Univ. Babeș-Bolyai,

Facultatea de Matematică și Informatică

Lect. dr. Darius Bufnea

Notițe de curs Programare Web: PHP (săptămâna 10 și 11 de școală)

Ce este PHP?

Este o tehnologie server side / limbaj de programare open-source de back-end – care se execută la nivelul serverului web. PHP este un limbaj de scripting (nu este compilat), este interpretat, numele său fiind abreviere de la "**P**HP: **H**ypertext **P**reprocessor". Ca să poată executa cod PHP, serverul web trebuie să fie dotat cu un modul care să permită execuția codului PHP. Apache (serverul web open source dominant pe platformele UNIX/Linux) suportă oarecum nativ execuția de cod PHP (dar și în cadrul Apache execuția de cod PHP poate fi dezactivată). Celalalt server web dominant din Internet , IIS de la Microsoft (Internet Information Service) suportă și el execuția de cod PHP prin intermediul unor module 3rd party (dar oarecum integrarea PHP-ului cu IIS-ul nu este la fel de naturală și firească precum legătura dintre Apache și PHP). Ca și concluzie, don't try to run PHP scripts using IIS…

Pe de altă parte, PHP-ul poate fi folosit și ca limbaj de scripting linia de comandă, existand un interpretor executabil numit chiar php (php.exe în Windows) care permite rularea de programe scris în PHP și în afara enviromement-ului oferit de un server Web.

AMP Stack sau stiva AMP

AMP este abreviere de la Apache, Mysql si PHP. De obicei o instalare de PHP nu vine de una singură, ea vine la pachet cu un server web, și după cum spuneam mai sus, cel mai natural se împacă cu Apache (open-source și el). Pe lângă Apache și PHP, în acest environment era nevoie și de un server de baze de date, MySQL fiind una dintre soluțiile open-source din domeniul serverelor de date care s-a bucurat de cel mai mare success, împreună cu serverul web Apache și cu PHP-ul formând o "stivă" de programare în domeniul aplicațiilor web (sau un ecosistem) numit AMP. Uneori îl veți regăsi referit și sub numele de LAMP stack (L venind de la Linux). Paranteză: MySQL dupa ce a ajuns în ograda Oracle a suferit un „fork", născându-se MariaDB (practic tot MySQL). PHP însa suportă și alte servere de date precum PostgreSQL (open source și el), SQL Server de la Microsoft sau Oracle.

Instalare

De ce avem nevoie ca să rulam cod PHP? În primul rând de un server web care să permită execuția de cod PHP, și după cum spuneam mai sus cel mai la îndemână este Apache. În funcție de sistemul de operare pe care vrem să facem instalarea:

Linux: majoritatea distribuțiilor Linux vin sau oferă spre download dintr-un repository atât serverul web Apache cât și modulele necesare ca acesta să fie capabil să ruleze cod PHP. De asemenea, cam toate distribuțiile Linux vin cu MySQL sau MariaDB ca server de baze de date implicit. În funcție de distribuție, aceste pachete se pot instala fie cu apt-get (Ubuntu based distributions) sau cu yum (Redhat/Fedora/CentOS based distributions).

MacOS: not a fan...

Windows: Windows nu are o tradiție prea bună / nu este cel mai bun enviroment posibil pentru ecosisteme open-source precum este stiva AMP. Cu toate acestea, există câteva distribuții bune ale acestei stive pe Windows, una dintre ele fiind XAMPP (Apache + MariaDB + PHP + Perl) pe care vă recomand să o instalați. Practic ce conține XAMPP? E un kit de instalare care instalează pe Windows un server web Apache cu suport de PHP integrat și un server de baze de date MySQL împreună cu o fereastră de control (Control Panel) care permite printre altele gestionarea facilă (oprirea / pornirea) acestor două servere.

Observație (pentru că am văzut mai mult de o dată această greșeală): nu căutați „XAMPP for Linux", nu există așa ceva! Distribuțiile Linux suportă nativ stiva AMP, având pachete dedicate pentru Apache, PHP și MySQL.

Înainte de a vă instala XAMPP

Este foarte posibil să aveți din alte surse gata instalate alte servere web sau să aveți deja MySQL instalat (l-ați instalat pentru MPP). Dacă instalați XAMPP, e posibil să vă loviți de tot felul de conflicte: MySQL din XAMPP nu poate ocupa portul 3306 (portul implicit al MySQL/MariaDB) pentru că acest port este deja ocupat, aveți deja un IIS care vă ocupa portul 80 pe care vrea sa îl ocupe și Apache-ul, portul 80 mai este ocupat uneori și de Skype sau diverși antiviruși.

Daca rulați într-un `cmd` un

```
netstat -aon
```

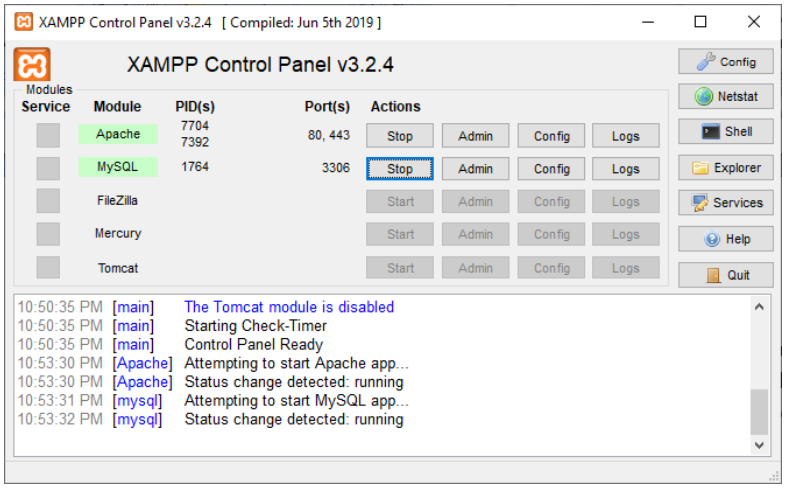
puteți vedea care porturi TCP sunt ocupate și de către ce proces (PID-ul procesului care ocupă un anumit port). Ulterior dacă va uitați în Task Managerul de Windows puteți vedea și identifica procesul cu un anumit PID.

O altă problema este ca multe IDE-uri (Integrated Development Environment - editoarele în care scrieți cod gen Eclipse, Netbeans, IteliJ, PHP Webstorm, Visual Studio) au „prostul obicei" - pentru a face munca web developer-ului mai ușoară - să pornească ele diverse servicii în spate cum ar fi un server web pentru a rula/depana un anumit proiect. Aceste servicii și porturile ocupate de aceste servicii pot intra iarăși în conflict cu diverse servere / pachete instalate de voi anterior.

Dupa instalarea XAMPP, ar trebui din Control Panel-ul acestuia să puteți porni cu succes atât Apache-ul cât și serverul MySQL/MariaDB. Dacă unul dintre ele nu pornește, trebuie să eliberați unul din porturile 80, 443 (HTTPS) sau 3306. Pentru aceasta:

- Verificați folosind comanda `netstat -aon` de mai sus dacă porturile sunt intr-adevăr ocupate și de către ce proces;

- Dacă puteți identifica ușor folosind comanda `netstat -aon` și Task Managerul de Windows procesul care ocupă portul/porturile problematice, opriți procesul respectiv.
- Verificați în Windows Services (click pe Start, scrieți Services, ENTER) că nu aveți un serviciu cu un nume „like" MySQL, IIS, Internet Information Server, Apache sau similar în starea Running (foarte probabil dintr-o instalare similară) și dacă identificați un astfel de proces opriți-l.



Observație: foarte mulți studenți își instalează pentru MPP un server MySQL care rulează tot timpul ca serviciu în background după pornirea laptopului consumând resurse inutile (la fel un SQL server pentru altă materie și un Oracle Express Edition pentru altă materie). Personal prefer serverul MySQL/MariaDB din XAMPP tocmai pentru flexibilitatea pe care mi-o oferă de al controla/porni/opri mai facilă și mai transparentă. Într-un scenariu în care aveti 2 servere MySQL instalate, trebuie să aveți grijă care server MySQL e pornit, sunt dese cazurile în care este pornit unul dintre ele și vreți să vă conectați la el cu anumite credențiale cu care de fapt ar trebuie să vă conectați la celalalt. Numele de utilizator implicit cu care vă conectați la serverul MySQL/MariaDB din XAMPP este root, fără parolă (parola vidă "").

Dacă instalarea XAMPP s-a făcut cu succes, ar trebui să puteți încarcă în browser următoarele URL-uri:

http://localhost

http://localhost/phpmyadmin - PhpMyAdmin este o aplicatie web scrisă în PHP care vă permite administrarea mai facilă a unui server de baze de date (MySQL/Maria DB în cazul nostru).

În caz că nu puteți elibera totuși porturile pentru a porni unul dintre servere, atât Apache cât și MySQL/MariaDB vă permit schimbarea porturilor pe care ascultă aceste servere. Schimbarea porturilor ar trebuie să fie însă ultima soluție pentru că ridică ulterior din nou alte probleme de conectare din partea clienților care vor să se conecteze la aceste servere ce folosesc porturi nestandard. Spre exemplu, conectarea din browser la un server web Apache care așteaptă pe portul 81 (nu pe portul standard 80 al protocolului HTTP) ar trebui să se facă cu un URL de forma: http://localhost:81. Schimbarea portului în Apache se face folosind directiva `Listen` din cadrul fișierului său de configurare `httpd.conf`.

Daca ați instalat XAMPP într-un director de forma C:\xampp, veți identifica următoarele două directoare:

```
c:\xampp\php\
```
și

`c:\xampp\mysql\bin\`

pe care vă recomand să le adăugați în variabila de mediu PATH pentru a avea acces la interpretorul php.exe și la clientul linie de comandă mysql.exe de oriunde (din orice director) din cadrul sistemului de fișiere în cadrul unui cmd.

Temă: vă recomand după parcurgerea acestui material să vă „jucați" în linia de comandă cu comenzile php.exe, mysql.exe si mysqldump.exe.

Apache-ul ca server web conține un director denumit `DocumentRoot` (acesta se specifică cu ajutorul unei directive denumite chiar `DocumentRoot` prezentă in fișierul de configurare `httpd.conf`) de unde vor fi „livrate" clienților (browserelor) toate fișierele cerute de aceștia. `DocumentRoot`-ul implicit pentru XAMPP este `c:\xampp\htdocs`, practic acest director corespunzând URL-ului rădăcină http://localhost, un fișier de `forma c:\xampp\htdocs\demo\lab1.php` fiind accesibil cu un URL de forma http://localhost/demo/lab1.php.

Personal recomand editarea și rularea fișierelor .php direct dintr-un subdirector al `DocumentRoot`-ului serverului web, după modificarea acestora fiind necesar un refresh / reload (CTRL-F5) în browser. Eventualele erori de execuție sunt raportare de serverul web într-un fișier denumit de obicei `error.log` sau `error_log` util a fi conspectat pentru un debugging mai facil. Ca alternativă, unele IDE-uri specializate oferă mecanisme avansate de debugging, depanare pas cu pas, la rularea de cod PHP într-un astfel de IDE fiind posibil ce acesta din urmă să își pornească propriul server web pe un port nestandard.

Observații:

Serverul vostru Apache din cadrul XAMPP este accesibil și de pe alte calculatoare din cadrul retelei locale cu un URL de forma: http://192.168.1.101 dacă acest lucru este permis de firewall-ul sistemului vostru de operare (a se înlocui cu IP-ul vostru privat din rețeaua locală) sau chiar de oriunde din Internet dacă faceți un DNAT la portul 80 de pe router (recapitulare de la Rețele...).

PHP

Codul php trebuie plasat în interiorul unui fișier cu extensia .php. Un astfel de fișier este executat la nivelul serverului web, execuția unui fișier php fiind în general invocată prin intermediul unui request HTTP sau de către un alt fișier PHP prin diferite mecanisme de includere.

Observație: Frecvent studenții deschid un fișier php direct in browser folosind un URL de tip file:// (spre exemplu file://d:/laboratoare/pw/php/pr1.php). ACEST LUCRU ESTE GREȘIT, într-o asemenea situație, fișierul php nu se execută – nu are cine să-l execute – pentru că serverul web care l-ar putea executa nu este specificat. Un fișier php trebuie întotdeauna „invocat" (apelat, referit) prin intermediul protocolului HTTP folosind un URL de forma http://numedeserver.com/demo.php. În acest exemplu, serverul web numedeserver.com este responsabil să găzduiască și să execute fișierul cu numele demo.php. Excepție de la această observație: rularea unui script php din linia de comandă folosind interpretorul php.

Un fișier .php poate conține fie exclusiv cod php, fie cod php imbricat cu cod html. O secvență de cod php din cadrul unui fișier .php poartă denumirea de scriptlet, fiind relativ frecvente situațiile în care în

cadrul unui fișier .php sunt prezente mai multe scriptlet-uri imbricate cu cod html (imbricate la rândul lor cu cod JavaScript care se execută pe client – și așa se ajunge la ceea ce se cheamă Spaghetti Code... de evitat ☺). Un scriptlet este delimitat de tag-ul `<?php ?>` (atenție, tag exclusiv PHP, interpretabil doar server side).

Un prim exemplu (accesibil aici):

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplul 1</title>
</head>
<body>
    Salut lume. Ora pe server este: <?php echo date('H:i', time()); ?>
</body>
</html>
```

Este util să vizualizați și codul sursă al paginii ce ajunge la browser (click dreapta, view source). Scriptlet-ul php plasat în interiorul tag-ului `<?php ?>` se execută pe back-end, iar la browser ajunge cod HTML „curat" – browser-ul nu este conștient de fapt că ceva se execută pe back-end, el cere prin intermediul protocolului HTTP și a unei metode HTTP precum GET sau POST o anumită resursă specificată prin intermediul URL-ului, resursa în cazul de față fiind fișierul .php a cărui execuție are loc la nivelul serverului web.

Observație: PHP-ul are un fișier de configurare denumit `php.ini` unde există diferite directive de configurare. Una dintre acestea, denumită `short_open_tag` (`= On` sau `= Off`) permite delimitarea scriptlet-urilor folosind așa numitul short tag `<? ?>`. Această abordare nu este recomandată, întrucât proiectul vostru php este posibil să ajungă să fie găzduit și rulat în cadrul unui server web care nu acceptă (și practic nu va recunoaște) short open tag-ul `<?`. În cazul modificării fișierelor `php.ini` sau `httpd.conf` trebuie sa reporniți serverul web Apache.

Daca un scriptlet se găsește la finalul unui fișier PHP, nemaiexistând cod HTML după acesta, marcajul de sfârșit de scriptlet (`?>`) poate să lipsească (altfel spus, dacă marcajul de sfârșit de scriptlet `?>` se regăsește la final în cadrul unui fișier PHP, acesta poate fi omis).

Elemente de sintaxă

Limbajul PHP a împrumutat elemente de sintaxă atât de la limbajul C cât și de la limbajul shell al interpretorului de comenzi SH (sau BASH) din Unix/Linux. Nu vom insista foarte multe pe elementele de sintaxă ale limbajului, prezentul material dorind să insiste mai multe pe aspectele „web related" oferite de limbaj. Vă rog însă ferm ca pe lângă prezentul material să parcurgeți și secțiunea PHP Tutorial de pe W3Schools, întreg limbajul și API-ul oferit de acesta fiind acoperit și de documentația oficială disponibilă aici.

Câteva elemente de sintaxă pe scurt:

- Instrucțiunile de control (`if`, `while`, `for`) sunt identice cu cele din C/C++/Java
- Fiecare linie de cod PHP trebuie să se termine cu caracterul „;"
- Afișarea de cod HTML din interiorul unui scriptlet PHP se poate face cu print sau echo

- Elementele de sintaxă nu sunt case sensitive, dar numele variabilelor sunt;
- Comentariile se introduc în maniera C/C++/JAVA:
  - Single line: // sau # (prealuat din sh)
  - Multiline /* … */

Limbajul php suporta și un tag de scriptlet special `<?= ?>` pentru afișarea de pe back-end de valori „inline" în cadrul codului HTML ce se trimite spre front-end. Folosind acest tag, linia de cod anterioară:

```
Salut lume. Ora pe server este: <?php echo date('H:i', time()); ?>
```

poate fi rescrisă:

```
Salut lume. Ora pe server este: <?= date('H:i', time()) ?>
```

Variabile in PHP

- Toate variabilele în PHP încep cu simbolul $, exemplu: `$varsta = 18;` (precedarea variabilelor cu caracterul $ este un element de sintaxă preluat de la limbajul SHELL, numele variabilelor de sistem UNIX/Linux fiind precedate și ele de caracterul $).
- Variabilele nu trebuie declarate;
- Tipul acestora se deduce în funcție de valoarea asociată, PHP fiind un limbaj weakly typed;
- Numele variabilei (identificatorul acesteia) trebuie să urmeze aceleași reguli ca în limbajele de programare cunoscute.

Stringuri in PHP

- Pot fi delimitate atât de "" (ghilimele) cât și de '' (apostroafe);
- Pentru concatenarea acestora se foloseste operatorul "." Exemplu: `$suma = "In cont am " . 3 . " lei";`
- Pentru compararea șirurilor se poate folosi == dar și `strcmp`;
- Majoritatea funcțiilor de lucru cu șiruri sunt preluate din limbajul C: `strlen, strstr, strpos, strtok`;

Observație: PHP suporta și el operatorul `===` specific limbajelor weakly typed. Spre exemplu, funcția `strpos($sir, $subsir)` verifică apariția `$subsir`-ului în `$sir`, returnând în caz afirmativ indexul primei apariții (valoare numerică) sau `FALSE` (boolean în PHP) dacă `$subsir` nu se regăsește în `$sir`. O comparație de forma `if (strpos($sir, $subsir) == FALSE)` nu este corectă dacă se dorește verificarea apariției `$subsir`-ului în `$sir`, această condiție având valoarea de adevăr `TRUE` și dacă `$subsir`-ul nu se `regăsește` în `$sir` și dacă acesta se regăsește la începutul `$sir`-ului (index 0) – acest lucru pentru că boolean-ul `FALSE` se evaluează la valoarea 0. Corectă în cazul de față este folosirea operatorului `===` care compară și tipul valorii întoarse de funcția `strpos`.

Tablouri în PHP

PHP-ul suportă atât tablouri clasice cu indici numerici, cât și tablouri asociative (un fel de map-uri) cu indicii string-uri. Exemple:

Tablouri unidimensionale (tablouri clasice cu indici valori întregi):

```
<?php
$echipe = array("CFR", "FCSB", "Dinamo");
print $echipe[0];
print count($echipe);
?>
```

Tablouri cu indici de tip string (asociative):

```
<?php
$capitale["Romania"] = "Bucuresti";
$capitale["Ungaria"] = "Budapesta";
$capitale["Franta"] = "Paris";
print $capitale["Romania"];
?>
```

Pentru iterarea unui tablou se poate folosi instrucțiunea de control `foreach`. Exemple:

```
<?php
foreach ($capitale as $capitala)
       print $capitala . "<br>\n";
foreach ($capitale as $tara=>$capitala)
       print $tara . " are capitala " . $capitala . "<br>\n";
?>
```

Observație: în exemplu de mai sus delimitatorul `\n` a fost folosit pentru a delimita liniile de cod sursă HTML care se trimite browser-ului în urma execuției codului PHP sau într-un eventual output pe consolă la execuția codului folosind interpretorul php la linia de comanda, în timp ce tagul `<br>` a fost folosit pentru a delimita efectiv liniile de output afișate de browser utilizatorului.

Limbajul PHP suporta și tablouri multidimensionale:

```
<?php
   $continente = array(
      "Europa"=>array("Romania", "Franta", "Ungaria"),
      "Asia"=>array("China", "India", "Japonia"),
      "Africa"=>array("Egipt", "Maroc", "Nigeria")
   );
   print $continente["Asia"][2];
?>
```

Funcții PHP

Declararea funcțiilor în PHP se face folosind cuvântul rezervat `function` asemănător modului în care se face declararea acestora în JavaScript. Parametrii se pot transmite fie prin valoare (implicit) identic modului în care se transmit în C/C++/Java, fie prin referință - pentru transmiterea parametrilor prin referință parametri formali se prefixează cu operatorul & (asemănător modului în care se transmit prin referință în C++).

Exemplu:

```
function aduna($i, $j) {
  return $i + $j;
}
print aduna(4, 7);
```

Observație: Este deosebit de frecvent ca o funcție în php să returneze valori de tipuri diferite. Este recomandată întotdeauna și verificarea tipului valorii returnate de funcție nu doar a valorii. Spre exemplu, funcția `strpos` amintită mai sus poate întoarce atât un index numeric cât și valoarea de adevăr `FALSE`. Un alt exemplu, diferite funcții de lucru cu baze de date, întorc fie o resursă, fie boolean-ul FALSE în caz de eroare.

Alte elemente de limbaj

Regula "celor trei pahare " într-o singura linie de cod:

```
list($a, $b) = array($b, $a);
```

Observație: `list` nu este o funcție, ci un element oferit de limbajul PHP care permite atribuirea de valori mai multor variabile folosind un singur operator de atribuire.

Variabile globale

Variabilele declarate în afara corpului unei funcții nu sunt accesibile implicit în interiorul acesteia. Pentru a avea acces la ele, acestea trebuie declarate ca globale folosind cuvântul rezervat `global`. Exemplu:

```
<?php
$pi = 3.14;
function circleArea($radix) {
    global $pi;
    return $pi * $radix * $radix;
}
print circleArea(5);
?>
```

Variabile predefinite PHP

PHP-ul oferă din start câteva tablouri asociative predefinite, majoritatea fiind legate de contextul web în care se execută codul PHP. Aceste variabile se mai numesc și superglobale (superglobals), ele fiind accesibile oriunde la nivelul codului PHP fără a mai fi declarate ca globale. Acestea sunt:

- `$GLOBALS`
- `$_SERVER`
- `$_GET`
- `$_POST`
- `$_FILES`
- `$_COOKIE`

- $_SESSION
- $_REQUEST
- $_ENV

cele mai importante fiind $GLOBALS, $_GET, $_POST și $_SESSION. Observație: numele tablourilor sunt GLOBALS, _GET, _POST și _SESSION, ele fiind precedate de $ precum orice nume de variabila în PHP.

Tabloul predefinit GLOBALS permite accesarea tuturor variabilelor globale. Astfel, folosind GLOBALS, funcția precedentă care calculează suprafața cercului poate fi rescrisa altfel:

```
<?php
$pi = 3.14;
function circleArea($radix) {
    return $GLOBALS["pi"] * $radix * $radix;
}
print circleArea(5);
?>
```

Despre $_GET, $_POST și $_SESSION vom vorbi în continuare în acest material.

Prelucrarea datelor din cadrul unui formular

Din cursurile precedente știm că datele din cadrul unui formular pot fi trimise pe back-end folosind una dintre metodele GET sau POST, iar script-ul de pe back-end care primește datele de la formular se specifică ca și valoare pentru atributul action al formularului. Elemente de tip input din cadrul unui formular (împreună cu elementele de tip textarea și option) vor fi accesibile în PHP fie prin intermediul tabloului asociativ $_GET fie $_POST în funcție de metoda prin care s-a făcut submit la formularul respectiv. În cadrul acestor tablouri, indecșii valorilor stocate vor fi numele inputurilor corespunzătoare din formular.

Prezentam mai jos un exemplu care se doreste a fi sugestiv în acest sens (disponibil aici). Front-end:

```
<form method="post" action="register.php">
Nume utilizator: <input type="text" name="username"><br>
E-mail: <input type="text" name="email"/><br>
Parola: <input type="password" name="pass"/><br>
Parola din nou: <input type="password" name="pass2"/><br>
<input type="Submit" value="Register">
</form>
```

Back-end (fișierul register.php):

```
Am primit de la browser:<br>
<?php
   print "Username: " . $_POST["username"] . "<br>\n";
   print "Email: " . $_POST["email"] . "<br>\n";
   print "Parola: " . $_POST["pass"] . "<br>\n";
   print "Parola2: " . $_POST["pass2"] . "<br>\n";
?>
```

```
</body>
```

Datele primite prin completarea formularului mai pot fi accesate și prin intermediul tabloului asociativ `$_REQUEST` (atât la submit-ul prin `GET` cat și la cel prin `POST`), iar `$_SERVER["REQUEST_METHOD"]` conține metoda HTTP prin care s-a realizat cererea.

IMPORTANT: Înainte de a fi folosite pe back-end, datele primite de la client, indiferent de metodă, trebuie VALIDATE OBLIGATORIU. Mai multe în acest sens mai târziu în acest material.