# M7 - Apuntes UF1

# HTTP

## Headers

HTTP (Hypertext Transfer Protocol). Tota la World Wide Web (WWW) fa servir aquest protocol. Quasi tot el que apareix en un navegador ha estat establert amb HTTP, amb request i responses, entre navegador i servidor. Els HTTP headers són la part central dels HTTP requests i responses, ja que transmeten informació sobre el navegador del client, pàgina sol·licitada, del servidor, etc. El client envia un request i el servidor un response. Quan s'observa el codi font d'un lloc web els headers no apareixen, però s'han rebut. Els HTTP requests s'envien i reben per a moltes altres coses, imatges, arxius CSS, arxius Javascript, etc. Quan càrregues una pàgina web s'envien i reben per cada resource.

## Estructura d'un HTTP request

Un HTTP request es compon de:

- **Mètode:** GET, POST, PUT, etc. Indica que tipus de request és.
- **Path:** la URL que se sol·licita, on es troba el *resource.
- **Protocol:** conté HTTP i la seva versió, actualment 1.1.

En els headers es troben les dades de les cookies. La majoria de headers són opcionals. Si s'envia informació al servidor a través de POST o PUT, aquesta va en el body.

## Mètodes HTTP

Els mètodes més importants d'HTTP (especialment per a fer aplicacions REST) són POST, GET, PUT, DELETE i HEAD. Durant el curs hem après GET i POST.

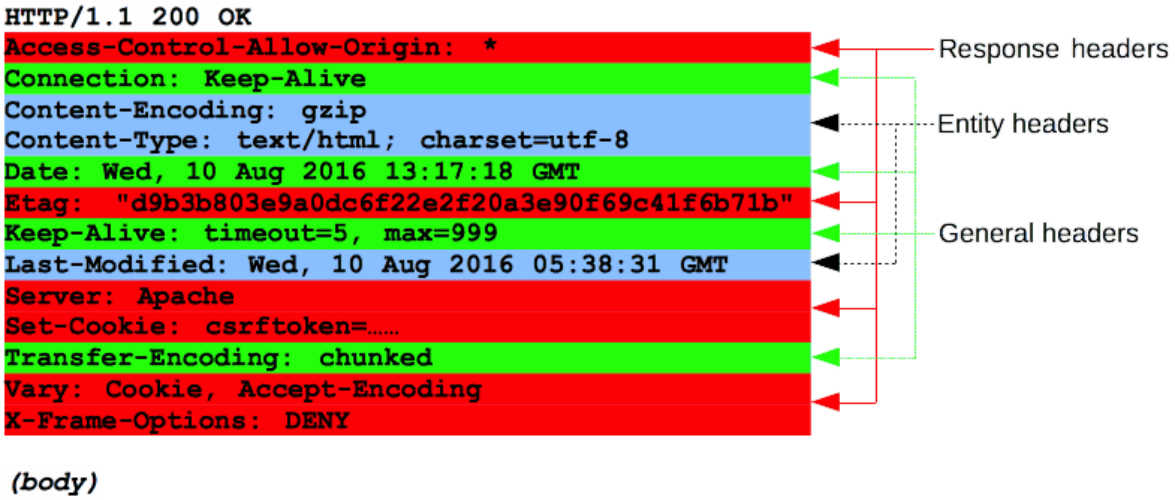
- **GET:** S'empra per llegir una representació d'un resource. Si és positiva, donarà la resposta en un format concret, ja sigui HTML, CSS, etc. En cas que sigui negatiu, mostrarà 404 (not found) o 400 (bad request). Els formularis poden usar el mètode GET, i els valors es mostren al header.
- **POST:** Es poden enviar dades il·limitades pel post a diferència del get que només pot 200 caràcters aproximadament. Si és positiu retorna 201 (created), el POST no mostra informació al header. Aquest mètode serveix per enviar text llarg o tipus d'arxiu.
- **PUT:** S'utilitza normalment per actualitzar continguts, però també per crear-los. No mostra informació al header. A diferència de POST és idempotent, si es crea o edita un resource amb PUT i es fa el mateix request una altra vegada, el resource encara és aquí i manté el mateix estat que en la primera crida. Si amb una crida PUT es canvia encara que sigui només un comptador en el resource, la crida ja no és idempotent, ja que es canvien continguts.
- **DELETE:** Simplement elimina un resource identificat en la URL. Si s'elimina correctament retorna 200 juntament amb un bodi response, o 204 sense bodi. DELETE, igual que PUT i GET, també és idempotent.

- **HEAD:** Igual que GET, però no retorna contingut HTTP Al enviar-se un HEAD request, només s'està interessat en el codi de resposta i headers HTTP. Amb això, el navegador pot comprovar si un document s'ha modificat o si un arxiu existeix.

## Estructura d'un HTTP response

Quan s'envia un HTTP request, el servidor respon amb un HTTP response, compost per:

- **Protocol:** Conté HTTP i la versió (1.1 actualment).
- **Estatus code:** És el codi de resposta. Un exemple seria OK, vol dir que el GET request ha estat satisfactori i el servidor retorna els continguts del document sol·licitat. 404: el servidor no ha trobat el resource sol·licitat.
- **Headers:** Informació sobre el programari del servidor. Quan es va actualitzar per últim cop el resource. La majoria són opcionals.
- **Body:** Informació que no sigui header.




## Headers HTTP en HTTP requests

Els headers poden obtenir-se amb $_SERVER. La funció getallheaders() retorna tots els headers.

Els headers més comuns són:

- **Host:** nom del host, incloent el domini i subdomini (si existeix). S'obté amb $_SERVER['HTTP_HOST'] o $_SERVER['SERVER_NAME''].
- **User-Agent:** informació com el nom o versió del navegador, sistema operatiu i l'idioma per defecte. Amb això els webs poden saber informació sobre el sistema dels visitants. Si visita des d'un mòbil, poden redirigir-lo a una versió mòbil.
- **Accept-Language:** header que mostra el llenguatge per defecte de l'usuari. Es pot redirigir amb $_SERVER['HTTP_ACCEPT_LANGUAGE'].
- **Accept-Encoding:** Formats de codificació suportats per el navegador. Es pot enviar un HTML comprimit, estalviant temps de càrrega.
- **If-Modified-Since:** Si el contingut no s'ha modificat, el servidor retornarà un codi de resposta 304 (Not modified) i el navegador carregarà el contingut de la cache.
- **Cookie:** Envia les cookies guardades al navegador per aquest domini.

- **Referer:** conté la URL de referència. La pàgina de destí apareixerà com referer de l'anterior.
- **Authorization:** Quan es sol·licita autorització, el navegador obre una finestra de login. Al enviar les dades de registre, el navegador envia un altre request que conté la dada codificada en base64. $_SERVER['PHP_AUTH_USER'] i $_SERVER['PHP_AUTH_PW'].

## Headers HTTP en HTTP responses

En PHP es poden establir els codis de resposta amb la funció header(). PHP ja envia alguns headers automàticament com per a carregar el contingut o establir cookies. Es poden veure els headers enviats, o que seran enviats, amb la funció headers_list(). Es pot saber si els headers ja s'han enviat amb headers_sent().

- **Content-Type:** mime-type del document. El navegador decideix com interpretar els continguts. Una pàgina HTML o un PHP amb output HTML. També pot decidir utilitzar una app externa o una extensió del navegador (Com els PDFs de Adobe Reader+)
  Ejemplo: Content-Type: text/html; o Content-Type: application/pdf.
- **Content-Disposition:** Indica al navegador que obri una caixa de descàrrega en lloc d'analitzar el contingut. Ha d'anar acompanyat del content-type
- **Content-Lenght:** El contingut s'enviarà al navegador. Es pot indicar la grandaria amb bytes. Això és especialment útil per a la descàrrega d'arxius, així el navegador pot calcular el progrés de la descàrrega.
- **Location:** S'utilitza en les redireccions. Envia també els codis de resposta.
  Exemple: header('Location: http://www.google.com');
- **Set-Cookie:** Quan un lloc web vol establir o actualitzar una cookie en el navegador es quan s'utilitza. Cada cookie s'envia en un header separat. S'estableix amb setcookie(); Si no s'especifica la data, s'eliminarà quan es tanqui el navegador.
- **WWW-Authenticate:** S'utilitza per identificar a l'usuari per HTTP. Quan el navegador veu el header, obre una finestra de login.
- **Content-Encoding:** Header enviat quan el contingut està comprimit.

## Autenticació HTTP

HTTP suporta diversos mecanismes (basats en l'ús del codi d'estat 401 i el response header WWW-Authenticate) d'identificació per a controlar l'accés a pàgines o altres resources.

- Basic: El client envia el nom d'usuari i contrasenya sense encriptar en codificació textual base64. Només ha d'usar-se amb https, ja que la contrasenya pot ser fàcilment capturada.
- Digest: Contrasenya enviada en forma de hash. Molt més difícil d'obtenir els valors.

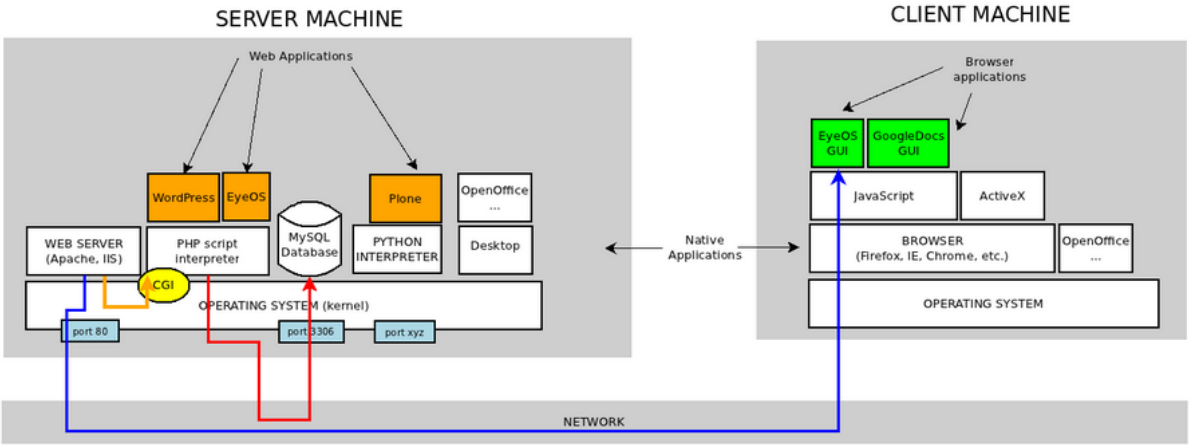
## Codis d'estat

Els codis d'estat HTTP són respostes breus a les sol·licituds web, dividits en 5 categories: 1XX (respostes informatives), 2XX (sol·licituds correctes), 3XX (redireccions), 4XX (errors del client) i 5XX (errors del servidor). Exemples incloent 200 (Correcte), 404 (No trobat) i 500 (Error intern del servidor). Indiquen l'estat d'una sol·licitud web i s'utilitzen en la comunicació servidor-client. Les respostes 2XX indiquen èxit, les 3XX redirigeixen, les 4XX

assenyalen errors del client i les 5XX representen errors del servidor. Els codis varien en significat, des de "Continuar" fins a "Error intern del servidor".

# PHP Elements del llenguatge

## Introducció




PHP és un llenguatge per generar codi HTML. De fet, les sigles **"PHP"** signifiquen "**P**HP: **H**ypertext **P**reprocessor" (definició recursiva).

PHP s'executa en el servidor, mai al client.

PHP és un llenguatge d'escriptura del servidor que ha estat dissenyat específicament per a la web.
Dins d'un html es pot incrustar PHP que s'executara cada vegada que es visiti la pàgina. El codi php s'interpreta al servidor web i genera HTML o una altra sortida que el visitant veurà.

Un fitxer PHP és només un fitxer HTML guardat amb extensio .php enlloc de .hmtl, que indica al servidor que busqui codi. PHP i HTML son el mateix fins que s'afegeixi l'etiqueta <?php

## Estàtiques VS Pàgines web dinàmiques

Pàgina web estàtica → no canvia mai.
Pàgina web dinàmica → pàgina canviant a partir de la lectura de codi del servidor abans d'enviar la pàgina al navegador.

PHP (Hypertext PreProcessor) neix el 1994 (Rasmus Lerdof). A 2007 s'instal·la a +21M de dominis a tot el món. És un projecte de codi obert i gratuït. Orientat principalment per al desenvolupament a nivell web. S'executa desde servidor i el resultat és despatxat a l'usuari mitjançant HTML pla. Això es realitza gràcies als servidors *PHP.

El motor PHP ignora tot el que hi ha fora de les etiquetes PHP.

## Guia d'Estil de Codi

La comunitat de desenvolupadors de PHP és àmplia i diversa, amb moltes biblioteques, frameworks i components disponibles. Per facilitar la col·laboració entre desenvolupadors i la combinació de diverses biblioteques en un projecte, és important que el codi PHP segueixi un estil de codificació comú.

El Framework Interop Group (anteriorment conegut com el 'PHP Standards Group') ha proposat i aprovat una sèrie de recomanacions d'estil, conegudes com a PSR-0, PSR-1, PSR-2 i PSR-4. Aquestes són pautes que diversos projectes importants han adoptat, com Drupal, Zend, CakePHP, phpBB, AWS SDK, FuelPHP, Lithium, entre altres.

Es recomana escriure codi PHP que segueixi un o més d'aquests estàndards, la qual cosa facilita la llegibilitat i la col·laboració entre desenvolupadors, fins i tot quan es treballa amb una gran quantitat de codi de tercers.

Es recomana utilitzar l'eina PHP_CodeSniffer per comprovar que el teu codi compleix amb les recomanacions d'estil i hi ha plugins per a editors de text com el Sublime Text 2 que ofereixen anàlisi en temps real.

Pots utilitzar PHP Coding Standards Fixer de Fabien Potencier per modificar la sintaxi del teu codi automàticament perquè compleixi aquests estàndards, estalviant temps i esforç en comparació amb la correcció manual.

És aconsellable utilitzar l'anglès per als noms de variables i la infraestructura del codi, mentre que els comentaris es poden escriure en un llenguatge fàcilment llegible per a tots els autors i desenvolupadors futurs que treballin en el codi base. Això ajuda a mantenir la coherència i facilita la col·laboració.

## Paradigmes de Programació

PHP és un llenguatge flexible i dinàmic. Ha evolucionat dràsticament a través dels anys.

- v5.0 → s'afegeix un model d'objectes (object oriented)
- v5.3 → funcions anonimes i espais de noms
- v5.4 → traits i millores xe rendiment
- v7.0 → control d'errors

## Programació Orientada a Objectes

PHP inclou l'habilitat de crear classes, classes abstractes, interfícies, herència, constructors, clonació d'objectes, excepcions i molt més.

## Programació Funcional

PHP pot declarar funcions de primera classe (és a dir, una funció pot ser assignada a una variable). Les funcions també poden ser referenciades per una variable i invocades dinàmicament. A més a més, les funcions poden ser passades com a arguments a altres funcions i poden retornar altres funcions.

Recursivitat → aspecte que permet a una funcio cridar-se a si mateixa. (PHP permet això, no obstant, PHP s'enfoca generalment en iteracions).

## Programació Meta

PHP suporta diverses formes de programació meta per mitjà de mecanismes com l'API o mètodes màgics (___get(), ___set(), ___clone()…).

# XDEBUG

És una de les eines més útils en el desenvolupament de programari, un depurador (debugger). Permet veure el traçat d'execució del codi i monitorar el contingut de la pila d'execució. XDebug, és un depurador per PHP, i es pot fer servir en diversos IDE's. (Si tens un problema i amb var_dump no veus solució, hauràs de debuggar).

Els depuradors gràfics fan molt fàcil el procés de recórrer el codi, inspeccionar variables, i avaluar el codi en temps d'execució. Diversos IDE's tenen integrat o poden ser integrats via plugin la depuració gràfica amb XDebug.

# Bones pràctiques - fonaments

PHP permet a qualsevol desenvolupador produir codi de manera ràpida i de manera eficient. Però sovint s'oblida les pràctiques bàsiques necessàries per fer un bon document PHP.

# Bones pràctiques - Data i Hora

PHP disposa de DateTime per a assistir en la lectura, escriptura, comparació i càlcul de la data i hora. Existeixen moltes funcions en PHP relacionades amb la data i hora pero ninguna tan eficaç.

```php
<?php
$data = '11.2.1971';
$inici = \DateTime::createFromFormat('d. m. Y', $data);

echo "Data d'inici: " . $inici->format('m/d/Y') . "\n";
```

Es pot utilitzar DateInterval per realitzar calculs amb DateTime. Mai es pot escriure codi que comptiamb el mateix nombre de segons en tots els dies ja que el canvi horari invalidaria aquesta suposició. En comptes d'aixo, utilitza intervals utilitzant diff() i DateInterval.

```php
<?php
// Crea una copia de $inici y afegeix un mes i 6 díes
$final = clone $inici;
$final->add(new \DateInterval('P1M6D'));

$diff = $final->diff($inici);
echo "Diferencia: " . $diff->format('%m mes, %d dies (total: %a díes)')
. "\n";
// Diferencia: 1 mes, 6 díes (total: 37 díes)
```

És possible comparar fàcilment els objectes DateTime:

```php
<?php
if($inici < $final) {
    echo "L'inici està abans que el final!\n";
}
```

Aquest últim exemple demostra com s'utilitza la classe DatePeriod per a iterar sobre esdeveniments periòdics. L'objecte pren dos objectes DateTime, un per a l'inici i l'altre per al final, i l'interval que defineix el nombre d'esdeveniments periòdics que es retornen.

```php
<?php
// Imprimir tots els dijous entre $inici i $final
```

```php
$intervalDePeriode = \DateInterval::createFromDateString('first thursday');
$iteradorDePeriode = new \DatePeriod($inicio, $intervalDePeriode, $final, \DatePeriod::EXCLUDE_START_DATE);
foreach($iteradorDePeriode as $data)
{
    // Imprimir cada data en el periode
    echo $data->format('m/d/Y') . " ";
}
```

## Bones pràctiques - Patrons de Disseny

En una aplicació és molt bo utilitzar patrons d'ús comú en el codi i patrons per a l'estructura general del seu projecte. Fer servir aquests patrons ajuda i facilita el maneig del codi i permet a altres desenvolupadors entendre fàcilment com encaixen totes les parts de l'aplicació. (Es veurà amb més profunditat més endavant).

## Bones pràctiques - Treballant amb UTF-8

S'ha de ser acurat, detallista i consistent.
Ara mateix, PHP no suporta unicode a baix nivell.
Intenta sempre usar les funcions mb_ per a treballar amb cadenes Unicode. Per exemple, si usa substr() sobre una cadena UTF-8, hi ha una gran possibilitat que el resultat inclogui alguns caràcters il·legibles. La funció adequada a usar és la seva contrapartida multibyte, mb_substr().

No totes les funcions de cadena tenen una contrapartida mb_.

S'ha d'usar la funció mb_internal_encoding() al principi de qualsevol script PHP que escrigui (o en la part superior de la seva script global), i la funció mb_http_output() just després, si el seu script genera sortides cap a un navegador.

## Bones pràctiques - Seguretat en Aplicacions Web

És imperatiu que prenguis les precaucions necessàries i treballi a millorar la seguretat del projecte.
Llegir la Guia de seguretat de OWASP
Aquesta guia conté tota una llista per a la seguretat d'una aplicació.

## Bones pràctiques - Proves

La creació de proves automatitzades per al codi PHP és considerada una bona pràctica i condueix a aplicacions sòlidament construïdes. Les proves automatitzades són una gran eina que assegura que l'aplicació no sigui afectada negativament en fer canvis o en afegir noves característiques al codi.

## Bones pràctiques - Caché

PHP és molt ràpid, però poden sorgir embotellaments quan realitza connexions remotes, carrega arxius i coses per l'estil. Afortunadament, existeixen diverses eines disponibles per a accelerar certes parts de l'aplicació o reduir el nombre de vegades que s'executen aquestes tasques que consumeixen tant de temps. (El caché)

## Estructura de fitxers

Un projecte web consisteix en un conjunt de fitxers dins diferents carpetes. Perquè un fitxer pugui ser interpretat per PHP ha de tenir l'extensió .php, i podrà haver-n'hi tants com considerem oportuns, sent el fitxer principal index.php. En els fitxers PHP, es barrejarà codi PHP amb codi HTML i d'altres llenguatges com JavaScript, i **la seva funció és generar una resposta en format HTML a les peticions rebudes des de clients.**

La instrucció “echo” serveix per mostrar el contingut de variables o cadenes. PHP buscarà en el codi les etiquetes (<?php ?>) i la resta ho ignorarà. Les instruccions PHP estan separades per **;** i els comentaris es fan amb **//**, **#** o **/* */**.

Les instruccions estaran formades per paraules reservades, com FUNCTION, RETURN, FOR, IF, i d'altres expressions. Aquestes **paraules reservades** i el nom de les funcions, **NO son case-sensitive**. Per tant, és igual IF que if. **Les variables, sí que són case-sensitive**, $valor i $Valor són dos variables diferents.

## Tipus de dades

PHP admet 8 tipus de dades primitives, els quals són:

- Escalars → boolean, integer, float, string.
- Composts → array, object
- Especials → resource, null

## Variables

Hi ha dos tipus en PHP:

- **Constants:** valors que durant tot el programa serà el mateix.
- **Variables:** guardarà un valor que podrà canviar durant el programa.

En PHP són molt diferents a Java i C++.

- Les variables comencen amb el caràcter $ i son case-sensitive.
- No és necessari definir el tipus de dada a emmagatzemar abans d’utilitzar-la, es creen en el moment de fer-les servir, i es modifica el seu tipus de forma dinàmica (a mesura que s’utilitzen).
- Tenen **tipat dèbil**, que vol dir que poden contenir dades de qualsevol tipus, a diferència de les variables “fortament tipificades” com a Java, que s’ha d’especificar el tipus quan es creen les variables. A PHP totes les variables poden contenir un boolean, float, String, int, etc.
- El nom de les variables pot començar per lletra o barra baixa (underscore) i pot contenir números.

Les variables es declaren automàticament quan se'ls assigna un valor, per exemple: $dia = 11; i es poden fer operacions amb operadors (+, -, *, /).

**MOLT IMPORTANT:** Fixeu-vos que per concatenar text (entre cometes) i les variables usem el punt (“.”). Tenim una 2a versió on podem posar les variables directament dintre del text, però només funciona amb les dobles cometes. Si volem posar un text amb dòlars potser caldrà posar cometes simples **(no substitueix els valors de les variables)**.

## Constants

Per definir una constant utilitzem la instrucció "**define**": define("PI", 3.1416); Codi: echo " El número Pi és " . PI . " "; enlloc de "PI" el navegador mostrarà "3.1416". Es pot fer servir també per estils. define("COLOR_LLETRA", "#FF5A14"); echo " <font color='". COLOR_LLETRA . "'>". MISSATGE."</font> "; Aquesta tècnica s'utilitza per exemple per definir en un arxiu a part els colors d'un tema. Per canviar un skin es pot fer canviant les definicions. De forma similar es poden canviar missatges en diferents idiomes.

## Àmbit de les variables

Les variables que es declaren fora de funcions o classes es troben en l'àmbit global (global scope), disponibles en qualsevol part del script. Les funcions són blocs independents, les variables que es defineixen dins d'una funció tenen àmbit local (local scope) i no afecten altres en el script global, de la mateixa forma que les variables globals no estan disponibles dins de les funcions. És possible emprar una variable global dins d'una funció amb la paraula global o amb l'array $GLOBALS:

```
$y = "Soc Y";
function funcioDos(){
   $GLOBALS['y'] =& $x;
}
```

## Strings - Cadenes amb cometes simples

Cadenes simples on els caracters s'utilitzen literalment.
$nom = Alex;
echo '$nom '; —> resposta: $nom

## Strings - Cadenes amb cometes dobles

Cadenes complexes, permeten caracters especials.
$nom = Alex;
echo "$nom"; → **Alex**

## Strings - Heredoc

Per declarar cadenes complexes igual que les cometes dobles (no recomanat).

## Operadors

- Operadors Aritmètics: +, -, *, /, %
- Operadors d'Increment:
    - ++$a (Pre-increment) → Incrementa $a en un, i després retorna $a.
    - $a++ (Post-increment) → Retorna $a i després incrementa en 1 a $a.
    - --$a (Pre-decrement) → El mateix que increment, però decrementant.
    - $a-- (Post-decrement) → El mateix que increment, però decrementant.
- Operadors de Cadenes: punt (.), cura amb les "" o ''.
- Operadors de Comparació: <, >, <=, >=, <>(és diferent), ==, ===, !==, !=.
- Operador Ternari: $resultat = (condició) ? condició_cert : condició_falsa; → If curt

- Operadors Lògics: &&, ||, !
- Operadors Assignació: =, +=, -=, *=, /=, %=, .=.

## Arrays

En PHP tenim:

- arrays ennumeratives → les mateixes arrays que a java i les per defecte.
- arrays asociatives → arrays on l'índex no és numèric i s'estableix manualment

```
$persona=array("nom"=>"Juan Antonio", "cognom1"=>"Aguilar", "cognom2"=>Amo", "edat"=>45);
```

També es poden fer arrays bidimensionals:

```
$notes["Mat"]["Primera"]="suficient";
      $notes["Mat"]["Segons"]="Bé";
      $notes["Mat"]["Ter"]="notables";
      $notes["Len"]["Primera"]="Excel·lent";
      $notes["Len"]["Segons"]="Excel·lent";
      $notes["Len"]["Ter"]="notables";
```

La funció range() retorna una matriu que només conté els elements indicats entre 2 números, ambdós inclosos.

## Arrays Associatius

Els arrays associatius són arrays que les seves claus són strings personalitzats. Per a accedir als valors d'un array associatiu es fa de la mateixa forma que amb arrays numèrics, mitjançant claudàtors:

```
$dni = [
    "nom" => "Juan Antonio",
    "primerCognom" => "Aguilar",
    "segonCognom" => "Amo",
    "numero" => "44003600A"
];
echo $dni["primerCognom"].PHP_EOL;
```

Les claus són case-sensitive i es poden fer servir accents.

Afegir o modificar elements d'un array associatiu es fa de la mateixa forma que en arrays numèrics:

```
$dni["dataExpedició"] = "11/1/2009";
```

Als arrays associatius es pot accedir utilitzant cometes dins dels claudàtors o sense utilitzar-les. No utilitzar-les funciona, encara que és incorrecte. Quan s'empren variables com keys no cal utilitzar cometes simples, perquè interpretarà literalment la variable. Si s'empren cometes dobles en les keys, PHP interpretarà correctament els seus valors. E_NOTICE → error. El que s'inclou en els claudàtors ha de ser una expressió. Pot ser una variable, una constant, el return d'una funció, etc.

## Funcions útils

| Funció | Descripció |
|---|---|

| | |
|---|---|
| sizeof ($arr) | Aquesta funció retorna el nombre d'elements d'una matriu. Utilitzeu aquesta funció per esbrinar quants elements conté una matriu; aquesta informació s'utilitza amb més freqüència per inicialitzar un comptador de bucles en processar la matriu. |
| array_values ($arr) | Aquesta funció accepta una matriu PHP i retorna una nova matriu que conté només els seus valors (no les seves claus). La seva contrapart és la funció array_keys (). |
| array_keys($arr) | Aquesta funció accepta una matriu PHP i retorna una nova matriu que conté només les seves claus (no els seus valors). La seva contrapart és la funció array_values ().<br>Utilitzeu aquesta funció per recuperar totes les claus d'un array associatiu. |
| array_pop($arr) | Aquesta funció elimina un element del final d'una matriu. |
| array_push($arr, $val) | Aquesta funció afegeix un element al final d'una matriu. |
| array_shift($arr) | Aquesta funció elimina un element del començament d'una matriu. |
| array_unshift($arr, $val) | Aquesta funció afegeix un element al començament d'una matriu. |
| each($arr) | Aquesta funció s'utilitza més sovint per recórrer iterativament una matriu. Cada vegada que es crida each( ), torna la parella actual de valor clau i trasllada el cursor de matriu cap a l'element següent. Això el fa més adequat per utilitzar-lo en bucle. |
| sort($arr) | Aquesta funció ordena els elements d'una matriu en ordre ascendent. Els valors de les cadenes estaran ordenats en ordre alfabètic ascendent. Nota: Altres funcions d'ordenació inclouen asort( ), arsort( ), ksort( ), krsort( ) i rsort( ). |
| array_flip($arr) | La funció intercanvia les claus i els valors d'una matriu associativa de PHP. Utilitzeu aquesta funció si teniu una estructura tabular (files i columnes) en una matriu i voleu intercanviar les files i columnes. |
| array_reverse($arr) | La funció inverteix l'ordre dels elements d'una matriu. Utilitzeu aquesta funció per tornar a ordenar una llista ordenada de valors de manera inversa per a un processament més fàcil, per exemple, quan intenteu començar amb el mínim o el màxim d'un conjunt de valors ordenats. |
| array_merge($arr) | Aquesta funció fusiona dos o més matrius per crear una única matriu composta. Les col·lisions clau es resolen a favor de la darrera entrada. Utilitzeu aquesta funció quan haureu de combinar dades de dues o més matrius en una sola estructura. Per exemple, registres de dues consultes SQL diferents. |
| array_rand($arr) | Aquesta funció selecciona un o més elements aleatoris d'una matriu. Utilitzeu aquesta funció quan haureu de seleccionar |

| | |
|---|---|
| | aleatòriament una col·lecció de valors discrets, per exemple, escollint un color aleatori d'una llista. |
| array_slice($arr, $offset, $length) | La seva funció és útil per extreure un subconjunt dels elements d'una matriu, i generar una altra matriu. L'extracció comença des del desplaçament de matriu $offset i continua fins que la porció de la matriu té un llarg de $elements. Utilitzeu aquesta funció per dividir una matriu més gran en altres més petites, per exemple, quan segmenteu una matriu per mida ("chunking") o per tipus de dades. |
| array_unique($data) | Aquesta funció separa una matriu de valors duplicats. Utilitzeu aquesta funció quan haureu d'eliminar elements no singulars d'una matriu, per exemple, quan creeu una matriu per retenir valors per a la clau primària d'una taula. |
| array_walk($arr, $func) | Aquesta funció "camina" per una matriu, aplicant una funció definida per l'usuari a cada element. Retorna la matriu canviada. Utilitzeu aquesta funció si necessiteu realitzar un processament personalitzat en tots els elements d'una matriu, per exemple, reduint un 10% de sèries de números. |

# Control de flux: CONDICIONALS

El control de flux és el que identifica un llenguatge de programació. Sense control de flux no hi ha programa. (HTML no té control de flux, per això no se'l considera llenguatge de programació).

Un programa està format per:

- Estructura de dades
- Control de flux
- Algorismes

Amb el control de flux el què podrem fer serà:

- Realitzar bucles
- Realitzar operacions condicionals

# Control de flux: BUCLES

El control de flux és un element clau en la programació que permet gestionar la lògica d'execució d'un programa. Aquest control de flux es pot realitzar mitjançant bucles i operacions condicionals. A continuació, es descriuen els principals tipus de bucles en PHP:

Bucles FOR: Aquests bucles s'utilitzen quan es coneix el nombre d'iteracions que es vol realitzar. S'inicialitza una variable, es defineix una condició de finalització i es determina com s'incrementa la variable a cada iteració. Aquesta estructura és útil per a realitzar operacions repetitives. Exemple:

```
for ($i = 1; $i <= 5; $i++) { // ($i = 0; $i < 5; $i++)
   echo "<H$i>Iteració $i: títol de categoria $i</H$i>\n";
}
```

Bucles FOREACH: Aquests bucles són adequats per recórrer estructures de dades, com ara matrius o altres col·leccions, sense la necessitat de conèixer el nombre d'elements. En cada iteració, es treballa amb una clau i un valor de l'estructura de dades. Exemple:

```
foreach ($array as $index => $valor) {
    // Bloc per cada iteració
}
```

Bucle WHILE: Aquest bucle s'utilitza quan no es coneix el nombre d'iteracions anticipadament. L'execució del bucle es basa en una condició, i el bucle continua mentre aquesta condició sigui certa. És essencial assegurar-se que la condició canviï durant les iteracions per evitar bucles infinits. Exemple:

```
while ($condició) {
    // Bloc per cada iteració
}
```

Bucle DO-WHILE: Aquest tipus de bucle és semblant al bucle while, però garanteix que el bloc de sentències s'executi almenys una vegada, ja que la condició es comprova al final de cada iteració. Com amb el bucle while, cal assegurar-se que la condició canviï en algun moment per evitar bucles infinits. Exemple:

```
do {
    // Bloc per cada iteració
} while ($condició);
```

És important recordar que la gestió adequada del control de flux és crucial per evitar errors i assegurar-se que els bucles compleixin les seves funcions de manera eficient. Cal tenir en compte les característiques específiques de les estructures de dades en PHP, com els índexs no correlatius en les matrius associatives, per evitar errors en les iteracions.

Fer servir <pre> per imprimir els resultats bé:

```
print "<pre>\n";
print_r($matriu);
print "</pre>\n";
```

## Variables pre-definides

PHP genera automàticament una sèrie de variables amb diversa informació sobre el client i el servidor.

## $_REQUEST / $_GET / $POST

$_request és una matriu associativa que conté les dades enviades pels formularis i les cookies guardades en l'ordinador del client.

$_get emmagatzema les variables enviades amb el mètode GET. Les dades es guarden de manera visible i amb caràcters màxims, i es dupliquen a _request

$_post emmagatzema les variables enviades amb el mètode POST. Les dades es guarden encriptades i sense caràcters màxims, i es dupliquen a _request.

## $_SERVER

$_SERVER és una matriu associativa que conté informació sobre capçaleres, rutes i ubicacions de scripts subministrada pel servidor.

$_SERVER[PHP_SELF] conté la direcció de la pàgina (relatiu a l'arrel, és a dir, sense el nom del servidor).

## Funcions

És un conjunt d'instruccions que es poden invocar les vegades que faci falta. Poden rebre paràmetres, que son variables dins de la funció, que se'ls assigna valors en el moment de la seva invocació i només estan disponibles dins de l'àmbit de la funció. És interessant crear funcions per a la majoria d'accions més o menys sistemàtiques que realitzem en els nostres programes.

Per declarar una funció es fa servir "function" seguit del nom de la funció. Després uns parèntesis per indicar els paràmetres, i posteriorment el codi.

## Paràmetres de les Funcions

La informació pot subministrar-se a les funcions mitjançant la llista de paràmetres, una llista de variables i/o constants separades per comes.PHP suporta passar paràmetres per valor (el comportament per defecte), per referència, i paràmetres per defecte. Per defecte, els paràmetres d'una funció es passen per valor (de manera que si canvies el valor de l'argument dins de la funció, no es veu modificat fora d'ella). Si vols permetre a una funció modificar els seus paràmetres, has de passar-los per referència.

Si vols que un paràmetre d'una funció sempre es passi per referència, pots anteposar un ampersand (&) al nom del paràmetre en la definició de la funció: function add_some_extra(&$cadena). Una funció pot definir valors per defecte per als paràmetres escalars estil C++: function makecoffee($type="caputxí"). El valor per defecte ha de ser una expressió constant, i no una variable, un membre d'una classe o una crida a una funció. Cal destacar que quan es fan servir paràmetres per defecte, aquests han d'estar a la dreta de qualsevol paràmetre sense valor per defecte; d'una altra manera les coses no funcionaran de la forma esperada, **($gust, $tipus="acidophilus").**

A partir de PHP4 es suporten les llistes de longitud variable de paràmetres en les funcions definides per l'usuari. No es necessita cap sintaxi especial, i les llistes de paràmetres poden ser escrites en la crida a la funció i es comportaran de la manera esperada. És molt fàcil, usant les funcions proporcionades per PHP.

- Func_num_args( ): Retorna el nombre de paràmetres passats a una funció de paràmetres variable.(L'índex d'un paràmetre dins d'una funció va des de 0 fins a n-1, on n representa el nombre de paràmetres de la funció)
- Func_get_arg( ): Retorna el valor d'un paràmetre en particular. Per exemple, per a retornar el valor del primer paràmetre d'una funció de paràmetres variable, podem escriure: $arg1 = func_get_arg(0);
- Func_get_args( ): Retorna tots els paràmetres passats a la funció en forma d'array.

```
function showtitles() {
   for ($i=0; $i<func_num_args(); $i++) {
```

```
        echo (func_get_arg($i). "\ n");
    }
}
```

## Estructurar el codi d'una aplicació amb les nostres pròpies llibreries de funcions

És recomanable col·locar les funcions en arxius externs, per després cridar-les. S'anomenen llibreries. La manera d'incloure'ls en el nostre script és a partir de la instrucció require o include:

- require("ruta_al_fitxer.php")
- include("ruta_al_fitxer.php")

Tant require() com include() fan el mateix treball, de portar-se codi que hi ha en arxius diferents dins del servidor, perquè puguem utilitzar-lo en crear una pàgina. La diferència fonamental entre require i include és que la primera requereix forçosament alguna cosa i l'altra no. És a dir, si fem un require() d'un arxiu i aquest no es troba disponible per qualsevol motiu, PHP pararà l'execució del codi i retornarà un "Error fatal". Si per contra fem un include() i l'arxiu que tractem de portar no es troba disponible, llavors el que PHP ens mostrarà és una señal d'advertiment, un "warning", però tractarà de continuar executant el programa.

Abans de llançar-nos a crear la nostra pròpia funció, val la pena donar un cop d'ull a la documentació per a veure si aquesta funció ja existeix o podem aprofitar-nos d'alguna de les existents per a alleugerir el nostre treball.

## Funcions pre-definides

PHP basa la seva eficàcia principalment en la seva enorme biblioteca de funcions. Una gran llibreria que creix constantment, a mesura que noves versions van sorgint i es van incorporant noves àrees de treball dins del llenguatge.

## Debuggar

Mostra el tipus i contingut d'una variable. $edad=38;
var_dump($edad);

## Funcions Matemàtiques

```
echo "L'arrel quadrada de 4 és: ".sqrt(4);
echo "Número aleatori entre 1 i 10: ".rand(1,10);
echo "Arrodoniment: ".round(3.45678);   //Mostraria 3
echo "Arrodoniment: ".round(3.5678);    //Mostraria 4
echo "Arrodoniment: ".round(3.5678,2);  //Mostraria 3.57
```

## Funcions per a Variables

```
echo gettype($edat);                    //Mostraria integer
is_string($variable), is_float($variable), …
isset($variable);                       //mostra si existeix la variable o no existeix
```

trim($variable); //Elimina espais per davant i per darrere d'una variable. Molt útil per a formularis.
if(empty($nom)) echo "Variable buida"; //Comprova si una variable està buida

## Funcions per a cadenes de text

strlen() //Comptar caràcters d'una cadena de text (d'un string)
strpos() //cerca un caràcter o text dins d'una cadena de caràcters.
str_replace() //reemplaça paraules d'una cadena de caràcters
strtolower(); //converteix a minúscules
strtoupper(); //converteix a majúscules
explode("separador",$text); //Converteix un text en array, tallant pel separador.
implode("separador",$text); //Converteix un array en text, afegint el separador per posicio
substr($text, $inici, $longitud); // Obtens la cadena resultant entre la posicio inici fins longitud
htmlspecialchars($text); // Transforma els caracters com < i > a caracters en format text (&lt)
htmlentities($text); //Retorna la mateixa cadena arreglant caracters ilegibles com la ñ
strip_tags($text,"etiq_permeses"); //Elimina totes les etiquetes d'HTML i PHP del text passat

## Funcions per a Array

asort() //Ordena Alfabèticament
arsort() // Ordena Alfabèticament en ordre invers
array_push($array,$element_a_afegir) //Afegir elements a l'última posició del array
array_pop($array) //Treu l'element de l'última posició del array
unset(array[posició]) //Treu l'element del array de la posició indicada
array_search() //Cerca un element dins d'un array
count //Compte els elements d'un array

## Dates

En PHP es fa servir la funció date() per donar format de les dates i temps. date(format, timestamp); $format és obligatori i especifica el format. $timestamp és opcional i especifica una marca de temps (un moment). Per defecte és l'actual.

El paràmetre de format de la funció date() és una cadena que pot contenir diversos caràcters que permeten generar dates en diversos formats. Caràcters de format relacionats amb les dates que s'utilitzen habitualment en la cadena de format:

| | |
|---|---|
| d | Day of the Month (Numeric 01 or 31) |
| D | Day of the Week (String Mon to Sun) |
| l | Day of the Week (String Mon to Sun) |
| F | Month (String Jan to Dec) |
| M | Month (String Jan to Dec) |
| m | Month (Numeric 01 to 12) |
| Y | Year in four-digit (Numeric 2008 or 2020 ) |
| y | Year in two-digit (Numeric 08 or 20 ) |
| h | Hour (in 12-hour format) |
| H | Hour (in 24-hour format) |
| a | AM or PM |
| i | Minute |
| s | Second |

La funció time() s'utilitza per obtenir l'hora actual. PHP pot convertir automàticament un valor de data en una marca de temps UNIX amb la funció mktime(), que accepta dia, mes, any, hora, minut i segons arguments i retorna una marca de temps UNIX corresponent a aquest instant a temps.

## Treball amb arxius

OPENDIR i READDIR, funcions amb les que podrem llegir els continguts d'una carpeta. En realitat necessitem 3 funcions per poder llegir el directori:

- opendir (obre carpeta)
- readdir (llegeix un nom d'arxiu)
- closedir (tanca la carpeta per si la vols obrir de nou).

## Solució simplificada pel readdir

```
<ol>
	<?php
		$dir = opendir("carpeta1");
		$arxiu = readdir($dir);
		while( $arxiu!==false ){
			echo "<li>$arxiu</li>";
			$arxiu = readdir($dir);
		}
		closedir($dir);
	?>
</ol>
```

- Obrim el directori amb OPENDIR
- Llegim el primer nom d'arxiu abans d'entrar a bucle WHILE
- Un cop dins el WHILE imprimim el nom de l'arxiu i llegim després el següent nom d'arxiu.
- Ens quedem dins el WHILE sempre i quan el nom de l'arxiu sigui diferent de false (voldrà dir que hem acabat de llegir el directori).

# Sessions

Una sessió de PHP ens permetrà enregistrar variables i reprendre-les més endavant (en posteriors visites o recàrregues de la web). Donat que HTTP és un protocol stateless o sense estat (també es sol dir protocol REST), cada cop que visitem una pàgina web es crea una nova connexió i a l'acabar el diàleg client-servidor (request-response) es destrueixen les dades. Està pensat per optimitzar els recursos de servidor, però ens complica força el poder mantenir un procés que ocupi diverses visites a la aplicació web.

Les sessions permeten que pàgines diferents puguin accedir a una variable comuna, la matriu $_SESSION. El treball amb sessions té tres parts:

- Creació o obertura de la sessió: Quan alguna pàgina crea una sessió utilitzant la funció corresponent, el servidor associa al navegador de l'usuari un identificador d'usuari únic. L'identificador es guarda en l'usuari en forma de cookie o, si el navegador de l'usuari no permet la creació de cookies, afegint l'identificador en la direcció de la pàgina.
- Utilització de la sessió: Si ja s'ha creat la sessió, les pàgines sol·licitades pel mateix navegador poden guardar i recuperar informació en el servidor, informació que s'associa a l'identificador d'usuari, per la qual cosa no és accessible a altres usuaris. La informació es conserva fins que l'usuari o el servidor destrueixin la sessió.
- Destrucció o tancament de la sessió: Tant l'usuari com el servidor poden tancar la sessió. L'usuari pot destruir la sessió tancant el navegador. El servidor pot destruir la sessió quan alguna pàgina utilitzi la funció corresponent o al cap d'un temps determinat (definit mitjançant la funció session_set_cookie_params()).

# Crear la sessió

En PHP, les sessions es creen mitjançant la funció session_start(). Si la sessió no existia, aquesta funció crea la sessió i li associa un identificador de sessió únic. Si la sessió ja existia, aquesta funció permet que la pàgina tingui accés a la informació vinculada a la sessió. És a dir, totes les pàgines que vulguin guardar dades en $_SESSION o llegir dades de $_SESSION han de començar amb la funció session_start(). La creació de sessions requereix l'enviament de capçaleres HTTP, per la qual cosa la funció session_start() ha d'utilitzar-se abans de començar a escriure el contingut de la pàgina. En cas contrari PHP produirà un avís i no es crearà la sessió. El motiu és que l'identificador de sessió s'utilitza en les capçaleres de resposta HTTP i les capçaleres s'envien abans del text de la pàgina.

# Utilitzar la sessió

Quan una pàgina ha creat una sessió o ha accedit a una sessió ja existent mitjançant session_start(), la pàgina té accés a la matriu $_SESSION que conté les variables d'aquesta sessió.

La matriu $_SESSION és una matriu associativa en la qual es poden definir valors com en qualsevol altra matriu. La diferència és que $_SESSION és accessible des de pàgines diferents (sempre que aquestes pàgines tinguin associada la mateixa sessió), mantenint-se els valors d'una pàgina a una altra.

El primer caracter ha de ser un _

## Tancar la sessió

Tancar una sessió en PHP implica destruir la matriu $_SESSION i l'identificador de la sessió associat. Això es pot fer de diverses maneres:

L'usuari pot tancar la sessió simplement tancant el navegador.
Un programa pot tancar la sessió mitjançant la funció session_destroy().
La durada d'una sessió es pot establir mitjançant session_set_cookie_params() en el moment de la seva creació.
Després de destruir una sessió, el programa que ho ha fet encara té accés als valors de $_SESSION creats abans de la destrucció, però les pàgines següents no en tenen accés.

## Nom de la sessió

En principi, quan el navegador es connecta a un servidor, la sessió és única, és a dir, totes les pàgines del mateix domini compartiran la mateixa matriu $_SESSION. La funció session_name() permet establir un nom de sessió específic, de manera que totes les pàgines que declarin el mateix nom de sessió accediran a la mateixa matriu $_SESSION i les que tinguin noms de sessió diferents accediran a matrius $_SESSION diferents.

El nom de sessió es case-sensitive (distingeix entre minúscules i majúscules), és a dir, dues sessions amb el mateix nom, però un en minúscules i un altre en majúscules, són dues sessions diferents. El nom de la sessió no pot contenir únicament números, ni tampoc pot contenir els caràcters, espai ( ), punt (.), ampersand (&), més (+), claudàtor esquerre ([) ni coixinet (#).

## Esborrar elements de la sessió

Per esborrar els valors de la matriu $_SESSION en PHP, es pot utilitzar la funció unset(). Per a eliminar tots els valors de $_SESSION, és millor utilitzar la funció session_unset(). No obstant això, cal tenir en compte que: Utilitzar unset($_SESSION) impedeix que la resta de la pàgina accedeixi o modifiqui els valors de $_SESSION, però la sessió encara manté aquests valors, de manera que altres pàgines podrien seguir veient-los.

La funció session_unset() elimina tots els valors de $_SESSION, però permet que la resta de la pàgina i altres pàgines accedeixin i modifiquin els valors de $_SESSION.

## Directiva session.save_handler

Per a utilitzar sessions mitjançant el mecanisme propi de PHP (és a dir, sense necessitat de crear funcions pròpies), la directiva session.save_handler de l'arxiu de configuració php.ini ha de tenir el valor files. La funció ini_set() abans d'obrir la sessió.

## Exemples Sessions

https://aulavirtual.iesthosicodina.cat/moodle/mod/lesson/view.php?id=59309&pageid=325

# Cookies

Les cookies són com a arxius de text que es guarden en la màquina client. A petició d'un servidor web, el navegador crea un arxiu d'aquest tipus.  Una vegada que ocorre això el servidor pot llegir i escriure contingut en aquest arxiu.

- Els servidors web només poden accedir a cookies establertes al seu propi domini
- Segons el protocol HTTP, les cookies no poden ser més grans de 4096 Bytes
- Hi ha un límit de cookies per domini. Depèn del navegador, però solen ser 20 cookies.
- També hi ha un límit en el nombre total de cookies (300)

Les cookies tenen una data d'expiració que permet al navegador eliminar cookies antigues quan ja no són requerides pel servidor web.

# Protocol HTTP

Les cookies es transmeten a través del protocol HTTP. Aquest és el protocol que utilitzen els navegadors per a enviar i rebre arxius d'un servidor

# Crear i llegir cookies

Les cookies es poden crear de diverses formes i amb diferents llenguatges.

El més important a l'hora de crear una cookie en PHP és que s'ha d'establir abans d'enviar qualsevol tipus de dada al navegador. Això inclou qualsevol tipus d'etiqueta html, echo, print, etc.
setcookie($nom, $valor, $expiracio, $ruta, $domini, $seguretat (true || false), $nomeshttp (true || false));

L'opció de $seguretat força al fet que la cookie només sigui enviada si s'ha establert una connexió segura HTTPS. $nomeshttp fa que la cookie només estigui disponible a través d'http.

Per llegir les cookies existeix un array anomenat $_COOKIE, que s'utilitza per a extreure els valors de la cookie.

# Exemple: login

https://aulavirtual.iesthosicodina.cat/moodle/mod/lesson/view.php?id=59309&pageid=326

# Exemple: idioma

https://aulavirtual.iesthosicodina.cat/moodle/mod/lesson/view.php?id=59309&pageid=326

## Treballant amb Sessions i Cookies

Son exemples de Sessions i Cookies, a més que explica les parts de cadascuna:
https://aulavirtual.iesthosicodina.cat/moodle/mod/lesson/view.php?id=59309&pageid=378

## Expressions regulars

Les expressions regulars permeten definir patrons de coincidència i aplicar-les a cadenes de text per a saber si la cadena (o part d'ella) compleix el patró i fins i tot realitzar transformacions de la cadena.

## Funcions d'expressions regulars compatibles amb Perl

La funció d'expressions regulars compatibles amb Perl comprèn una cadena amb un patró i retorna 1 si el patró ha coincidit o 0 si no.

## Errors en PHP

Podem dir que un error és una condició de tenir un coneixement incorrecte o fals, o també definir-ho com un estat de programa invalidat inesperat a partir del qual és impossible recuperar-se.

S'envia al navegador un missatge d'error amb nom de fitxer, número de línia i un missatge que descriu l'error.

Tipus d'error:

- Error d'anàlisi (error de sintaxi)
  - Es produeix si hi ha un error de sintaxi en el script. La sortida es Parse Error. Els motius habituals son: cometes sense tancar, falten o sobren parèntesis, claudators sense tancar, falta punt i coma.
- Error fatal
  - Es produeix quan PHP entén el codi, però no el pot fer. Exemple: intentar accedir a funcions no definides
- Error d'alerta
  - Es produeix, principalment, quan falta incloure un fitxer o s'ha utilitzat el nombre de paràmetres incorrectes en una funció
- Error de notificació
  - Es produeix quan s'intenta accedir a una variable no definida.

L'script només s'atura en l'error d'anàlisi i l'error fatal.

# Apuntes Alex

## Protocolo HTTP

- **TCP:** Se encarga de que la información llegue sin errores.
- **IP:** Se encarga de que llegue al destino.
- **HTTP:** Se encarga de pedir recursos al servidor.

El URL (Uniform Resource Locator) da acceso a un recurso usando un protocolo de comunicación.

**¿Qué convierte la URL en un código?** El DNS.

El protocolo HTTP (Hypertext Transfer Protocol) define la sintaxis y la semántica que utilizan los elementos de software en la arquitectura web para comunicarse. (Protocolo orientado a transacciones que sigue el esquema petición-respuesta entre cliente y servidor).

**Haces petición:**
MÈTODE VERSIÓ URL<crlf>
CAPÇALERA: Valor<crlf>
. . . CAPÇALERA: Valor<crlf>
Línia en blanc <crlf>
COS DE LA SOL.LICITUD

**El ordenador te envía respuesta con cabecera:**
GET http://es.kioskea.net HTTP/1.0 Accept : Text/html If-Modified-Since : Saturday, 15-January-2000 14:37:11 GMT User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)

**Y seguidamente lo que contiene (imagen, HTML, lo que sea):**
GET: solicita el recurso situado en la URL especificada.
HEAD: solicita el encabezado del recurso.
POST: envía datos al recurso.
PUT: envía datos a la URL especificada.
DELETE: borra el recurso de la URL específica.

**CÓDIGOS DE RETORNO:**

| | | | |
|---|---|---|---|
| 10x | Missatge d'informació | **ERRORS DE CLIENT** | |
| 20x | Èxit | 40x | Error degut al client |
| 200 | OK | 400 | BAD REQUEST |
| 201 | CREATED | 401 | UNAUTHORIZED |
| 202 | ACCEPTED | 402 | PAYMENT REQUIRED |
| 203 | PARTIAL INFORMATION | 403 | FORBIDDEN |
| 204 | NO RESPONSE | 404 | NOT FOUND |
| 205 | RESET CONTENT | **ERRORS DE SERVIDOR** | |
| 206 | PARTIAL CONTENT | 50x | Error degut al servidor |
| 30x | Redirecció | 500 | INTERNAL ERROR |
| 301 | MOVED | 501 | NOT IMPLEMENTED |
| 302 | FOUND | 502 | BAD GATEWAY |
| 303 | METHOD | 503 | SERVICE UNAVAILABLE |
| 304 | NOT MODIFIED | 504 | GATEWAY TIMEOUT |

## Lenguajes de programación

HTML: lenguaje estático para el desarrollo en sitios web
JAVASCRIPT: lenguaje de cliente interpretado sin compilación, es imperativo y estructurado, dinámico y funcional! Alerta: nunca poner credenciales ni nada por el estilo pq es to peligroso en cuanto a seguridad y cositas así.

<script type="text/javascript"> … </script>

PHP: lenguaje de servidor interpretado sin compilación, páginas web dinámicas
<?php
$missatge = “Hola”;
echo $missatge;
?>

ASP: PHP, pero para ricos de mierda sin criterio alguno que ven que algo es de Microsoft y dicen: este es el bueno seguro (es broma microsoft xfavor contratame).
<% %>

## Diferencias entre un archivo PHP y HTML

No existen diferencias entre ellos excepto que se incluya la etiqueta <?php. Si no existe esta etiqueta, se interpreta como un HTML normal, por lo tanto, aparece escrito. En caso de que la etiqueta exista, será un archivo PHP. Ambas páginas son estáticas, si solo hay HTML. Si en una página PHP se añaden las etiquetas PHP (<?php), el servidor será capaz de interpretar el lenguaje.

Se puede mezclar HTML con PHP. Se puede generar PHP en medio de HTML.

PHP al ser un lenguaje no tipado, es decir que puedes asignar una variable y puede ser de cualquier tipo, puede llegar a causar problemas, porque lo que hace es ir cambiando la variable en función del código. En este caso mostrará el párrafo, ya que es el último valor que se asigna.

```php
<?php
    $valor = 7;
    $valor = 7.55;
    $valor = "<P>El primer paràgraf en PHP: fem servir l'instrucció 'echo'. </P>";
?>

<HTML>
<BODY>
    <H1>Primer arxiu PHP</H1>
    <?php
    #El comentario en PHP se genera con #. Tambien se puede utilizar // y /*
       echo $valor;
    ?>
</BODY>
</HTML>
```

Por otro lado, con el recurso var_dump, este valor se guarda y, por lo tanto, imprime los 3. El último está comentado, porque no es necesario, porque es el último valor que se guarda. Ejemplo:

```php
<?php
    $valor = 7;
    var_dump($valor);
    $valor = 7.55;
    var_dump($valor);
    $valor = "<P>El primer paràgraf en PHP: fem servir l'instrucció 'echo'. </P>";
    //var_dump($valor);
?>

<HTML>
<BODY>
    <H1>Primer arxiu PHP</H1>
    <?php
    #El comentario en PHP se genera con #. Tambien se puede utilizar // y /*
       echo $valor;
    ?>
</BODY>
</HTML>
```

En las versiones antiguas a la 7 se podían realizar operaciones con un int y un String, este se convertía automáticamente en 0. A partir de la 7, salta un pantallazo en blanco. Pero, por otro lado, se puede sumar un 7.5 + “1.8”, ya que PHP hace la conversión automática de String a float, pero no es del todo perfecta. (Mirar primer ejercicio).

Variables de sistema → variables especiales y únicas del propio sistema. Muchas cosas de configuración se establecen en **php.ini**

‘ ‘ → comillas simples, saldrá exactamente lo que hay dentro.
“ ” → comillas dobles, texto complejo, entiende de código.

Heredoc → para declarar cadenas complejas (no recomendado por Toni, sin embargo, dice que hay gente a la que le encanta, además el Toni dice que le encanta Eclipse, así que viendo sus gustos, capaz que Heredoc es la puta polla).
\ → caracter de escape
== → comparamos valor
=== → comparamos valor y tipo
Cuando trabajamos con string (y con cualquier parida de PHP prácticamente tmb), lo primero es ver si hay una funcion que hace lo que queremos nosotros.

Operación ternaria → especie de if resumido

Los array son diferentes de java, en PHP, por cada posición, puedes tener una variable de diferente tipo, además las posiciones no tienen por qué seguir un orden numérico y pueden tener identificadores alfabéticos. Los arrays sin orden se llaman arrays asociativos.

bucles foreach → forma típica de iterar estructuras de más de un elemento en PHP
while → cuando no sabemos el número de iteraciones.

$server → tiene info del servidor
$request/get/post → guarda las variables de nuestra petición. Lo que haces en get y/o en post, se copia en request.

Diferencia entre get y post: get es visible y tiene un número máximo de caracteres, post no.
funcion → palabra clave function, se le pueden definir parámetros obligatorios y no obligatorios.

Los parámetros recibidos en funcion pueden ser trabajados por:
valor → comportamiento por defecto, como java, lo que modificas no se ve reflejado afuera.
referencia → lo que modificas, se ve reflejado afuera.