder treballar amb l'element "fitxer" des del codi PHP un cop rebut aquest des del formulari s'ha d'utilitzar el "superglobal" *$_FILES["nomCampFormulari"];*, el qual és un array on cada element (que representa cadascun dels fitxers pujats mitjançant cadascun dels elements "file" que eventualment puguin haver al formulari), és al seu torn un array associatiu format per, entre altres, els següents elements que descriuen cada fitxer concret en qüestió (com es pot veure si es fa que "processar.php" tingui el codi PHP següent):

"*name*" : nom del fitxer pujat
"*type*" : tipus MIME del fitxer pujat (d'aquesta dada n'informa el navegador)
"*tmp_name*": ruta absoluta completa (incloent el nom) del fitxer temporal al servidor que es correspon al fitxer pujat abans de què el codi PHP decideixi què en vol fer
"*size*" : mida del fitxer pujat (en bytes)

```
<!DOCTYPE html><html><body><?php
print_r($_FILES["fitxer"]);
?><html><body>
```

A continuació es mostra un codi d'exemple més elaborat que realitza totes aquestes accions esmentades abans. A destacar l'ús de les funcions PHP relacionades amb la gestió de fitxers, com *file_exists("/ruta/fitxer");*, que retorna *true* si el fitxer (o directori!!) indicat com a paràmetre existeix (i *false* si no) o, sobre tot, *move_uploaded_file("/ruta/fitxer/pujat","/ruta/fitxer/definitiu");*, funció específicament dissenyada per moure el fitxer recentment pujat pel formulari (normalment allotjat en una memòria temporal) a la seva ubicació (i nom) definitiu (i que retornarà *false* si ha ocorregut algun error).

NOTA IMPORTANT: Per a què el codi següent funcioni correctament, cal que existeixi prèviament la carpeta "/var/www/html/uploads" (suposant que "/var/www/html" sigui el DocumentRoot) i que aquesta carpeta tingui com a propietari/grup el propi usuari de l'Apache. És a dir, cal haver fet *sudo mkdir /var/www/html/uploads* i *sudo chown -R apache:apache /var/www/html/uploads* (a Fedora, o *sudo chown -R www-data:www-data /var/www/html/uploads* (a Ubuntu). A més, caldrà que el subsistema SELinux (present a Fedora només) estigui establert en mode "permissiu" (això es pot fer, per l'arranc actual, executant la comanda *sudo setenforce permissive* o bé, per tots els arrancs, establint la línia *SELINUX=permissive* dins de l'arxiu "/etc/selinux/config".

```
<!DOCTYPE html><html><body><?php
$target_file = "uploads/" . $_FILES["fitxer"]["name"];
$uploadOk = 1;
//Si no s'arriba a aquesta pàgina des del formulari anterior, no té cap sentit interpretar el seu codi
if(isset($_POST["submit"])) {
	//Faig algunes comprovacions sobre el fitxer pujat per decidir si finalment la pujada es farà correctament
	if (file_exists($target_file)) {
		echo "El fitxer indicat ja existeix (segurament s'hagi pujat anteriorment).<br>";
		$uploadOk = 0;
	}
	if ($_FILES["fitxer"]["size"] > 500000) {
		echo "El fitxer indicat és massa gran.<br>";
		$uploadOk = 0;
	}
	/*Una altra opció (no excloent) d'assegurar-se que el fitxer pujat és una fotografia (de qualsevol format)
	és emprar la funció getimagesize() per veure si no retorna false*/
	if ($_FILES["fitxer"]["type"] != "image/jpeg" && $_FILES["fitxer"]["type"] != "image/png") {
		echo "El fitxer indicat no és ni JPG, ni PNG (que són els únics tipus acceptats)";
		$uploadOk = 0;
	}
	//Un cop fetes les comprovacions, procedeixo a efectuar la pujada del fitxer (o no, segons el resultat d'aquestes
	if ($uploadOk == 0) {
		echo "ERROR: no s'ha pogut pujar el fitxer.<br>";
	}  else {
		if (move_uploaded_file($_FILES["fitxer"]["tmp_name"], $target_file)) {
			echo "El fitxer ". $_FILES["fitxer"]["name"] ." ha sigut guardat correctament al servidor";
		} else {
			echo "ERROR: no s'ha pogut guardar el fitxer indicat al seu destí permanent";
		}
	}
}
?></body></html>
```

Validació d'entrades

Un aspecte molt important de la gestió de formularis és controlar que les dades introduïdes per l'usuari siguin correctes. Penseu que un formulari és una porta d'entrada a valors que no controlem i, per tant, hem de poder rebutjar totes aquestes dades que no s'ajustin al tipus de valor esperat, tant per coherència de tipus en el processament que en poguem fer d'elles en els nostres scripts PHP (incloent eventuals emmagatzematges en bases de dades!) com també per seguretat (evitant possible entrada de cadenes malicioses al servidor). Per tant, per exemple, si volem que l'usuari escrigui una adreça de correu, hem d'assegurar-nos que si no escriu correctament una cadena amb una forma com "xxx@xxx.xx", aquesta dada sigui rebutjada; si volem que l'usuari escrigui la seva edat, hem d'assegurar-nos que indiqui un nombre sencer dins d'un cert rang, etc, etc.

Per aconseguir aquest control es pot tirar fer-ho des del costat del navegador (amb Javascript; seria la forma preferible perquè les dades no arribarien a sortir del client si fossin incorrectes) o bé fer-ho des del costat del servidor (un cop les dades introduïdes per l'usuari hi han arribat; en tot cas no està de més tenir un "segon factor de comprovació"). Per realitzar la validació de dades des del servidor (és a dir, la comprovació que aquestes dades estan en el format correcte) així com també la seva "sanitització" (és a dir, l'eliminació dels caràcters que poguessin ser "il·legals" per aquell format), PHP ofereix una funció específica anomenada *filter_var($nomVariable, NOM_FILTRE);*, on "NOM_FILTRE" serà algun dels valors dels elements de l'array retornat per la funció *filter_list();* i representa el tipus de "format" contra el qual es vol validar el valor de la variable indicada, per comprovar així si concorda (retornant *filter_var()* llavors el valor *true*) o no (retornant llavors el valor *false*). D'entre els filtres que podem indicar podem destacar *FILTER_VALIDATE_INT, FILTER_VALIDATE_FLOAT, FILTER_VALIDATE_BOOL, FILTER_VALIDATE_MAIL, FILTER_VALIDATE_URL, FILTER_VALIDATE_DOMAIN, FILTER_VALIDATE_IP, FILTER_VALIDATE_MAC*, entre altres (es pot consultar la llista completa a https://www.php.net/manual/en/filter.filters.php).

NOTA: Aquesta funció no només es pot utilitzar per dades obtingudes des d'un formulari sinó des de qualsevol altre origen de dades extern, com ara el resultat d'una consulta a una base de dades, d'una "cookie", etc
NOTA: Existeixen altres funcions més senzilles que simplement retornen un valor booleà (true o false) segons si el valor indicat com a parèntesi es detecta que és un número sencer (*is_int($nomVariable);*) o decimal (*is_float($nomVariable);*) o numèric en general (*is_numeric($nomVariable);*) o un valor booleà (*is_bool($nomVariable);*) o de cadena (*is_string($nomVariable);*) o un array (*is_array($nomVariable);*) o un objecte (*is_object($nomVariable);*), etc. En aquest sentit, també existeixen les funcions pròpies de la llibreria CTYPE (https://www.php.net/manual/en/book.ctype.php), les quals fan les comprovacions a nivell de caràcters

Ja hem dit que la funció *filter_var()* no només serveix per validar sinó també per "sanititzar" les dades d'entrada. Això ho pot fer amb el filtre corresponent. Examples d'aquest tipus de filtres són *FILTER_SANITIZE_NUMBER_INT* (que elimina tots els caràcters de la dada que no siguin dígits, "+" o "-"), *FILTER_SANITIZE_NUMBER_FLOAT* (que elimina tots els caràcters de la dada que no siguin dígits, "+", "-", ".", "," ,"e" o "E"), *FILTER_SANITIZE_EMAIL* (que elimina tots els caràcters que no siguin legals en una adreça de correu), *FILTER_SANITIZE_URL* (que elimina tots els caràcters que no siguin legals en una URL)...

NOTA: Altres funcions que també serveixen per "sanititzar" són *htmlspecialchars($nomVariable);* la qual "escapa" certs caràcters emprats a l'HTML (com "<" o ">" o "&") per tal d'evitar possibles atacs XSS si la variable en qüestió fos intepretada com a contingut HTML (d'altres funcions similars però complementàries són *htmlentities($nomVariable);* o *strip_tags($nomVariable);*) o també *trim($nomVariable);* la qual elimina els espais en blanc, tabuladors i noves línies del davant i del darrera del valor de la variable en qüestió, etc

Per exemple, el codi següent comprova si l'entrada a avaluar es correspon a un número sencer; si no és el cas (fixeu-vos en l'ús del símbol "!" per negar la condició), el "sanititza" i al final, el mostra a pantalla.

```
<!DOCTYPE html><html><body><?php
$num="aa46bb";
if(!filter_var($num,FILTER_VALIDATE_INT)){
    echo("El valor $num no és un nombre sencer. Es procedirà a sanititzar-l>
    $nounum=filter_var($num, FILTER_SANITIZE_NUMBER_INT);
}
echo $nounum;
?></body></html>
```

NOTA: Existeixen usos més avançat de la funció *filter_var()*, on s'involucra un tercer paràmetre que aquí no hem esmentat, i que es pot observar en els següents exemples: https://www.w3schools.com/php/php_filter_advanced.asp
NOTA: A https://www.w3schools.com/php/php_ref_filter.asp es troben llistades més funcions relacionades amb el filtres

---

Si els filtres predefinits descrits als paràgrafs anteriors no fossin el suficient detallats per detectar el patró d'alguna dada d'entrada que rebem, sempre podem aprofitar-nos de la capacitat de PHP per treballar amb expressions regulars. Concretament, *preg_match("/exprReg/xxx", $nomVariable);* és la funció que farem servir més, ja que serveix per comparar el valor (de tipus cadena) de la variable indicada al segon paràmetre (també podria ser una cadena directament) amb el patró indicat (entre "/") al primer paràmetre: si hi ha concordància, retorna *true* i si no, *false*; els "xxx" representent els modificadors que s'hi poden afegir, opcionalment, a l'expressió regular indicada (com per exemple, "*i*" (per fer la comparació de forma "case-insensitive" -per defecte és "case-sensitive!"-), "*m*" (per fer recerques a múltiples línies) o "*u*" (per indicar que l'expressió regular està escrita fent servir codificació Unicode), etc.

NOTA: Sobre el símbols concrets admesos en les expressions regulars, i el seu significat podeu consultar els documents addicionals accessoris proporcionats pel professor o bé https://www.php.net/manual/en/reference.pcre.pattern.syntax.php o també https://www.w3schools.com/php/php_ref_regex.asp i també https://www.regular-expressions.info/quickstart.html i http://www.hackingwithphp.com/4/8/6/regular-expression-syntax-examples
NOTA: Existeixen possibilitats avançades en relació a la recerca de patrons amb expressions regulars que no desenvoluparem aquí, però que les mencionarem per si fos d'interès; concretament, el concepte de comportament "greedy" i "lazy" a l'hora de buscar concordàncies (https://phpenthusiast.com/blog/php-regular-expressions#lazy-greedy-expressions) o la capacitat de capturar grups ($1,$2...) per treballar-hi específicament (https://phpenthusiast.com/blog/php-regular-expressions#capturing-groups)

Una altra funció també comuna i similar és *preg_match_all("/exprReg/xxx", $nomVariable);* , la qual retorna en aquest cas el nombre de vegades que alguna subcadena concordant amb l'expressió regular indicada s'ha trobat dins del valor (de tipus cadena) de la variable indicada (o d'una cadena directament). Opcionalment aquesta funció pot tenir un tercer paràmetre que serà el nom d'un array d'arrays que emmagatzemarà en el seu primer element (un array) totes aquestes subcadenes concordants que s'hagin trobat, en ordre.

Finalment, una altra funció interessant és *preg_replace("/exprReg/xxx", "subcadena", $nomVariable);* , la qual remplaça amb la subcadena indicada com a segon paràmetre totes les ocurrències de l'expressió regular indicada com a primer paràmtre dins del valor de la variable (o cadena) indicada com a tercer paràmetre.

NOTA: A http://www.hackingwithphp.com/4/8/5/regular-expression-replacements es detallen més casos d'ús i possibilitats que ofereix la funció *preg_replace()*
NOTA: A https://www.php.net/manual/en/ref.pcre.php es troben llistades més funcions relacionades amb la gestió de les expressions regulars. En podem destacar, per exemple *preg_split("/exprReg/xxx",$nomVariable);* , (que retorna un array els elements dels quals són cadascun dels trossos de la cadena que representa el valor de la variable indicada com a segon paràmetre que han coincidit amb l'expressió regular indicada com a primer paràmetre), o *preg_grep("/exprReg/xxx", $nomArray);* (que retorna un array, els elements dels quals són cadascun dels elements de l'array indicat com a segon paràmetre el valor dels quals ha coincidit amb l'expressió regular indicada com a primer paràmetre), entre altres.
NOTA: Teniu més exemples d'ús d'expressions regulars en PHP a https://diego.com.es/expresiones-regulares-en-php

Per exemple, el codi següent, cridat per exemple així: *curl http://ip.serv.web/script.php?unnom=Pepe* (on "script.php" representa el nom del fitxer on està aquest codi escrit i "ip.serv.web" és la IP del servidor web on s'executa l'intèrpret PHP) retornarà "Hola Pepe" però en canvi cridat així *curl http://ip.serv.web/script.php?unnom=$%·/* retornarà el missatge d'error :

```
<!DOCTYPE html><html><body><?php
$unnom = $_GET["unnom"]);
if (!preg_match("/^[a-zA-Z]*$/",$unnom)) {
  die("Només s'admeten lletres <br>";)
}
echo("Hola $unnom <br>");
?></body></html>
```

Un altre exemple és el següent, on, un cop executat el codi (des del terminal mateix), apareixeran quatre concordàncies però si canviem l'expressió regular existent per aquesta altra: "*/ain/*", llavors n'apareixeran només tres, mentre que si la canviem per "*/ain\b/i*" llavors n'apareixeran només dues i si la canviem per "*/ain\b/*" només n'apareixerà una.

```
<?php
$num=preg_match_all("/ain/i", "The rain in SPAIN falls mainly on the plains", $results);
echo $num;
print_r($results[0]);
?>
```

Cookies i sessions

Una "cookie" és un petit fitxer de text que un servidor PHP pot enviar al client (per a què aquest el guardi en un espai específic al seu disc dur) el primer cop que aquest el visiti. Els següents cops, serà llavors el client qui enviarà la "cookie" en el seu poder al servidor, per a què aquest pugui llegir el seu contingut i actuar en conseqüència (cosa que inclou, per exemple, tornar a reenviar la "cookie" de nou al client amb un valor actualitzat, per exemple). Per tant, és fàcil veure que les "cookies" són un mecanisme que permet desar informació en la màquina client per tal que estigui disponible pel servidor en algun moment de més endavant. Així, mitjançant "cookies" es pot desar certa configuració personal de l'usuari, com, per exemple, la llengua en què ha escollit visitar el web (o qualsevol altra dada personalitzada del client): la propera vegada que l'usuari visiti el servidor PHP, si hem escrit el codi corresponent, podrem obtenir aquesta informació desada i presentar a l'usuari el web amb la seva llengua escollida directament.

NOTA: Tot i que les "cookies" queden desades a la màquina client i no al servidor web, és molt difícil utilitzar-les per fins malintencionats ja que estan dissenyades amb moltes limitacions precisament per aquesta raó de seguretat; concretament, tenen una mida màxima d'1 kilobyte (per tant, no poden tenir un contingut arbitràriament gran) i, sobre tot, caduquen al cap d'un cert temps definible pel servidor web (i per tant, fent-les inútils passat aquest temps). A més, només les pot consultar el servidor web que l'ha dipositada (així doncs, la informació que conté no és visible per la resta de pàgines web).

L'exemple següent, dividit en dues parts, mostra l'ús bàsic d'una "cookie": la primera part crea la "cookie" en sí amb la funció específica ***setcookie()***, donant-li un nom (en aquest cas, "Usuari"), assignant-li un valor concret (en aquest cas, "Pepito"), establint-hi una data de caducitat (en segons...en aquest cas s'ha indicat un mes a partir del moment en què es crea -86400s és un dia-) i llavors depositant-la remotament a qualsevol client que hi accedeixi a la pàgina PHP en qüestió (per això és molt important que la funció *setcookie()* estigui escrita sempre abans d'enviar l'etiqueta *<!DOCTYPE html>*...si s'escriu després, un cop ja s'hagi començat a enviar contingut HTML, ja haurà sigut massa tard per poder enviar la "cookie"); la segona recupera del client el valor emmagatzemat en aquesta "cookie" (gràcies a l'ús de l'array superglobal ***$_COOKIE["nomCookie"]***, que serveix precisament per llegir el valor de la "cookie" indicada, la qual haurà sigut rebuda automàticament des del navegador en haver sigut depositada en alguna visita anterior al mateix lloc web -i si no s'ha acomplert la data de caducitat-) i mostra aquest valor a pantalla. L'efecte de tot plegat és que la primera vegada que es visiti amb el navegador la pàgina amb el codi de l'exemple es veurà el missatge "1ª vegada que..." i, a partir de llavors, la segona, tercera, quarta...vegada que es torni a visitar, ja només es veurà la frase "Cookie 'Usuari' rebuda...".

```
<?php
setcookie("Usuari", "Pepito", time() + (86400 * 30));
?>
<!DOCTYPE html><html><body><?php
if(!isset($_COOKIE["Usuari"])) {
    echo("1ª vegada que has visitat aquesta web (no s'ha rebut la cookie 'Usuari' del navegador");
} else {
    echo("Cookie 'Usuari' rebuda, amb valor ". $_COOKIE["Usuari"]);
}
?></body></html>
```

NOTA: L'array $_COOKIE no s'interpreta correctament si s'indica dins d'una cadena; és per això que l'hem hagut de concatenar amb el símbol "." al darrer echo de l'exemple anterior per mostrar el seu valor

NOTA: En teoria es pot modificar el valor de qualsevol element de l'array $_COOKIE però aquest canvi només existirà mentre s'executi el codi PHP en qüestió però no es traslladarà a la "cookie" emmagatzemada al client: per modificar una "cookie" al client cal fer servir la funció *setcookie()* igual que es faria en el cas de si fos una "cookie" nova

NOTA: La funció *setcookie()* retornarà cert si la galeta s'ha creat sense cap dificultat i fals si hi ha hagut algun problema. Els seus únics paràmetres obligatoris són el nom i el valor inicial que se li vol assignar; si no s'indica data de caducitat, la galeta deixarà d'existir tan bon punt finalitzi la sessió d'usuari actual (això, a la pràctica, vol dir quan es tanqui el navegador sencer). D'altra banda, s'hi poden afegir més paràmetres:

- *4rt paràmetre: indica la ruta de la URL (és a dir, la part que ve després del nom de domini) a partir de la qual la "cookie" es tindrà en compte. Per defecte és "/", que vol dir que la "cookie" serà reconeguda per totes les rutes
- *5è paràmetre: indica el domini en què la "cookie" tindrà validesa (és a dir, on podrà ser llegida i manipulada). Per defecte és el corresponent al "virtualhost" del lloc web que l'ha emès per primer cop
- *6è paràmetre: valor booleà que indica si volem que la "cookie" s'utilitzi només en una connexió HTTPS (si val *true*) o també en una connexió HTTP (si val *false*, per defecte)

NOTA: La funció *setcookie()* en realitat el que fa és implementar la capçalera HTTP "Set-cookie" en la resposta del servidor (https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie) . És per això que cal indicar-la abans que s'envïi qualsevol tipus de cos, a l'igual que passa amb la funció, més genèrica, *header("nomCapçalera:valor");*, que serveix per establir el valor de qualsevol capçalera de resposta HTTP (per més informació, veure https://www.w3schools.com/php/func_network_header.asp)

Si en provar l'exemple anterior, es volgués tornar a l'estat inicial per torna a començar com quan el navegador no disposava de la "cookie", caldrà esborrar-la del magatzem de "cookies" gestionat pel client emprat. En el cas, per exemple, de usar el navegador Firefox, això es fa anant a "*about:preferences#privacy*" i clicant sobre el botó "Gestiona les dades": al quadre que hi apareix s'hi podrà buscar la "cookie" concreta que es vol eliminar. També ho podeu fer, de forma més tècnica, anant al menú "Més eines->Web developer tools" i allí, a l'apartat "Galetes" de la pestanya "Emmagatzematge". En tot cas, a https://support.mozilla.org/ca/kb/storage?as=u&utm_source=inproduct&redirectslug=permission-store-data teniu més informació.

NOTA: Un altre "truc" per eliminar una "cookie", que en aquest cas es pot fer des del propi servidor sense haver d'accedir directament al client és establir-la de nou amb la funció *setcookie()* però aquest cop amb una data de caducitat anterior a la data actual

Tal com s'ha comentat en una nota anterior, per modificar el valor d'una "cookie" ja enviada al client, s'ha de fer servir de nou la funció *setcookie()* indicant el mateix nom de "cookie" i el nou valor que li volem assignar; en tot cas, recordeu que aquesta manipulació s'haurà de realitzar abans que s'envïi al client qualsevol codi HTML.

NOTA: Si es vol treballar amb "cookies" amb la comanda *curl* en lloc d'un navegador, a continuació s'indiquen els paràmetres adients:
-c /ruta/fitxer.txt : Guarda les cookies rebudes del servidor amb qui s'ha contactat
-b /ruta/fitxer.txt : Envia el fitxer indicat (que contindrà les "cookies" guardades amb el paràmetre *-c* d'alguna petició anterior) al servidor que s'hagi indicat
-b "nomCookie=valor" : Envia una cookie concreta amb un valor concret al servidor que s'hagi indicat

---

Les sessions us permeten emmagatzemar dades que podran ser compartides entre les diferents pàgines del web. Sense aquest mecanisme, cada petició HTTP a una pàgina web és tornar a començar de nou (ja que el protocol HTTP "no té memòria"); les sessions permeten tenir una navegació "amb estat" entre diferents pàgines web dins d'un determinat "context" comú. Alguns exemples de dades que solen guardar-se a les sessions és, per exemple, saber si l'usuari ha accedit al web mitjançant uns determinats valors d'usuari/contrasenya, o el manteniment d'un carret de la compra; en general, seran casos on tindrem pàgines PHP que rebran inicialment les dades d'un formulari però que no tindran per què mostrar-les a l'usuari immediatament sinó que, després de processar-les de la manera que sigui, les reenviaran a una altra pàgina pertanyent a la mateixa sessió, la qual potser necessitarà passar al seu torn aquestes dades a una tercera pàgina de la mateixa sessió, etc, etc, segons vagi essent la navegació de l'usuari.

Les dades guardades en una sessió, a diferència de les cookies, estan emmagatzemades de forma centralitzada al servidor PHP i estan disponibles durant tot el temps que l'usuari estigui interactuant amb la pàgina (l'anomenat temps de sessió) fins que o bé es tanqui el navegador o es destrueixi explícitament la sessió (tal com de seguida veurem).

NOTA: Concretament, per defecte les sessions s'emmagatzemen a la carpeta "/tmp" del servidor (la ruta concreta vindrà indicada per la directiva de configuració *session.save_path* dins de l'arxiu "php.ini") però aquesta ubicació es pot canviar per qualsevol altra mitjançant la directiva *session.save_handler* (per exemple, un SGBD si es vol que sigui una sessió de tipus permanent); mireu https://www.php.net/manual/en/session.customhandler.php per més informació i també l'article següent: http://www.hackingwithphp.com/10/3/7/files-vs-databases-session_set_save_handler

A continuació mostrarem l'ús de sessions fent servir quatre pàgines PHP d'exemple: "1.php", "2.php", "3.php" i "4.php". Comencem; concretament, a "1.php" (el codi següent), és on iniciem la sessió, gràcies l'execució de la funció ***session_start();***, funció que ha de ser escrit al principi de tot el codi, abans de qualsevol etiqueta HTML. A partir de llavors, podrem establir qualsevol variable com a "variable de sessió" simplement afegint-la com element de l'array superglobal ***$_SESSION["nomVariable"]***; :

```
<?php
session_start();
?>
<!DOCTYPE html><html><body><?php
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo("Session variables are set. <br> Go to <a href='2.php'>second page</a>");
?></body></html>
```

NOTA: Al codi anterior s'han assignat valors de cadena a les variables de sessió, però aquestes poden contenir qualsevol altre tipus de valor, fins i tot complexes com arrays o objectes sencers. D'altra banda, a qualsevol variable de sessió pot aplicar-se les funcions *isset()* o *empty()* ja conegudes (i més), com qualsevol altra variable PHP.

La gràcia d'establir variables de sessió és que qualsevol d'elles estarà disponible a qualsevol altra pàgina PHP que pertanyi a la mateixa sessió on es va definir inicialment simplement accedint a l'element adient de array superglobal *$_SESSION[]*, tal com veiem al següent codi ("2.php").

```
<?php  session_start();  ?>
<!DOCTYPE html><html><body><?php
//Mostro el valor de les variables que es van establir a la pàgina anterior
echo("Favorite color is " . $_SESSION["favcolor"] . ".<br>");
echo("Favorite animal is " . $_SESSION["favanimal"] . ".<br>");
echo("Go to <a href='3.php'>third page</a>");
?></body></html>
```

Noteu com al codi anterior també hem fet servir la funció *session_start()*; però ara no per iniciar cap sessió nova sinó per continuar la que es va iniciar a la pàgina anterior. (aquesta funció serveix per les dues coses). ¿I com sap aquesta funció si ha de començar una sessió nova o continuar-ne una altra? Doncs dependrà de la pàgina on està escrita aquesta funció *session_start()* si rep (o no) l'"identificador de sessió començada", identificador que pot rebre's com a valor d'una "cookie" generada a la pàgina inicial i enviada al client (quan es va detectar precisament que no n'hi havia cap, d'identificador establert) o bé a través d'un paràmetre específic dins de la "querystring"

> NOTA: La "cookie" que conté l'identificador de sessió s'anomenarà per defecte "PHPSESSID", nom que correspon al valor per defecte de la directiva *session.name* de l'arxiu "php.ini". Igualment, el nom del paràmetre dins de la URL amb l'identificador de la sessió, si es fa servir en lloc del mecanisme de "cookies", també s'anomenarà "PHPSESSID". En tot cas, es pot consultar (i modificar si s'indiquen els paràmetres adients!) mitjançant la funció *session_name();*

És perfectament possible alterar el valor de qualsevol variable de sessió durant l'existència d'aquesta. Això és el que demostra el següent codi, pertanyent a la pàgina "3.php", on es veu que això és tan senzill com assignar el nou valor desitjat a l'element de l'array *$_SESSION[]* que ens interessi modificar, i ja està.

```
<?php  session_start();  ?>
<!DOCTYPE html><html><body><?php
//Canvio el valor de les variables de sessió i mostro el seu nou valor
$_SESSION["favcolor"] = "red";
$_SESSION["favanimal"] = "dog";
echo("Favorite color now is " . $_SESSION["favcolor"] . ".<br>");
echo("Favorite animal now is " . $_SESSION["favanimal"] . ".<br>");
echo("Go to <a href='4.php'>fourth page</a>");
?></body></html>
```

Finalment, al codi següent ("4.php") veiem com es destrueixen explícitament totes les variables de sessió i, seguidament, com es destrueix la sessió en sí (alliberant, per tant, els recursos emprats al servidor):

```
<?php  session_start();  ?>
<!DOCTYPE html><html><body><?php
//Esborro totes les variables de sessió. També ho podria haver fet així: $_SESSION = array();
session_unset();  //Per esborrar només una variable de sessió, es pot fer unset($_SESSION["nomVar"]);
//Destrueixo la sessió
session_destroy();
echo("Go to <a href='1.php'>beginning</a>");
?></body></html>
```

> NOTA: Tota sessió es destrueix automàticament quan l'usuari tanca el navegador però tot i així, és recomenable gestionar-ho de forma explícita per controlar millor aquest aspecte
>
> NOTA: En principi no seria necessari, però abans de destruir la sessió també es pot esborrar la "cookie" que emmagatzema l'identificador de la sessió en qüestió al sistema client. Això es pot fer escrivint, abans de *session_destroy();* el següent codi: *if(isset($_COOKIE[session_name()])) { setcookie(session_name(), "", time() - 42000); }*
>
> NOTA: Una altra funció interessant de gestió de les sessions és *session_status();* , funció sense paràmetres que retorna l'estat de la sessió (un dels valors PHP_SESSION_ACTIVE, PHP_SESSION_NONE o PHP_SESSION_DISABLED), entre altres. En tot cas, teniu més informació sobre la gestió de sessions a https://www.php.net/manuaen/book.session.php

EXERCICIS:

1.-a) Desenvolupa un formulari web que contingui els següents elements (no importa l'estètica ni la temàtica):

*Una caixa de text buida
*Un "textarea" buit
*Un desplegable que mostri tres opcions (valors de cadena) diferents a triar, amb una d'elles ja triada
*Un conjunt de tres botons radio, amb un d'ells inicialment seleccionat ja
*Un conjunt de tres "checkboxs", cap d'ells inicialment seleccionat
*Un botó d'enviar que faci servir el mètode POST

b) Desenvolupa un script PHP que reculli les dades del formulari anterior i que, a partir d'aquí:

*Que comprovi (amb *isset(), empty()*...com vulguis) que s'hi hagi introduït algun valor a la caixa de text i també al textarea. Si no fos així, caldria mostrar un missatge d'error a l'usuari i finalitzar l'execució de l'script (amb *die()*)

NOTA: En lloc d'usar *die()*, una alternativa seria que es creés una excepció; d'aquesta forma que el codi que la capturés mostraria igualment el missatge d'error però es donaria la possibilitat de seguir executant la resta de codi següent. En tot cas, recorda que tots els missatges d'error generats per l'intèrpret PHP-FPM els podràs veure dins del fitxer "/var/log/php-fpm/www-error.log"

*Que comprovi (amb *filter_var(), is_int()*...com vulguis) que el valor introduït a la caixa de text sigui un nombre sencer. Si no fos així, caldria mostrar un missatge d'error a l'usuari i finalitzar l'execució de l'script

*Que comprovi (amb *empty(), count()*...com vulguis) que s'ha triat com a mínim un "checkbox". Si no fos així, caldria mostrar un missatge d'error a l'usuari i finalitzar l'execució de l'script

*Que, un cop feta totes les comprovacions anteriors, mostri a pantalla tots els valors recollits del formulari anterior

NOTA: Enlloc d'haver d'escriure per una banda el codi HTML i per l'altra el codi PHP, és possible escriure-ho tot plegat en un únic fitxer (és a dir, incloure el codi PHP que processarà les dades introduïdes en un formulari HTML dins del mateix fitxer que defineix el propi formulari en sí). L'únic que cal fer per això és tenir en compte que el valor de l'atribut *action=* de l'etiqueta *<form>* apunti a ell mateix. La manera més flexible de fer-ho, per no haver d'escriure explícitament el nom del propi fitxer (el qual podria canviar) és indicar el valor predefinit **$_SERVER["PHP_SELF"]**, que justament retorna el nom que tingui en aquell moment el fitxer on s'està executant l'script en qüestió. But as this returns the existing filename from the URL, you must be a little careful because users may inject some unwanted code from the URL, so to avoid it, we can use the *htmlspecialchars()* function to convert any special character in the string (URL in this case) into HTML entities, like this: *<form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="POST">*

2.-a) Si ens trobéssim amb el següent codi PHP de validació, el qual recull un valor escrit en una hipotètica caixa de text anomenada "pepe", ¿per a què creus que serviria? (es fan servir les funcions predefinides *checkdate()* i *strtotime()* , a més de *explode()*, de PHP)

```
$dt = $_POST["pepe"];
$dtarray = explode("/",$dt);
$day = $dtarray[0];
$month = $dtarray[1];
$year = $dtarray[2];
if(!checkdate($month, $day, $year))  {
    die("Must be in d/m/y format");
}  else  {
    if(strtotime($dt)<strtotime("now"))  {
        echo("Date supplied is before present day");
    }
}
```

b) Si ens trobéssim amb el funció PHP, hipotèticament pensada per fer-se servir en un codi de validació de dades introduïdes en un formulari web, i més específicament en un conjunt de "checkbox", ¿per a què creus que serviria?

```
function IsChecked($chkname,$value)  {
   if(!empty($_POST[$chkname]))  {
        foreach($_POST[$chkname] as $chkval)  {
             if($chkval == $value) {   return true;   }
        }
   }
   return false;
}
```

3.-a) Crea una pàgina HTML pura anomenada "gracies.html" amb un contingut com aquest:
<DOCTYPE html><html><body>Gràcies</body></html>

> NOTA IMPORTANT: Per a què el codi PHP demanat al següent apartat funcioni correctament, cal que la carpeta on es creï l'arxiu demanat (anomenat "dadesrecollides.csv"), la qual, en aquest cas, serà "/var/www/html", tingui els permisos adients per a què l'usuari amb què s'està executant l'intèrpret PHP-FPM (que és l'usuari de l'Apache, anomenat "apache" a Fedora i "www-data" a Ubuntu) hi pugui escriure. El més fàcil, doncs, serà fer *sudo chmod -R o+w /var/www/html* (tot i que el més raonable seria que la carpeta on s'escriguís qualsevol cosa fos una diferent de la del "DocumentRoot"). Cal executar aquesta comanda o si no el proper apartat no funcionarà A més, caldrà fer una altra cosa si estem executant l'intèrpret PHP-FPM en un sistema Fedora, que és establir el subsistema SELinux (present només a Fedora) en mode "permissiu"; això es pot fer, per l'arranc actual, simplement executant la comanda *sudo setenforce permissive* o bé, per tots els arrancs, establint la línia *SELINUX=permissive* dins de l'arxiu "/etc/selinux/config".

b) Manté el mateix formulari que el de l'exercici 1 però ara canvia el codi PHP que processa les dades escrites per l'usuari en ell per tal que ara, després de fer totes les comprovacions indicades a l'exercici 1, en lloc de simplement mostrar les dades a pantalla, les guardi en un fitxer i, tot seguit, que automàticament redireccioni l'usuari a la pàgina "gracies.html" (és a dir, que implementi la funcionalitat suggerida en el següent codi d'exemple, que hauràs d'adaptar segons el teu exercici concret). Comprova finalment que, efectivament, les dades introduïdes en el formulari apareixeran escrites a l'arxiu indicat (ubicat a la mateixa carpeta on es trobi el propi script):

```
<?php
 $fs = fopen("dadesrecollides.csv","a");  //El fitxer s'ha d'obrir en mode "append" per no sobreescriure contingut
 fwrite($fs,$_POST["unavariable"] . ", " . $_POST["unaaltra"] . "," .  "ETC" . "\n");
 fclose($fs);
 header("Location: gracies.html");
?>
```

> NOTA: Fes servir el "truc" de la funció ***implode()*** indicat en una "nota" de la teoria per tal de gravar en l'arxiu CSV cadascun dels possibles valors que s'hagin sel·leccionat als "checkbox"
> NOTA: Fixa't en l'ús de la funció ***header()*** per tal d'afegir una capçalera a la resposta del servidor que en aquest cas és la capçalera "Location", la qual serveix per informar al navegador que cal que continuï la seva navegació en un altre destí. Això a la pràctica fa que la pàgina PHP sigui "invisible" per l'usuari perquè no li transmet cap contingut visible: només en fa un processament i tot just després de fer-lo reenvia l'usuari, com si fos un trampolí, a una tercera pàgina que, aquesta sí, serà visible. Aquesta forma de funcionar veurem que s'utilitza molt quan implementem diverses pàgines pertanyents a una mateixa sessió, on la transmissió de variables de sessió entre pàgines sovint es realitza a partir de pàgines PHP "frontissa" invisibles.

c) Modifica el formulari HTML per a què inclogui un element "input" que permeti pujar fitxers al servidor. Modifica llavors el codi PHP que processa les dades rebudes per a què, a més de fer el que ja feia (guardar en un fitxer CSV les dades recollides del formulari), a més a més sigui capaç de rebre el fitxer indicat per l'usuari i guardar-lo en una carpeta anomenada "uploads" dins del sistema de fitxers del servidor Apache. A la teoria tens els codis de referència per poder-ho implementar tot.

4.-a) ¿Què faria aquest codi? (pots provar-ho per comprovar-ho)

```
<?php   $str1 = "The quick          brown fox";   echo preg_replace("/\s+/", "", $str1)."\n";   ?>
```

aII) I aquest?

```
<?php   $str = "Twinkle, little star,\nHow I wonder what you are.\nUp above the world so high.";
echo preg_replace("/\s+/", " ", trim($str))."\n";  ?>
```

b) ¿Què faria aquest codi? (pots provar-ho er comprovar-ho)

```
<?php   $str1 = "$12,334.00A";   echo preg_replace("/[^0-9,.]/", "", $str1)."\n";   ?>
```

c) ¿Per a què serviria el següent codi, pertanyent a una hipotètica pàgina PHP?

```
if(!preg_match("/\/[A-Za-z0-9-_.]+\.(png|jpe?g)\?.*$/", $_SERVER["REQUEST_URI"] ) { die("No ho fas bé");  }
```

d) La següent funció, ¿retornaria *true* o *false*?

```
preg_match("/w3scho{1,2}ls\.$/i", "Visit W3Schools.");
```

5.-a) A partir de llegir l'article següent, http://www.hackingwithphp.com/10/1/0/cookies-vs-sessions (i també http://www.hackingwithphp.com/10/1/1/cookies i http://www.hackingwithphp.com/10/1/2/sessions) , digues en quines circumstàncies seria preferible utilitzar "cookies" i en quines altres variables de sessió

b) Implementa un comptador de visites mitjançant l'ús de cookies. En concret, crea una pàgina anomenada "comptador.php" amb el següent codi i visita-la varis cops. ¿Què passa? ¿Per què? Tanca el navegador i torna a visitar aquesta mateixa pàgina. ¿Què passa ara? ¿Per què?

```
<?php
 if(isset($_COOKIE["comptador"])) {
   setcookie("comptador", $_COOKIE["comptador"] + 1, time() + 365 * 24 * 60 * 60);
   $missatge = "Nombre de visites: " . $_COOKIE["comptador"];
 } else {
   setcookie("comptador", 1, time() + 365 * 24 * 60 * 60);
   $missatge = "Benvingut a la nostra pàgina web!";
 }
?>
<!DOCTYPE html><html><body>
<?php echo($missatge); ?>
</body></html>
```

c) Explica per a què serviria el següent pedaç de codi, el qual hipotèticament forma part d'un fòrum de missatges:

```
<?php
   if (!isset($_COOKIE["Ordering"])) {  setcookie("Ordering", $_POST["ChangeOrdering"], time() + 36000);  }
?>
<form method="post" action="$_SERVER["PHP_SELF"]"> Reorder messages:
   <select name="ChangeOrdering">
       <option value="DateAdded ASC">Oldest first</option>
       <option value="DateAdded DESC">Newest first</option>
       <option value="Title ASC">By Title, A-Z</option>
       <option value="Title DESC">By Title, Z-A</option>
   </select>
   <input type="submit" value=" Save Settings " />
</form>
```

6.-a) Crea un formulari amb una caixa de text i una altra de tipus "password" que representarà un formulari d'inici de sessió. A partir de les dues dades introduïdes (nom i contrasenya), crea sengles variables de sessió. El formulari de login ha de donar pas a una segona pàgina, la qual només constarà d'un enllaç a una tercera pàgina, la qual només constarà d'un enllaç a una quarta pàgina, la qual haurà de mostrar el nom i contrasenya introduïda al formulari d'inici.

> NOTA IMPORTANT: Has de poder evitar que algun usuari "aterri" directament en una pàgina intermitja pertanyent a una sessió (això pot passar si escriu directament la seva URL al navegador sense haver passat prèviament per la pàgina inicial). Per fer això hauràs de comprovar sempre primer l'existència (amb *isset()*, *empty()*, etc) de les variables de sessió que t'interessin. Una comprovació complementària que podries fer és veure si la variable $_SERVER["REQUEST_METHOD"] val "POST"

b) Implementa un comptador de visites fent servir variables de sessió En concret, crea una pàgina anomenada "comptador2.php" amb el següent codi i visita-la varis cops. ¿Què passa? ¿Per què? Tanca el navegador i torna a visitar aquesta mateixa pàgina. ¿Què passa ara? ¿Per què?

```
<?php
   session_start();
   if(isset($_SESSION["comptador"]))  {
      $_SESSION["comptador"] = $_SESSION["comptador"] + 1;
      $missatge = 'Nombre de visites: ' . $_SESSION["comptador"];
   } else  {
      $_SESSION["comptador"] = 1;
      $missatge = 'Benvingut a la nostra pàgina web!";
   }
?>
<!DOCTYPE html><html><body>
<?php echo($missatge); ?>
</body></html>
```