

Curs 5

2018/2019

# Programarea aplicațiilor web

# Introducere

---

# PAW

- Programarea aplicațiilor web
  - An V RC
    - 2C/1L/1P

# Nota

- An V
  - 33% E
  - 66% Aplicatii
    - 33% L (0%)
    - 33% P (66%)

# Documentatie

- <http://rf-opto.etti.tuiasi.ro> + WWW



The screenshot displays the website for RF-OPTO at ETTI. The header features the RF-OPTO logo, the ETTI logo, and the University of Technical Sciences (UTS) logo. Below the header is a navigation menu with links for Main, Courses, Master (highlighted), Staff, Research, and Students. A secondary menu lists Radiocommunication Systems, Microwave IC, and Web Design (highlighted). The main content area is titled "Modern Web Applications Design" and provides details for the course TPAW (2014-2015), including the coordinator, code, discipline type, credits, and enrollment year. It also lists activities, evaluation methods, grades, attendance, and materials.

**RF-OPTO**

English | Romana

Main Courses **Master** Staff Research Students

Radiocommunication Systems Microwave IC **Web Design**

## Modern Web Applications Design

**Course: TPAW (2014-2015)**

**Course Coordinator:** Assist.P. Dr. Radu-Florin Damian  
**Code:** DS507RC  
**Discipline Type:** DOMS; Alternative, Synthesis  
**Credits:** 7  
**Enrollment Year:** 5, Sem. 10

### Activities

**Course:** Instructor: Assist.P. Dr. Radu-Florin Damian, 2 Hours/Week, Specialization Section, Timetable:  
**Laboratory:** Instructor: Assist.P. Dr. Radu-Florin Damian, 1 Hours/Week, Half Group, Timetable:

### Evaluation

Type: **Examen**

**A:** 33.33%, (Test/Colloquium)  
**B:** 33.33%, (Seminary/Laboratory/Project Activity)  
**D:** 33.33%, (Homework/Specialty papers)

### Grades

Not yet

### Attendance

Not yet

### Materials

**Course Slides**

# CURS

I.	HTML si XHTML (recapitulare)	1 oră
II	CSS	2 ore
III	<del>Baze de date, punct de vedere practic</del>	<del>1 oră</del>
IV	<del>Limbajul de interogare SQL</del>	<del>4 ore</del>
V	PHP - HyperText Preprocessor	8 ore
VI	XML - Extended Mark-up Language si aplicatii	4 ore
VII	Conlucrare intre PHP/MySql, PHP/XML, Javascript/HTML	2 ore
VIII	Exemple de aplicatii	6 ore
	Total	28 ore

# rf-opto.etti.tuiasi.ro

- [http://rf-opto.etti.tuiasi.ro/master\\_it.php](http://rf-opto.etti.tuiasi.ro/master_it.php)
- Laborator 2011-2012
  - Exercitii/Performanta MySql
  - optional

Capitolul II

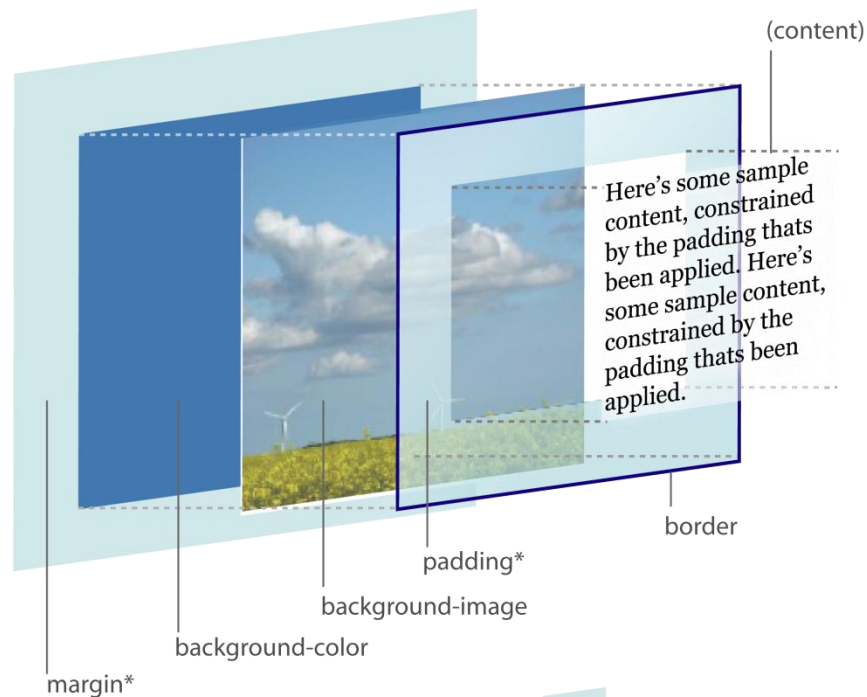
**CSS**



# CSS Box Model

- Orientat in jurul conceptului de "cutie" – Box model

THE CSS BOX MODEL HIERARCHY



\* transparent elements

# CSS Zen Garden

- <http://www.csszengarden.com/>
- un fisier html comun
- schimbarea formei permisa numai prin intermediul CSS

# CSS Zen Garden – HTML original

## css Zen Garden

### The Beauty of CSS Design

A demonstration of what can be accomplished visually through CSS-based design. Select any style sheet from the

Download the sample [html file](#) and [css file](#)

### The Road to Enlightenment

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, and broken CSS

Today, we must clear the mind of past practices. Web enlightenment has been achieved thanks to the tireless effort

The css Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with your eyes.  
Become one with the web.

### So What is This About?

There is clearly a need for CSS to be taken seriously by graphic artists. The Zen Garden aims to excite, inspire, and educate. When you visit, one will load the style sheet into this very page. The code remains the same, the only thing that has changed is the style.

CSS allows complete and total control over the style of a hypertext document. The only way this can be illustrated is by placing it in the hands of those able to create beauty from structure. To date, most examples of neat tricks and hacks

# CSS Zen Garden



## Css Zen Garden

*the beauty of css design.*

*A demonstration of what can be accomplished visually through CSS-based design. Select any style sheet from the list to load it into this page.*

[DOWNLOAD THE SAMPLE HTML FILE AND CSS FILE](#)

### *The Road To Enlightenment*

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, and broken CSS support.

Today, we must clear the mind of past practices. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C,

SELECT A DESIGN:

[Under the Sea!](#)  
by Eric Stoltz

[Make 'em Proud](#)  
by Michael McAghon and  
Scotty Reif'snyder

# Forme in HTML

- necesare pentru ca utilizatorul sa poate trimite date server-ului
- `<form>...</form>`
- Attribute specifice:
  - action: adresa documentului care preia datele
    - `<form action="<?php echo $_SERVER['PHP_SELF'];?>">`
    - `<form action="actiuni/fisier2.html">`
  - method: modalitatea de transmitere a datelor: post sau get
    - `<form method="post" action= ... >`

# Metode de transmitere

- **post** datele sunt transmise in bloc
- **get** datele sunt atasate adresei documentului de procesare : [results.php?prob=81&an=2009](http://results.php?prob=81&an=2009)
- **get** trebuie folosit numai cand datele sunt "idempotente",
  - nu cauzeaza efecte colaterale
  - nu modifica starea server-ului (baze date, etc)
- se poate simula realizarea unei forme (**get**) prin scrierea corespunzatoare a link-urilor

# Elemente de interactiune cu utilizatorul

- in interiorul etichetei `<form>...</form>`
  - `input`
  - `select/option`
  - `textarea`
  - `button`
- **toate** elementele vor avea un nume
  - atribut: `name=""`
  - numele va fi intalnit in uri-ul generat prin get, sau in numele variabilei trimisa prin post
- trimiterea datelor se face prin intermediul unui buton cu tipul `type="submit"`

# INPUT

- `<input .../>`
- Attribute:
  - type: text | password | checkbox | radio | submit | reset | file | hidden | image | button
  - name: numele variabilei
  - value: valoarea trimisa server-ului la selectie (valoarea initiala in anumite cazuri)
  - checked/src/size/maxlength in functie de tip



# Example

```
<input name="textfield" type="text" value="ceva" />
```

```
<input name="Ok" type="submit" value="Trinite" />
```

```
<label><input name="check" type="checkbox" value="5" checked />check1</label>
```

```
<label><input name="RG1" type="radio" value="a" checked="checked" />but1</label>
```

```
<label><input name="RG1" type="radio" value="b" />but2</label>
```

```
<input name="hid" type="hidden" value="6" />
```

ceva

Trinite



check1



but1



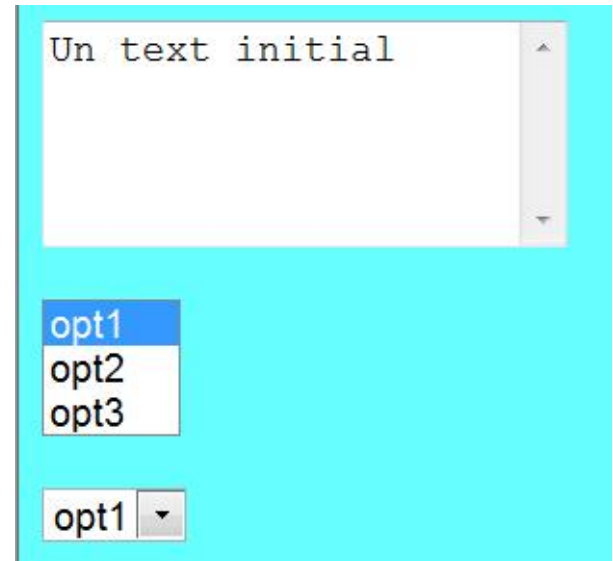
but2

# TEXTAREA/SELECT

```
<textarea name="textarea" cols="20" rows="5">Un text  
initial</textarea><br /><br />
```

```
<select name="select." size="3">  
  <option value="1" selected="selected">opt1</option>  
  <option value="2">opt2</option>  
  <option value="3">opt3</option>  
</select><br /><br />
```

```
<select name="select...">  
  <option value="1" selected="selected">opt1</option>  
  <option value="2">opt2</option>  
  <option value="3">opt3</option>  
</select>
```



The image shows three different visual representations of a select element on a light blue background. At the top is a text area containing the text "Un text initial". Below it is a list box with three options: "opt1", "opt2", and "opt3", where "opt1" is highlighted with a blue background. At the bottom is a dropdown menu with "opt1" selected and a small downward-pointing arrow on the right side.

# CURS

I.	HTML si XHTML (recapitulare)	1 oră
II	CSS	2 ore
III	<del>Baze de date, punct de vedere practic</del>	<del>1 oră</del>
IV	<del>Limbajul de interogare SQL</del>	<del>4 ore</del>
V	PHP - HyperText Preprocessor	8 ore
VI	XML - Extended Mark-up Language si aplicatii	4 ore
VII	Conlucrare intre PHP/MySql, PHP/XML, Javascript/HTML	2 ore
VIII	Exemple de aplicatii	6 ore
	Total	28 ore

Hypertext PreProcessor

**PHP**

# PHP - Concepte

- limbaj interpretat – compilat “on the fly” de interpretorul PHP de pe server
- script-urile PHP contin sursele
  - exista posibilitatea pre-compilarii surselor pentru spor de viteza
  - Hip-Hop for PHP / Facebook
  - HipHop Virtual Machine / Facebook
- orientat spre aplicatii web dinamice (biblioteci)
- poate fi integrat in HTML – utilizarea tipica

# Integrare

```
539         <td><?php echo $row['Documente'];?>&nbsp;</td>
540         <td><?php echo $row['user_creat'];?>&nbsp;</td>
541         <td class="smaller"><a href="control_lot.php?id=<?php echo $row['ID_LOT'];?>">dezactiveaza</a><br /><a href=
"control_lot.php?id=<?php echo $row['ID_LOT'];?>">modifica</a></td>
542     </tr>
543     <?php $index++;
544     } while ( $row = mysql_fetch_assoc($result));?>
545 </table><?php
546 }
547 else
548 {
549     echo "<p>Nu exista loturi active</p>";
550 }
551 ?>
552
553 <p class="title">Loturi inactive</p>
554 <?php
555 $query = "SELECT l.*, c.`nume_user` AS `user_creat`
556         FROM `lot` AS l
557         LEFT JOIN `users` AS c ON (l.`User`=c.`id_user`)
558         WHERE l.`Activ` = 0 ORDER BY l.`ID_LOT` DESC";
559 $result = mysql_query($query);
560 $total=0;
561 if ($result && (mysql_num_rows($result) > 0))
562     {
563     $total=mysql_num_rows($result);
564     $row = mysql_fetch_assoc($result);
565     }
566 if ($total>0)
567 {?>
568 <table align="center">
569     <tr class="lista_titlu">
570         <td align="center">Nr. </td>
```

# Separare cod PHP

- un fisier sursa PHP este un fisier HTML (in general) cu sectiuni de cod PHP
- interpretorul PHP cauta sectiunile pe care trebuie sa le interpreteze si interiorul lor proceseaza instructiuni ca fiind PHP
- ce se gaseste in **exteriorul** acestor sectiuni este trimis spre server-ul web **nemodificat**

# Separare cod PHP

- `<?php ... ?>`
  - stil XML – implicit, disponibil intotdeauna, recomandat
- `<? ... ?>`
  - scurt, este de obicei dezactivat
  - lipsa in PHP 7
- `<script language="php"> ... </script>`
  - stil script, disponibil
- `<% ... %>`
  - stil ASP, de obicei dezactivat
  - lipsa in PHP 7



# Variante de integrare

- echo .... afiseaza un text la "iesire" (echivalent cu printf() din C)
- poate realiza procesarea datelor
- in exemplu se trimite spre iesire un sir static (echivalent cu puts() din C)
- "iesire" in marea majoritate a cazurilor reprezinta datele trimise clientului de server-ul web
- "iesire" poate fi considerata de obicei:
  - documentul curent
  - pozitia curenta

# Variante de integrare

- Toate variantele ofera aceeasi sursa HTML pentru browser
- E **recomandata** cea care lasa structura HTML nemodificata si doar datele dinamice sunt rezultatul procesarii
- Codul HTML + PHP e interpretat mult mai elegant in editoarele WYSIWYG

```
<h2>Rezultate comanda</h2>  
<?php echo '<p>Comanda receptionata</p>';?>
```

```
<h2>Rezultate comanda</h2>  
<p><?php echo 'Comanda receptionata';?></p>
```

```
<?php echo '<h1>Magazin online XXX SRL</h1>';?>  
<?php echo '<h2>Rezultate comanda</h2>';?>  
<?php echo '<p>Comanda receptionata</p>';?>
```

```
<?php  
echo '<h1>Magazin online XXX SRL</h1>';  
echo '<h2>Rezultate comanda</h2>';  
echo '<p>Comanda receptionata</p>';  
?>
```

# PHP – instructiuni

- instructiunile PHP trebuie sa se termine cu ;
  - exceptie: se poate omite ; la sfarsitul blocului de cod php: ... `echo 'ceva' ?>`
- instructiunile pot sa fie scrise pe acelasi rand (fara trecerea la linia noua)
  - `echo 'ceva1'; echo 'ceva2'; ...`
  - nu este in general recomandat

# PHP – comentarii

- comentariile in PHP respecta regulile C si Perl
- la sfarsit de linie:
  - `// echo ceva; //comentariu`
  - `# echo ceva; #comentariu`
- comentariu bloc
  - `/* ... */`
  - `/* un comentariu  
pe mai multe  
linii */`

# PHP – constante

- Ca orice limbaj de programare PHP se bazeaza pe utilizarea
  - constante
  - variabile
  - functii
- Definirea constantelor:
  - `define('PRETCARTE', 100);`
  - "case sensitive"
  - prin conventie, numai cu litere mari
  - `echo PRETCARTE; // 100`

# PHP – variabile

- variabila – semnul \$ urmat de un nume
- numele e “case sensitive”
- o greseala frecventa e uitarea semnului \$
  - PHP Notice: Use of undefined constant an – assumed \$an (sau ‘an’) in D:\\Server\\
- Tipuri de date
  - scalar
  - compus
  - special

# PHP – tipuri de date

- scalar
  - boolean
  - integer
  - float (double)
  - **string**
- compus
  - array
  - object
- special
  - resource
  - NULL

# PHP – tipuri de date

- **declararea** variabilelor **nu** e necesara decat cand se declara un domeniu de definitie (variabile globale)
  - `global $a, $b;`  
`$c=$a+$b;`
- eliberarea memoriei nu este necesara, se face automat la terminarea executiei



# PHP – tipuri de date

```
$var = expresie
```

- Controlul variabilelor se face automat, “on the fly”
  - Daca \$var nu era definita anterior, in urma atribuirii se defineste de tipul dat de rezultatul expresiei
  - Daca \$var era definita, de un anumit tip (oarecare), in urma atribuirii devine de tipul dat de rezultatul expresiei
  - La finalizarea executiei script-ului se elimina variabila din memorie (automat)

# PHP – tipuri de date

- tipul de date este in totalitate dependent de ceea ce se stocheaza
- PHP reactualizeaza tipul pentru a putea primi ceea ce se stocheaza

```
<?php
echo $variabila ; // tip Null, neinitializat – valoare NULL (doar)
$variabila = "0"; // $variabila tip string (ASCII 48)
$variabila += 2; // $variabila tip integer (2)
$variabila = $variabila + 1.3; // $variabila tip float (3.3)
$variabila = 5 + "10 obiecte"; // $variabila tip integer (15)
$var2=5; // $var2 tip integer (5)
$variabila=$var2."10 obiecte"; // $variabila tip string "510 obiecte"
?>
```

# PHP – operatori

- In general similari celor din C/C++
- Operatori
  - Aritmetici
  - Atribuire
  - Bit
  - Comparare
  - Incrementare/Decrementare
  - Logici
  - **Sir**

# PHP – operatori

- Aritmetici
  - $-$a$  – Negare
  - $$a + $b$  – Adunare
  - $$a - $b$  – Scadere
  - $$a * $b$  – Inmultire
  - $$a / $b$  Impartire
  - $$a \% $b$  Modulo (rest)
- Sir
  - **$$a.$b$  – Concatenare sir a si sir b**

# PHP – operatori

## ■ Atribuire

- `$a=$b`
- `$a+=$b` ( $a=a+b$ )
- `$a-=$b` ( $a=a-b$ )
- `$a/=$b` ( $a=a/b$ )
- `$a*=$b` ( $a=a*b$ )
- `$a%=$b` ( $a=a\%b$ )
- `$a.= $b` ( $a=a$  concatenat  $b$  - siruri)

# PHP – operatori

- Operatori la nivel de bit
  - similari celor din C
  - `~, &, |, ^, <<, >>`
- Operatori logici
  - ofera rezultat boolean true/false
  - similari celor din C
  - `&&, ||, !`
  - suplimentar
    - `and, or, xor` – echivalenti dar de prioritate mai mica
    - `$a=55/0 or die('impartire prin 0');`

# PHP – operatori

- Operatori de comparare
  - ofera rezultat boolean true/false
  - similari celor din C
  - == , != , > , < , <> , >= , <=
  - **suplimentar**
    - === identic, valoare egala **SI** de acelasi tip
    - !== “neidentic”, valoare diferita **SAU** de tipuri diferite

# Precedenta operatorilor

non-associative	clone new	<a href="#">clone</a> and <a href="#">new</a>
left	[	<a href="#">array()</a>
non-associative	++ --	<a href="#">increment/decrement</a>
right	~ - (int) (float) (string) (array) (object) (bool) @	<a href="#">types</a>
non-associative	instanceof	<a href="#">types</a>
right	!	<a href="#">logical</a>
left	* / %	<a href="#">arithmetic</a>
left	+ - .	<a href="#">arithmetic</a> and <a href="#">string</a>
left	<< >>	<a href="#">bitwise</a>
non-associative	< <= > >= <>	<a href="#">comparison</a>
non-associative	== != === !==	<a href="#">comparison</a>
left	&	<a href="#">bitwise</a> and <a href="#">references</a>
left	^	<a href="#">bitwise</a>
left		<a href="#">bitwise</a>
left	&&	<a href="#">logical</a>
left		<a href="#">logical</a>
left	? :	<a href="#">ternary</a>
right	= += -= *= /= .= %= &=  = ^= <<= >>=	<a href="#">assignment</a>
left	and	<a href="#">logical</a>
left	xor	<a href="#">logical</a>
left	or	<a href="#">logical</a>
left	,	many uses



# Precedenta operatorilor PHP7

non-associative	clone new	<a href="#">clone</a> and <a href="#">new</a>
left	[	<a href="#">array()</a>
non-associative	++ --	<a href="#">increment/decrement</a>
right	~ - (int) (float) (string) (array) (object) (bool) @	<a href="#">types</a>
non-associative	instanceof	<a href="#">types</a>
right	!	<a href="#">logical</a>
left	* / %	<a href="#">arithmetic</a>
left	+ - .	<a href="#">arithmetic</a> and <a href="#">string</a>
left	<< >>	<a href="#">bitwise</a>
non-associative	< <= > >= <>	<a href="#">comparison</a>
non-associative	== != === !== <=>	<a href="#">comparison</a>
left	&	<a href="#">bitwise</a> and <a href="#">references</a>
left	^	<a href="#">bitwise</a>
left		<a href="#">bitwise</a>
left	&&	<a href="#">logical</a>
left		<a href="#">logical</a>
right	??	<a href="#">comparison</a>
left	? :	<a href="#">ternary</a>
right	= += -= *= /= .= %= &=  = ^= <<= >>=	<a href="#">assignment</a>
left	and	<a href="#">logical</a>
left	xor	<a href="#">logical</a>
left	or	<a href="#">logical</a>
left	,	many uses

# Exemplu de separare cod php

## Raspuns dinamic

- `<form action="rezultat.php" method="post">`

```
<html>
<head>
<title>Magazin online XXX SRL</title>
</head>
<body>
<h1>Magazin online XXX SRL</h1>
<h2>Rezultate comanda</h2>
<p><?php echo 'Comanda receptionata';?></p>
</body>
</html>
```

**Magazin online XXX SRL**

**Rezultate comanda**

Comanda receptionata

```
<
<
<
</head>
<body>
<h1>Magazin online XXX SRL</h1>
<h2>Rezultate comanda</h2>
<p>Comanda receptionata</p>
</body>
</html>
```

# PHP – Functii

- conceptual similare celor din C/C++
- functiile nu trebuie declarate inainte de a fi folosite
- numele functiilor este “case-insensitive”
- un mare numar de functii cu utilitate directa in aplicatiile web exista in bibliotecile PHP
- unele biblioteci trebuie activate in momentul configurarii PHP
  - `extension=php_gd2.dll` (linia 639) // pentru functii de procesare grafica de exemplu
  - `extension=php_mysql.dll` (linia 651) // pentru functii de acces la baze de date MySql
  - `extension=php_mysqli.dll` (linia 652) // pentru functii de acces la baze de date MySql (**obligatoriu** de la PHP 5.6)

# Utilizarea functiilor PHP

- `<form action="rezultat.php" method="post">`

```
<p>Comanda receptionata la data:
```

```
<?php echo date('d/m/Y')." ora ".date('H:i');?></p>
```

**Magazin online XXX SRL**

**Rezultate comanda**

Comanda receptionata la data: 10/03/2010 ora 13:36

```
<body>
```

```
<h1>Magazin online XXX SRL</h1>
```

```
<h2>Rezultate comanda</h2>
```

```
<p>Comanda receptionata la data:
```

```
10/03/2010 ora 13:36</p>
```

```
</body>
```

# Elemente de control

- majoritatea notiunilor si sintaxei sunt similare celor din C/C++
- instructiune compusa: separata de acolade {...}
- if / else / elseif – executie conditionata

```
<?php
if ($a > $b) {
    echo "a mai mare ca b";
} elseif ($a == $b) {
    echo "a egal cu b";
} else {
    echo "a mai mic ca b";
}
?>
```

# Elemente de control

- while
- do-while
- for
- switch
- return
- break
- goto
  
- Similare cu echivalentele C/C++

```
$i = 1;  
while ($i <= 10) {  
    echo $i++;  
}
```

```
$i = 10;  
do {  
    echo $i--;  
} while ($i > 0);
```

```
for ($i = 1; $i <= 10; $i++) {  
    echo $i;  
}
```

```
switch ($i) {  
    case 0:  
        echo "i este 0";  
        break;  
    case 1:  
        echo "i este 1";  
        break;  
    default:  
        echo "i nici 1 nici 0";  
        break;  
}
```

# Elemente de control

- `include()`
- `require()`
- `include_once()`
- `require_once()`
  
- pentru inserarea **SI** evaluarea fisierului folosit ca parametru
- folosite pentru a nu multiplica sectiunile de cod comune
- `require` opreste executia script-ului curent daca fisierul parametru nu este gasit
- `..._once()` verifica daca respectivul fisier a mai fost introdus si **nu** il mai introduce inca o data

**Continuare**



# Laborator L3

- Sa se creeze un magazin simplu virtual care:
  - sa prezinte utilizatorului o lista de produse si preturi (constanta – maxim 5 produse)
  - sa preia de la acesta numarul de produse dorit
  - sa calculeze suma totala
  - sa adauge TVA **19%**
  - sa prezinte un raport care sa contina:
    - total de plata
    - ora comenzii

# Laborator L3 - continuare

- se creaza macar 3 pagini:
  - lista produse
  - formular comanda
  - rezultat
- forma paginilor:
  - tabel/CSS

culoare	<b>IMAGINE</b>	culoare
	<b>Continut</b> (cu alta culoare fundal)	

# Laborator – L3 - rezultat

## Magazin online Firma X SRL

### Lista Produse

Nr.	Produs	Pret
1	Carti	100
2	Caiete	50
3	Penare	150
4	Stilouri	125
5	Creioane	25

[Comanda](#)

## Magazin online Firma X SRL

### Realizati comanda

Nr.	Produs	Pret	Cantitate
1	Carti	100	<input type="text" value="1"/>
2	Caiete	50	<input type="text" value="2"/>
3	Penare	150	<input type="text" value="1"/>
4	Stilouri	125	<input type="text" value="0"/>
5	Creioane	25	<input type="text" value="0"/>

## Magazin online Firma X SRL

### Rezultate comanda

Pret total (fara TVA): 350

Pret total (cu TVA): 416.5

Comanda receptionata la data: 17/03/2010 ora 08:24

# Interactiunea cu utilizatorul

- Datele introduse de utilizator in forme se regasesc (in functie de metoda aleasa pentru forma) in una din variabilele:
  - `$_POST` – method="post"
  - `$_GET` – method="get"
  - `$_REQUEST` – ambele metode
- variabilele sunt **matrici** cu **cheia** data de atributul **name** din forma HTML
  - `<input type="text" name="carti_cant" size="3" maxlength="3" />`
  - `$_POST['carti_cant']` contine valoarea introdusa

# Laborator – L3 – sursa 1

```
<?php
define('PRET_CARTE',100);
define('PRET_CALET',50);
define('PRET_PENAR',150);
define('PRET_STILOU',125);
define('PRET_CREION',25);
?><h1>Magazin online Firma X SRL</h1>
<h2>Realizati comanda</h2>
<form action="rezultat.php" method="post">
<table border="0">
<tr bgcolor="#cccccc"><td>Nr.</td><td width="150">Produs</td><td width="50">Pret</td><td
width="15">Cantitate</td></tr>
<tr><td>1</td><td>Carti</td><td align="center"><?php echo PRET_CARTE;?></td><td align="center"><input
name="carte_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>2</td><td>Caiete</td><td align="center"><?php echo PRET_CALET;?></td><td align="center"><input
name="caiet_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>3</td><td>Penare</td><td align="center"><?php echo PRET_PENAR;?></td><td align="center"><input
name="penar_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>4</td><td>Stilouri</td><td align="center"><?php echo PRET_STILOU;?></td><td align="center"><input
name="stilou_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>5</td><td>Creioane</td><td align="center"><?php echo PRET_CREION;?></td><td align="center"><input
name="creion_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr>
<td colspan="4" align="center"><input type="submit" value="Trimite" /></td></tr>
</table>
</form>
```

# Laborator – L3 – sursa 2

```
<?php
define('PRET_CARTE',100);
define('PRET_CAIET',50);
define('PRET_PENAR',150);
define('PRET_STILOU',125);
define('PRET_CREION',25);
?><h1>Magazin online Firma X SRL</h1>
<h2>Rezultate comanda</h2>
<p>Pret total (fara TVA): <?php
$pret=$_POST['carte_cant']*PRET_CARTE+$_POST['caiet_cant']*PRET_CAIET
+$_POST['penar_cant']*PRET_PENAR+$_POST['stilou_cant']*PRET_STILOU+$_
POST['creion_cant']*PRET_CREION;
echo $pret;?></p>
<p>Pret total (cu TVA): <?php
$pret*=1.19;
echo $pret;?></p>
<p>Comanda receptionata la data: <?php echo date('d/m/Y')." ora
".date('H:i');?></p>
```

```
echo "<pre>";
print_r($_POST);
echo "</pre>";
```



# Includerea / controlul formei in fisierile PHP - Template



# Analiza critica

- design?
  - in aplicatiile web forma este importanta
  - nu trebuie sa fie inovativa ci familiara
  - "Don't make me think!"
- ~~■ capacitatea de extindere?~~
  - ~~■ mai multe produse~~
  - ~~■ schimbare de pret~~

# Template

- Sablon
- controlul simultan al formei pentru toate paginile din site
- separarea aplicatiei de forma

# Lista produse

**Magazin**      **Firma X SRL**

**Magazin online Firma X SRL**

**Lista Produse**

Nr.	Produs	Pret
1	Carti	100
2	Caiete	50
3	Penare	150
4	Stilouri	125
5	Creioane	25
<a href="#">Comanda</a>		

# Elemente de control

- `include()`
- `require()`
- `include_once()`
- `require_once()`
  
- pentru inserarea **SI** evaluarea fisierului folosit ca parametru
- folosite pentru a nu multiplica sectiunile de cod comune
- **require** opreste executia script-ului curent daca fisierul parametru **nu** este gasit
- **...\_once()** verifica daca respectivul fisier a mai fost introdus si **nu** il mai introduce inca o data

# Exemplu – design 2

- sectiunile repetabile pot fi mutate intr-un fisier separat si introduse cu require()
- se identifica zonele comune

```
<html>
<head>
<title>Magazin online Firma X SRL</title>
</head>
<body bgcolor="#CCFFFF">
<table width="600" border="0" align="center">
<tr><td></td></tr>
<tr><td height="600" valign="top"
bgcolor="#FFFFCC">
Continut
</td></tr>
</table>
</body>
</html>
```

# Lista produse

antet.php

```
<html>
<head>
<title>Magazin online Firma X
SRL</title>
</head>
<body bgcolor="#CCFFFF"><?php
define('PRET_CARTE',100);
define('PRET_CAIET',50);
define('PRET_PENAR',150);
define('PRET_STILOU',125);
define('PRET_CREION',25);
//orice cod comun PHP
?><table width="600" border="0"
align="center">
<tr><td></td></tr>
<tr><td height="600" valign="top"
bgcolor="#FFFFCC">
<h1>Magazin online Firma X SRL</h1>
```

subsol.php

```
</td></tr>
</table>
</body>
</html>
```

\*.php

```
<?php require('antet.php');?>
<h2>Lista Produse</h2>
<table border="1">
...
</table>
<?php require('subsol.php');?>
```

# Lista produse/template

**Magazin**      **Firma X SRL**

**Magazin online Firma X SRL**

**Lista Produse**

Nr.	Produs	Pret
1	Carti	100
2	Caiete	50
3	Penare	150
4	Stilouri	125
5	Creioane	25

[Comanda](#)

# Avantajul lucrului cu sabloane

- viteza de dezvoltare a aplicatiei
- separare clara a formei de aplicatie
- forma unitara
  - “don’t make me think”
- modificarea simultana a formei pentru toate paginile din site
- posibilitatea definirii datelor comune intr-un singur fisier
  - `define('PRET_CARTE',100);`



# Variabile tip string

# PHP – tipuri de date

- scalar
  - boolean
  - integer
  - float (double)
  - **string**
- compus
  - array
  - object
- special
  - resource
  - NULL

# Variabile tip string

- Scopul final al PHP e popularea cu date (sub forma de text) a campurilor existente intr-un schelet HTML
- Ca urmare datele de tip sir de caractere (string) sunt tratate mai complex decat echivalentul C/C++
  - mai multe modalitati de definire
  - mai multe modalitati de interpretare
  - **mult** mai multe functii

# Variabile tip string

- definire variabila de tip string
  - utilizare apostrof ` `
  - utilizare ghilimele " "
  - definiri tip bloc
    - heredoc <<< "X"
    - nowdoc <<<'X' (PHP>5.3.0)

# Variabile tip string ` `

- apostroful ` ` e utilizat pentru definirea sirurilor primare de caractere
  - se defineste o suita de caractere
  - prelucrarile in interiorul sirului sunt reduse
    - \' reprezinta caracterul apostrof
    - \\ si \ reprezinta caracterul backslash
    - doar atat!!!

# Variabile tip string ""

- ghilimelele "" sunt utilizate pentru definirea sirurilor de caractere complexe
  - prelucrarile in interiorul sirului sunt mai complexe decat echivalentul C/C++
    - caracterele ASCII speciale, identic cu C++: \n, \r, \t, \\", \v, \e, \f, \x, \u
    - \" caracterul ghilimele
    - \\$ caracterul \$
    - se interpreteaza **variabile** in interiorul sirului !!!

# Variabile tip string “ ”

- caracterul \$ indica faptul ca urmeaza un nume de variabila
  - interpretorul foloseste toate caracterele care pot genera nume de variabile valide (\$x, \$x->y, \$x[y])
  - daca e nevoie de exprimare mai complexa a variabilelor (de exemplu matrici cu 2 indici x[y][z] sau cu indici neintregi) se foloseste sintaxa complexa: **{ }**

# Variabile tip string " "

- sintaxa **simpla** pentru interpretarea variabilelor in interiorul sirurilor

```
<?php
$juice = "apple";

echo "He drank some $juice juice.";
// He drank some apple juice.
echo "He drank some juice made of $juices.";
// He drank some juice made of . //s character valid pentru variabile

?>
```



# Variabile tip string ""

- sintaxa **simpla** pentru interpretarea variabilelor in interiorul sirurilor

```
<?php
$juices = array("apple", "orange", "koolaid1" => "purple");
class people {
    public $john = "John Smith";
}

$people = new people();
echo "$people->john drank some $juices[o] juice.";
// John Smith drank some apple juice.
?>
```

# Variabile tip string " "

- sintaxa **complexa** pentru interpretarea variabilelor in interiorul sirurilor **{ }**

```
<?php
$juice = "apple";

echo "He drank some juice made of $juices.";
// He drank some juice made of . //s caracter valid pentru variabile
echo "He drank some juice made of ${juice}s."
// He drank some juice made of apples. // {} arata unde se incheie
numele variabilei
?>
```

# Variabile tip string " "

- sintaxa **complexa** pentru interpretarea variabilelor in interiorul sirurilor **{ }**

```
<?php
$juices = array(array("apple", "orange"), "koolaid1" => "purple");
class people {
    public $name = "John Smith";
}

$obj->values[3] = new people();
echo "$obj->values[3]->name drank some $juices[0][1] juice.";
// drank some juice.
echo "{ $obj->values[3]->name } drank some { $juices[0][1] } juice.";
// John Smith drank some apple juice.
?>
```

# Variabile globale

# Variabile globale

- Variabilele globale (predefinite)
  - accesibile script-urilor PHP prin conlucrarea cu server-ul
  - Exemple:
    - `$_SERVER` — Server and execution environment information
    - `$_GET` — HTTP GET variables
    - `$_POST` — HTTP POST variables
    - `$_FILES` — HTTP File Upload variables
    - `$_REQUEST` — HTTP Request variables
    - `$_SESSION` — Session variables
    - `$_ENV` — Environment variables
    - `$_COOKIE` — HTTP Cookies

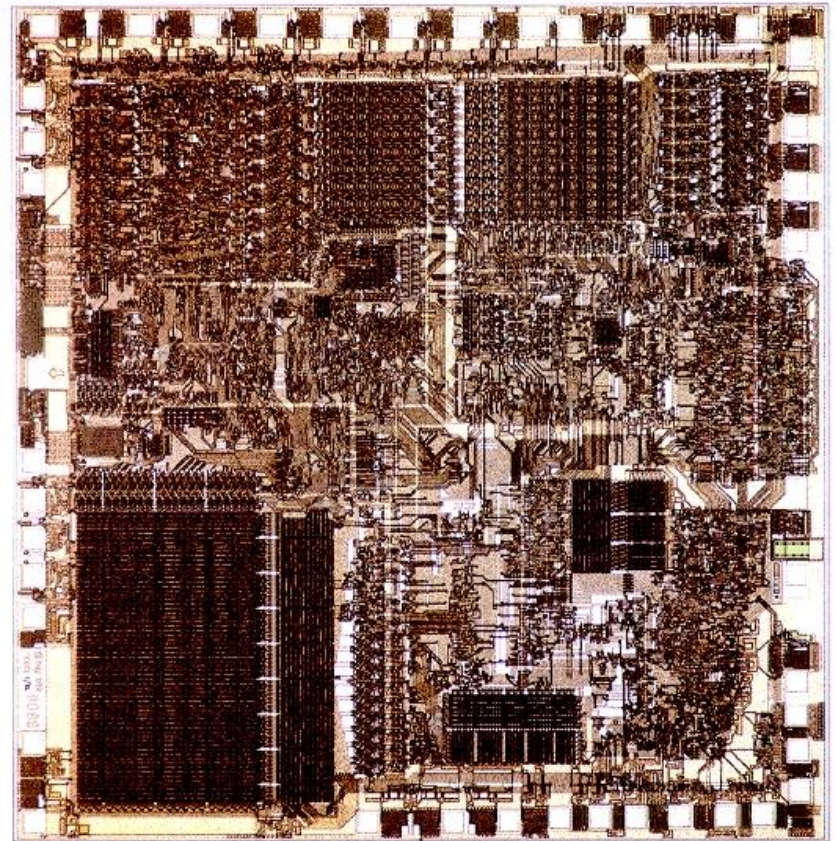
# Interactiunea cu utilizatorul

- Datele introduse de utilizator in forme se regasesc (in functie de metoda aleasa pentru forma) in una din variabilele:
  - `$_POST` – method="post"
  - `$_GET` – method="get"
  - `$_REQUEST` – ambele metode
- variabilele sunt **matrici** cu **cheia** data de atributul **name** din forma HTML
  - `<input type="text" name="carti_cant" size="3" maxlength="3" />`
  - `$_POST['carti_cant']` contine valoarea introdusa

# Structuri repetitive – tablouri

# Impresionant?

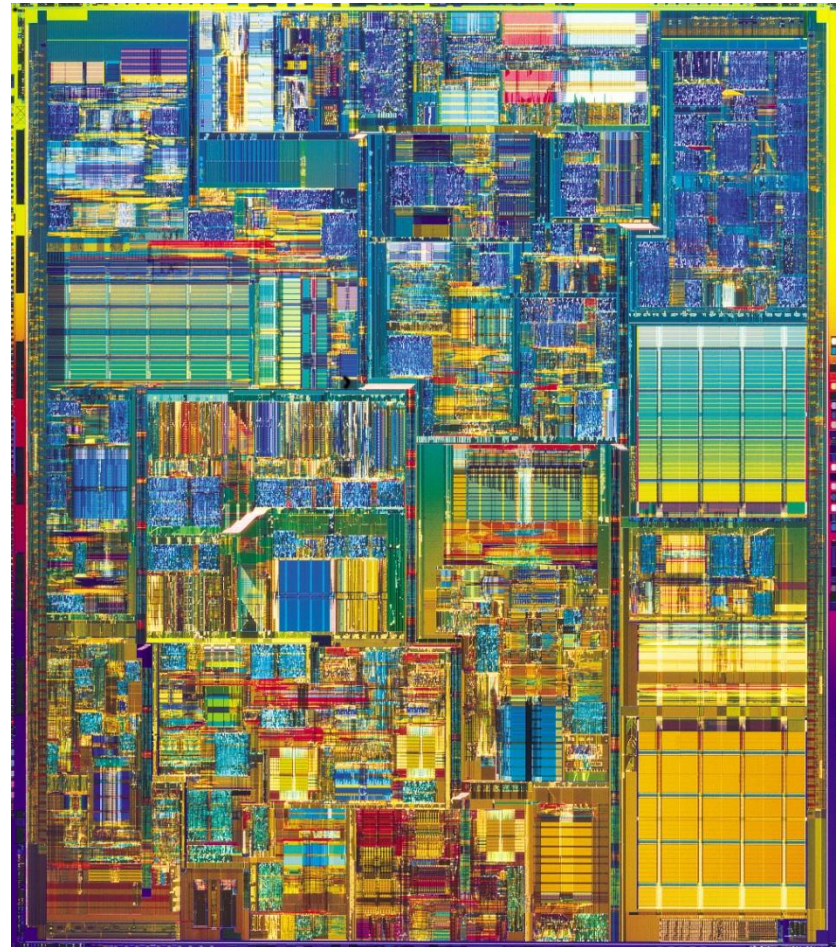
- Intel® 8086
- 29.000 tranzistoare pe CPU
- 1978
- 1 MB date
- 4.7 MHz





# Impresionant?

- Intel® Itanium® processors (Tukwila)
- 2009
- 2 miliarde tranzistoare pe CPU
- 16 EB date (16 G GB)
- > 3 GHz



# Concepte

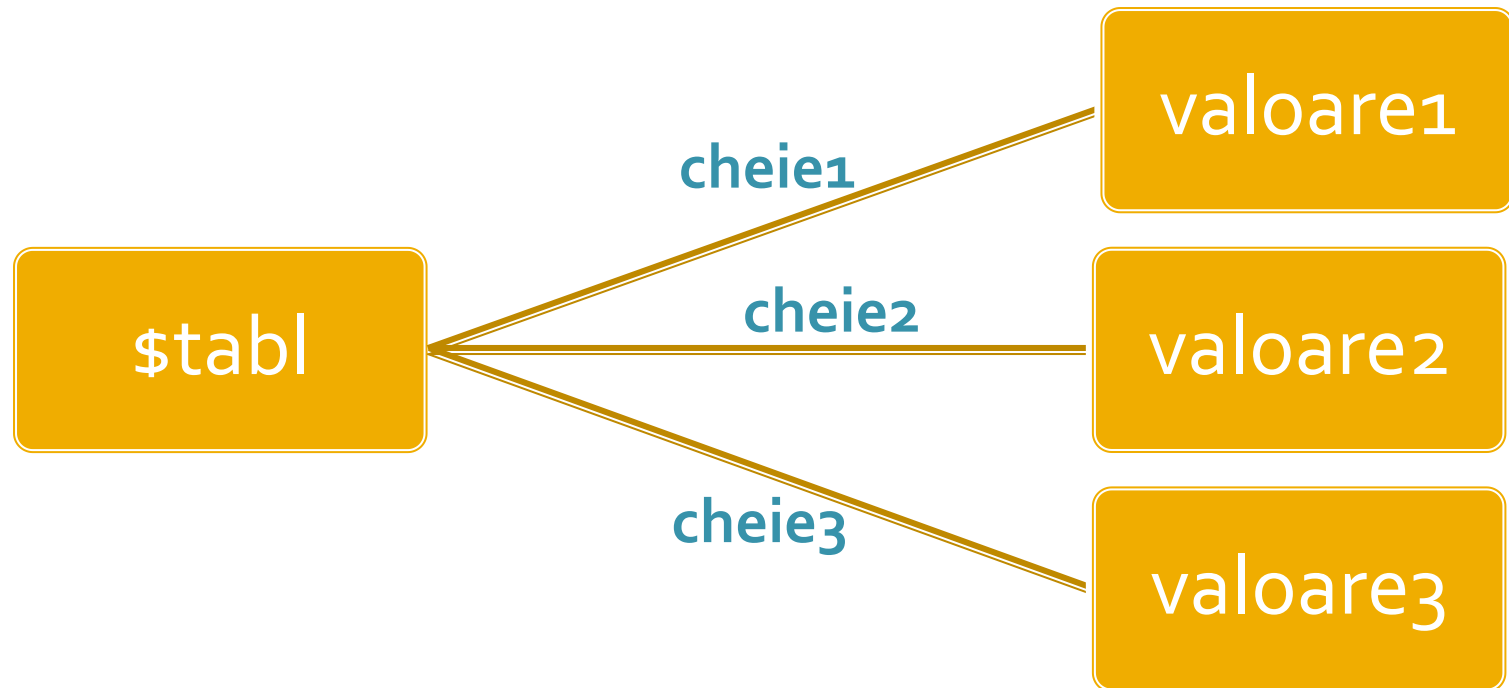
- Efectuare foarte rapida a unui numar **mic** de instructiuni, de **complexitate redusa**, repetate de un numar foarte mare de ori
- Programare: coborarea rationamentului la nivelul de **complexitate redusa**, cu obtinerea performantei prin structuri repetitive simple efectuate rapid.
- Operatii repetitive / date repetitive

# Tablouri in PHP

- tabloul este tipul de variabila care asociaza **valori** unor **chei**
- spre deosebire de C, Basic, **cheile nu sunt** obligatoriu numere **intregi**, pot fi si **siruri**
- implicit cheile sunt intregi succesivi (pentru fiecare element adaugat) si primul element este 0.
- definirea unei perechi cheie / valoare
  - cheie => valoare
- definirea unui tablou
  - `$matr = array("definirea perechilor chei/valori")`
  - pereche: cheie => valoare, ...

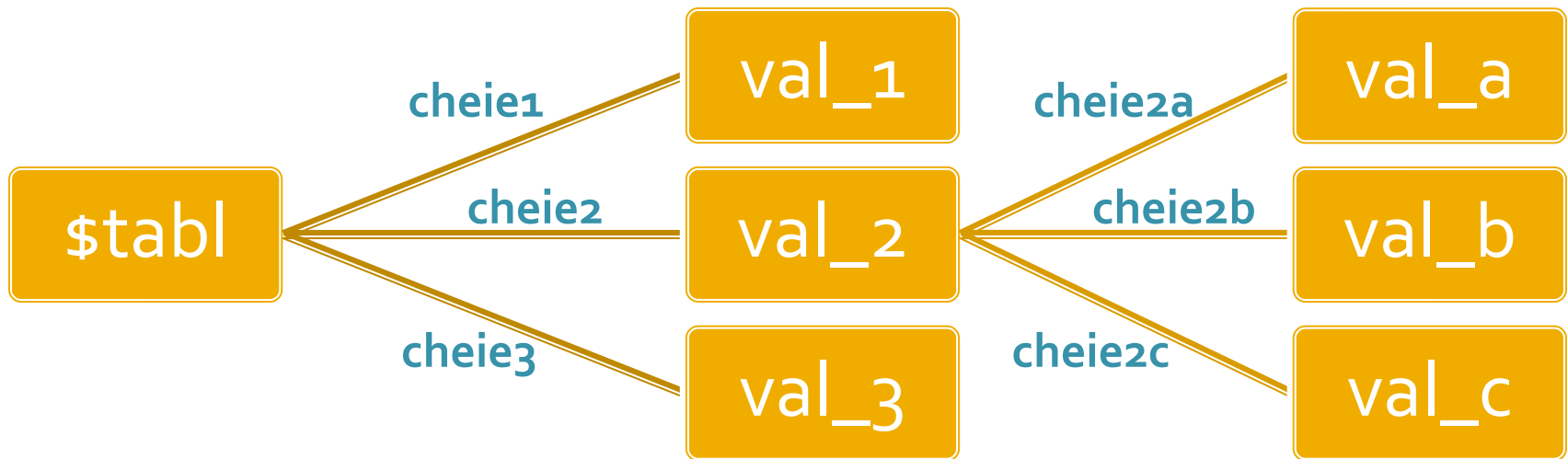
# Tablou = Arbore

- \$tabl = array(cheie1 => valoarea1, cheie2 => valoarea2, cheie3 => valoarea3)



# Tablou = Arbore

- In particular, una sau mai multe dintre din valori poate fi la randul ei tablou, ducand la **ramificarea** arborelui
- `$tabl = array(cheie1 => val_1, cheie2 => array(cheie2a => val_a, cheie2b => val_b, cheie2c => val_c), cheie3 => val_3)`



# Tablouri in PHP

```
$matr = array(1, 2, 3, 4, 5);
```

```
$matr[0]=1
```

```
$matr[1]=2
```

```
$matr[2]=3
```

```
$matr[3]=4
```

```
$matr[4]=5
```

```
$matr = array('a' => 1, 'b' => 2, 3, 4, 5);
```

```
$matr['a']=1
```

```
$matr['b']=2
```

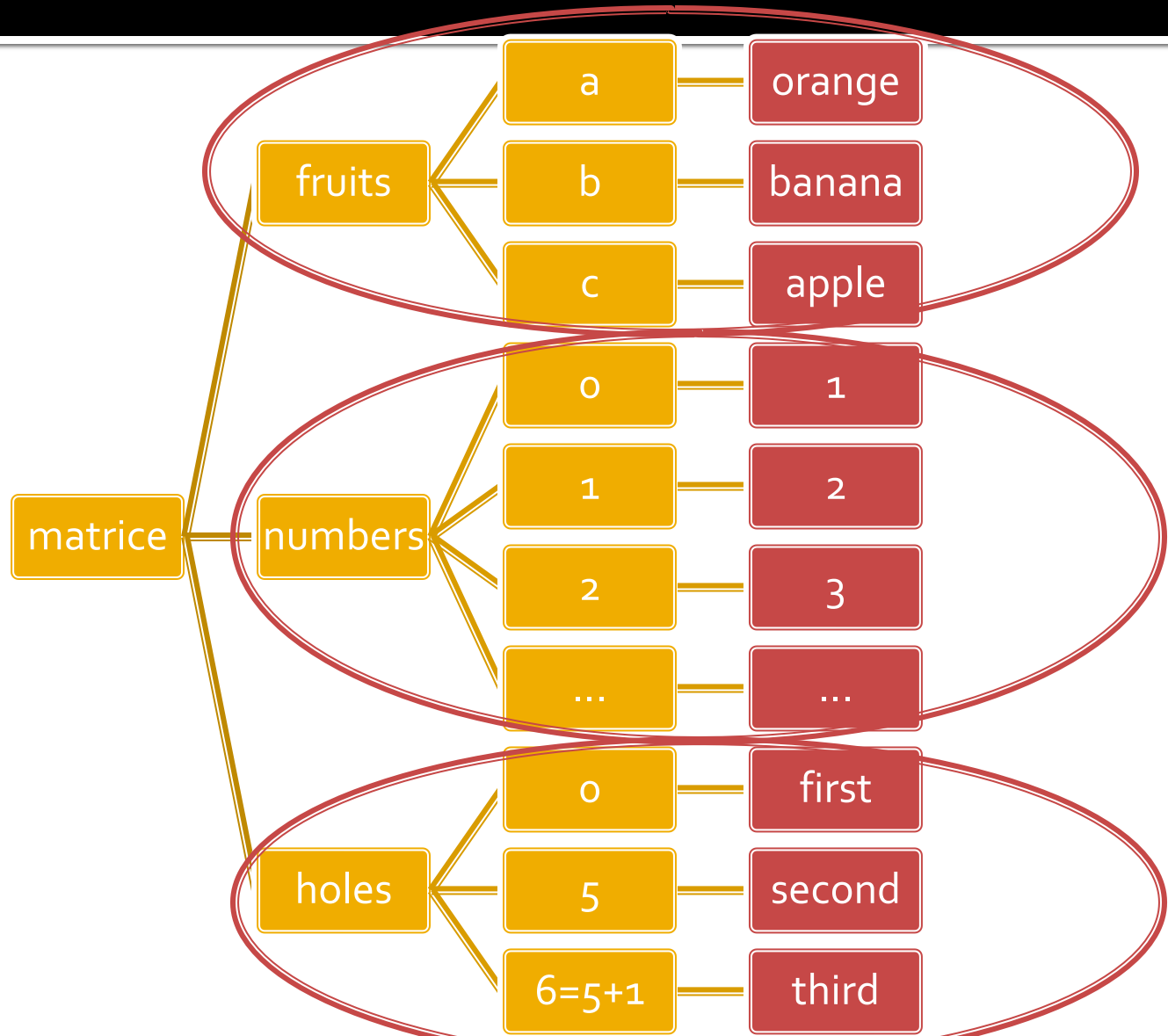
```
$matr[0]=3
```

```
$matr[1]=4
```

```
$matr[2]=5
```

```
$matrice= array (  
    "fruits" => array("a" => "orange", "b" => "banana", "c" => "apple"),  
    "numbers" => array(1, 2, 3, 4, 5, 6),  
    "holes" => array("first", 5 => "second", "third")  
);
```

# Tablou = arbore



# Afisarea tablourilor

```
echo "<pre>";  
print_r ($matr);  
echo "</pre>";
```

```
$matr= array (  
"fruits" =>  
array("a" => "orange", "b" => "banana", "c" => "apple",  
"ultim"),  
"numbers" =>  
array(1, 2, 3, 4, 5, 6),  
"holes" =>  
array("first", 5 => "second", "third")  
);  
echo $matr;  
echo "<pre>";  
print_r ($matr);  
echo "</pre>";
```

```
Array  
  
Array  
(  
    [fruits] => Array  
        (  
            [a] => orange  
            [b] => banana  
            [c] => apple  
            [0] => ultim  
        )  
    [numbers] => Array  
        (  
            [0] => 1  
            [1] => 2  
            [2] => 3  
            [3] => 4  
            [4] => 5  
            [5] => 6  
        )  
    [holes] => Array  
        (  
            [0] => first  
            [5] => second  
            [6] => third  
        )  
)
```



# Chei

- Chei numerice
  - implicite
  - similare celorlalte limbaje de programare
  - dificil de utilizat (trebuie retinuta valoarea logica a unei anumite chei numerice)
- Chei sir
  - claritate mai mare
  - eficienta numerica mai mica
  - tablourile au un index numeric intern, implicit ascuns, accesibil prin functii :  
**index => cheie => valoare**

# Elemente de control

- `for` – util dacă la definirea tablourilor sunt folosite cheile numerice implicite (numere întregi)
- `do ... while` și `while` se pot folosi împreună cu funcții specifice caracteristice tablourilor `next()`, `prev()`, `end()`, `reset()`, `current()`, `each()`
- `foreach` - elementul de control al iteratiilor cel mai potrivit pentru chei alfanumerice

# Elemente de control – foreach

- `foreach (array_expression as $key => $value) statement`
- `foreach (array_expression as $value) statement`
- iterarea prin fiecare element al tabloului
- la fiecare element variabila declarata in instructiune variabila locala `$key` ofera acces la cheia curenta iar variabila locala `$value` ofera acces la valoarea asociata
- `foreach()` lucreaza cu o **copie** a tabloului deci tabloul original nu va fi modificat prin schimbarea continutului variabilelor `$key` si `$value`

# Elemente de control – foreach

```
$matr = array (  
    "fruits" => array("a" => "orange", "b" => "banana", "c" => "apple", "ultim"),  
    "numbers" => "in loc de numere",  
    "holes" => "in loc de ce era"  
);  
foreach ($matr as $scheie => $continut)  
    echo "matr[".$scheie."]=".$continut."<br />";
```

```
matr[fruits]=Array  
matr[numbers]=in loc de numere  
matr[holes]=in loc de ce era
```

# Tablouri – functii utile

- `current($matr)` – returneaza elementul indicat de indicele intern al tabloului (`~v[i]`)
- `next($matr)` – incrementeaza indicele intern si returneaza valoarea stocata acolo (`~v[++i]`)
- `prev($matr)` – decrementeaza indicele intern si returneaza valoarea stocata acolo (`~v[--i]`)
- `end($matr)` – muta indicele intern la ultimul element si returneaza valoarea stocata acolo (`~i=N-1;v[i]`)
- `reset($matr)` – muta indicele intern la primul element si returneaza valoarea stocata acolo (`~i=0;v[i]`)

# Tablouri – functii utile

- `sort($matr)` – ordoneaza in ordine crescatoare a **valorilor** un tablou, cheile sunt sterse si recreate
  - `$fruits = array("lemon", "orange", "banana", "apple");`  
`sort($fruits);`
  - `fruits[0] = apple, fruits[1] = banana, fruits[2] = lemon, fruits[3] = orange`
- `rsort($matr)` – similar, descrescator

# Tablouri – functii utile

- `asort($matr)` ordoneaza in ordine crescatoare a **valorilor** un tablou, cheile sunt pastrate, inclusiv asocierea cheie => valoare
  - `$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" => "apple");`  
`asort($fruits);`
  - `c = apple, b = banana, d = lemon, a = orange`
- `arsort($matr)` – similar, descrescator

# Tablouri – functii utile

- `ksort($matr)` ordoneaza in ordine crescatoare a **cheilor** un tablou, cheile sunt pastrate, inclusiv asocierea cheie => valoare
  - `$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" => "apple");`  
`ksort($fruits);`
  - a = orange, b = banana, c = apple , d = lemon
- `krsort($matr)` – similar, descrescator



# Exemplu utilizare tablouri

# Analiza critica

- ~~design?~~
  - ~~in aplicatiile web forma este importanta~~
  - ~~nu trebuie sa fie inovativa ci familiara~~
  - ~~"Don't make me think!"~~
- capacitatea de extindere?
  - mai multe produse
  - schimbare de pret

# Exemplu

- In exemplul anterior utilizarea tablourilor va aduce urmatoarele avantaje:
  - codul va fi mai concis
  - codul va fi mai general (valabil si pentru 5 produse si pentru 1000)
  - scalabilitate crescuta (se pot adauga usor produse)

# Exemplu

- fiecare produs e caracterizat de:
  - nume
  - pret
  - (eventual) descriere
  - cantitate comandata
- putem folosi unul din attribute ca si cheie (numele in exemplu)
- se poate controla (prin atributul name = "") structura variabilei globale `$_POST`

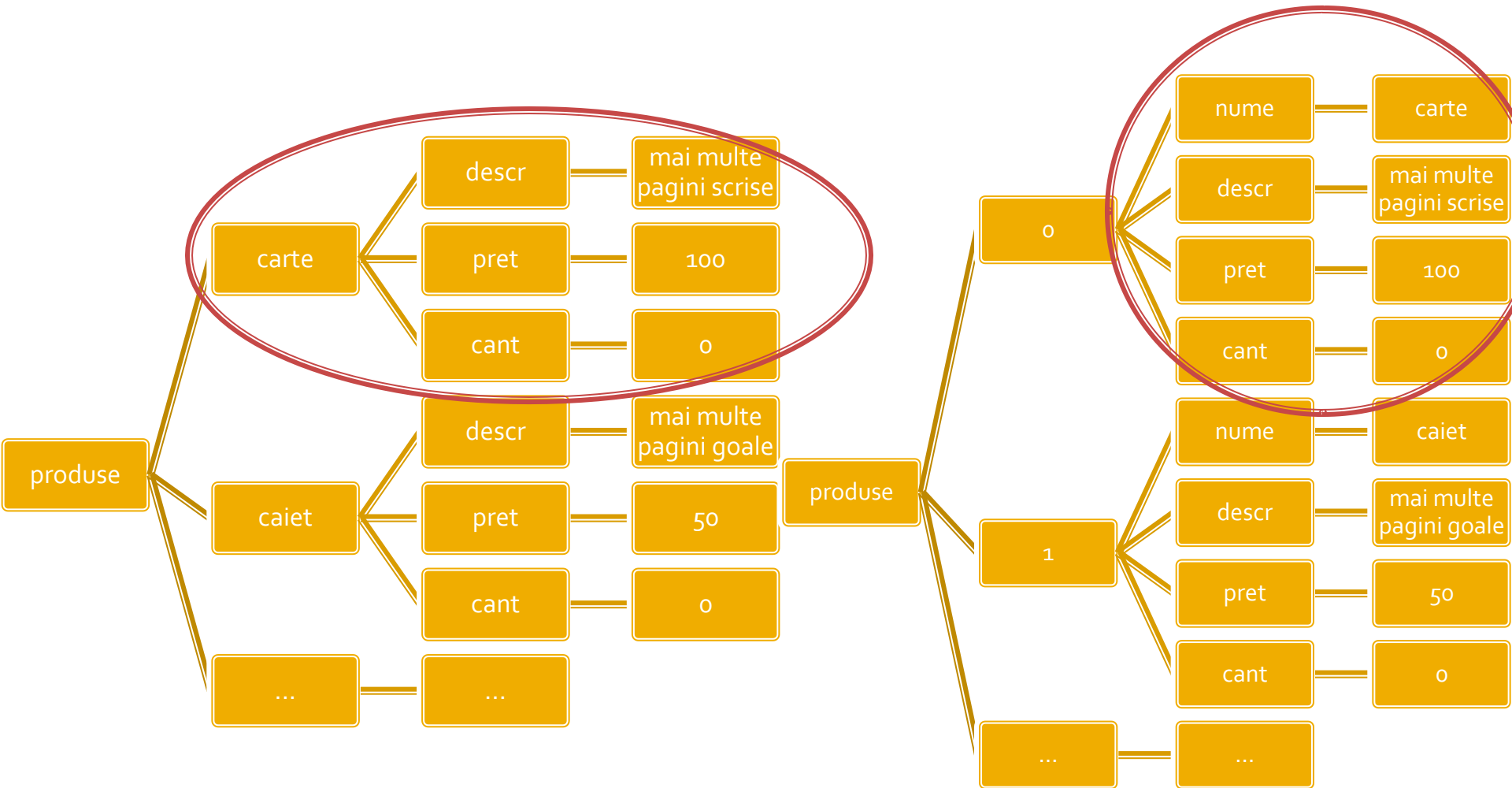
# Tablou produse

- una din structurile posibile

```
$produse = array ( 'carte' => array ("descr" => "mai multe pagini scrise", "pret" => 100, "cant" => 0),  
                  'caiet' => array ("descr" => "mai multe pagini goale", "pret" => 50, "cant" => 0),  
                  'penar' => array ("descr" => "loc de depozitat instrumente", "pret" => 150, "cant" => 0),  
                  'stilou' => array ("descr" => "instrument de scris albastru", "pret" => 125, "cant" => 0),  
                  'creion' => array ("descr" => "instrument de scris gri", "pret" => 25, "cant" => 0)  
                );
```

- se urmareste obtinerea unei structuri clare
  - usor de modificat/adaugat date
  - usor de utilizat
- daca definitia se introduce in fisierul antet va fi accesibila in toate fisierele individuale

# Tablou produse



# Laborator – L3 – sursa 1

```
<?php
define('PRET_CARTE',100);
define('PRET_CALET',50);
define('PRET_PENAR',150);
define('PRET_STILOU',125);
define('PRET_CREION',25);
?><h1>Magazin online Firma X SRL</h1>
<h2>Realizati comanda</h2>
<form action="rezultat.php" method="post">
<table border="0">
<tr bgcolor="#cccccc"><td>Nr.</td><td width="150">Produs</td><td width="50">Pret</td><td
width="15">Cantitate</td></tr>
<tr><td>1</td><td>Carti</td><td align="center"><?php echo PRET_CARTE;?></td><td align="center"><input
name="carte_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>2</td><td>Caiete</td><td align="center"><?php echo PRET_CALET;?></td><td align="center"><input
name="caiet_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>3</td><td>Penare</td><td align="center"><?php echo PRET_PENAR;?></td><td align="center"><input
name="penar_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>4</td><td>Stilouri</td><td align="center"><?php echo PRET_STILOU;?></td><td align="center"><input
name="stilou_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr><td>5</td><td>Creioane</td><td align="center"><?php echo PRET_CREION;?></td><td align="center"><input
name="creion_cant" type="text" value="0" size="3" maxlength="3" /></td></tr>
<tr>
<td colspan="4" align="center"><input type="submit" value="Trimite" /></td></tr>
</table>
</form>
```

# Crearea listei de produse

```
<?php require('antet.php');?>
<h2>Lista Produse</h2>
<table border="1">
<tr bgcolor="#cccccc"><td>Nr.</td><td width="150">Produs</td><td width="150">Descriere</td><td
width="50">Pret</td></tr>
<?php
$index=1;
foreach ($produse as $prod => $detalii)
{
    echo "<tr><td>".$index."</td><td>".ucfirst(strtolower($prod))."</td><td>".$detalii['descr']."</td><td
align='center'">".$detalii['pret']."</td></tr>";
    $index++;
}
?>
<?php
$index=1;
foreach ($produse as $prod => $detalii)
{?>
<tr><td><?php echo $index;?></td><td><?php echo ucfirst(strtolower($prod));?></td><td><?php echo
$detalii['descr'];?></td><td align="center"><?php echo $detalii['pret'];?></td></tr>
<?php $index++;
} ?>
<tr><td colspan="4" align="center"><a href="formular.php">Comanda</a></td></tr></table>
<?php require('subsol.php');?>
```





# Crearea listei de produse

**Magazin**

**Firma X SRL**

## Magazin online Firma X SRL

### Lista Produse

Nr.	Produs	Descriere	Pret
1	Carte	mai multe pagini scrise legate	100
2	Caiet	mai multe pagini goale legate	50
3	Penar	loc de depozitat instrumente de scris	150
4	Stilou	instrument de scris albastru	125
5	Creion	instrument de scris gri	25
1	Carte	mai multe pagini scrise legate	100
2	Caiet	mai multe pagini goale legate	50
3	Penar	loc de depozitat instrumente de scris	150
4	Stilou	instrument de scris albastru	125
5	Creion	instrument de scris gri	25
<a href="#">Comanda</a>			

# Subdivizare \$\_POST

- atributul **name** in forma devine **cheie** in tabloul global \$\_POST
  - `<input type="text" name="carti_cant" size="3" maxlength="3" />`
  - `$_POST['carti_cant']` contine valoarea introdusa
- realizand atributul **name** ca tablou, se obtine in \$\_POST un "subtablou" (ramificare locala a arborelui) care grupeaza elementele input
  - `<input type="text" name="cant[carti]" size="3" maxlength="3" />`
  - `$_POST ['cant'] ['carti']` contine valoarea introdusa

# Subdivizare \$\_POST

- realizand atributul `name` ca tablou, se obtine in `$_POST` un "subtablou" (ramificare locala a arborelui) care grupeaza elementele dorite
  - `<input type="text" name="cant[carti]" size="3" maxlength="3" />`
  - `$_POST ['cant'] ['carti']` contine valoarea introdusa
- Este necesar pentru a grupa elementele similare pe care sa le prelucram la receptie cu `foreach`
- `$_POST` contine si alte elemente pe care le dorim eventual tratate separat
  - numele (`name`) si valoarea butonului "submit" apar in `$_POST` de exemplu

# Crearea formei de comanda

```
<?php require('antet.php');?>
<h2>Realizati comanda</h2>
<form action="rezultat.php" method="post">
<table border="0">
<tr bgcolor="#cccccc"><td>Nr.</td><td width="150">Produs</td><td width="50">Pret</td><td
width="15">Cantitate</td></tr>
<?php
$index=1;
foreach ($produse as $prod => $detalii)
    {?>
<tr><td><?php echo $index;?></td><td><?php echo ucfirst(strtolower($prod));?></td><td
align="center"><?php echo $detalii['pret'];?></td><td><input name="<?php echo
"cant[".$prod."];?>" type="text" value="0" size="3" maxlength="3" /></td></tr>
<?php $index++;
    }?>
<tr><td colspan="4" align="center"><input type="submit" value="Trimite" /></td></tr>
</table>
</form>
<?php require('subsol.php');?>
```

# Crearea listei de produse

**Magazin** **Firma X SRL**

**Magazin online Firma X SRL**

**Realizati comanda**

Nr.	Produs	Pret	Cantitate
1	Carte	100	<input type="text" value="0"/>
2	Caiet	50	<input type="text" value="0"/>
3	Penar	150	<input type="text" value="0"/>
4	Stilou	125	<input type="text" value="0"/>
5	Creion	25	<input type="text" value="0"/>

Trimite

# Prelucrarea comenzii

```
<?php require('antet.php');?>
<h2>Rezultate comanda</h2>
<p>Pret total (fara TVA):
<?php
$pret=0;
$afis="";
foreach ($_POST['cant'] as $prod => $cant)
    {
        $pret += $cant*$produse[$prod]['pret'];
        $afis .= "+".$cant."x".$produse[$prod]['pret'];
    }
echo $pret;
?>
<p>Obtinut astfel: <?php echo $afis;?></p>
<p>Pret total (cu TVA): <?php echo $pret*1.19;?></p>
<p><?php
echo "<pre>";
print_r ($_POST);
echo "</pre>";
?>
</p>
<p>Comanda receptionata la data: <?php echo date('d/m/Y')." ora ".date('H:i');?></p>
<?php require('subsol.php');?>
```

# Prelucrarea comenzii

**Magazin**

**Firma X SRL**

## Magazin online Firma X SRL

### Rezultate comanda

Pret total (fara TVA): 600

Obtinut astfel: +2x100+2x50+2x150+0x125+0x25

Pret total (cu TVA): 714

```
Array
(
    [cant] => Array
        (
            [carte] => 2
            [caiet] => 2
            [penar] => 2
            [stilou] => 0
            [creion] => 0
        )
)
```

Comanda receptionata la data: 17/03/2010 ora 13:55

# Memorarea datelor



# Scrierea datelor pe disc

- Pentru a oferi posibilitatea beneficiarului aplicatiei (vanzator) sa poata adauga/sterge/modifica produse
  - din interfata browser
  - fara sa aiba cunostinte de programare
- E necesar ca tabloul **\$produse** sa fie creat in timpul rularii plecand de la un suport extern de date, accesibil pentru scriere vanzatorului
- Ulterior se va implementa aplicatia ce utilizeaza baze de date – momentan se vor scrie datele pe disc

# Utilizarea fisierelor – Functii

- `pointer = fopen(cale,mod)` deschide un fisier pentru operatii descrise de "mod". Se returneaza un pointer spre fisier de tip resursa care va fi folosit la operatiile urmatoare
- `fwrite(pointer,date)` – scrie datele in fisier (date – de tip string)
- `string = fread(pointer,cantitate)` citeste "cantitate" octeti din fisier
- `$matr = file(cale)` deschide fisierul identificat cu "cale" si citeste fiecare linie (incluzand \n) intr-un element distinct in matrice. \$matr de tip array, matrice de siruri

# Crearea fisierului

```
$produse = array ( 'carte' => array ("descr" => "mai multe pagini scrise legate", "pret" => 100, "cant" => 0),  
    'caiet' => array ("descr" => "mai multe pagini goale legate", "pret" => 50, "cant" => 0),  
    'penar' => array ("descr" => "loc de depozitat instrumente de scris", "pret" => 150, "cant" => 0),  
    'stilou' => array ("descr" => "instrument de scris albastru", "pret" => 125, "cant" => 0),  
    'creion' => array ("descr" => "instrument de scris gri", "pret" => 25, "cant" => 0)  
    );  
  
$handle = fopen("produse.txt", "wb");  
foreach ($produse as $prod => $detalii)  
    fwrite($handle,$prod."\t".$detalii['descr']."\t".$detalii['pret']."\t"  
.$detalii['cant']."\r\n");
```

# Crearea fisierului

- crearea initiala se poate face prin modificarea o singura data a fisierului antet.php existent astfel incat sa scrie datele pe disc

```
$produse = array ( 'carte' => array ("descr" => "mai multe pagini scrise", "pret" => 100, "cant" => 0),  
                  'caiet' => array ("descr" => "mai multe pagini goale", "pret" => 50, "cant" => 0),  
                  'penar' => array ("descr" => "loc de depozitat instrumente", "pret" => 150, "cant" => 0),  
                  'stilou' => array ("descr" => "instrument de scris albastru", "pret" => 125, "cant" => 0),  
                  'creion' => array ("descr" => "instrument de scris gri", "pret" => 25, "cant" => 0)  
                );  
$handle = fopen("produse.txt", "wb");  
foreach ($produse as $prod => $detalii)  
    fwrite($handle,$prod."\t".$detalii['descr']."\t".$detalii['pret']."\t".$detalii['cant']."\r\n");
```

# Citirea fisierului pentru crearea matricii

```
$matr=file("produse.txt");  
echo "<pre>";  
print_r ($matr);  
echo "</pre>";  
foreach ($matr as $linie)  
{  
    $valori=explode("\t",$linie,4);  
    $produse[$valori[0]]=array ("descr" => $valori[1], "pret" => $valori[2], "cant" => $valori[3]);  
}
```

```
Array  
(  
    [0] => carte          mai multe pagini scrise legate  100    0  
    [1] => caiet         mai multe pagini goale legate   50     0  
    [2] => penar         loc de depozitat instrumente de scris 150    0  
    [3] => stilou       instrument de scris albastru    125    0  
    [4] => creion       instrument de scris gri 25      0
```

# produse.txt

- se pot utiliza si alte caractere pentru separare
  - esential: sa nu apara in date
  - TAB are efect vizual si in fisiere text

carte	mai multe pagini scrise legate	100	0
caiet	mai multe pagini goale legate	75	0
penar	loc de depozitat instrumente de scris	150	0
stilou	instrument de scris albastru	125	0
creion	instrument de scris gri	25	0

# Laborator 4

# Laborator 4

- Sa se continue magazinul virtual:
  - pentru usurinta modificarilor ulterioare aplicatia lucreaza cu matrici
  - sa prezinte utilizatorului o lista de produse si preturi
  - lista de produse si preturi se citeste dintr-un **fișier**
  - se preia comanda si se calculeaza suma totala
- Optional
  - se creaza o pagina prin care vanzatorul poate **modifica** preturile si produsele

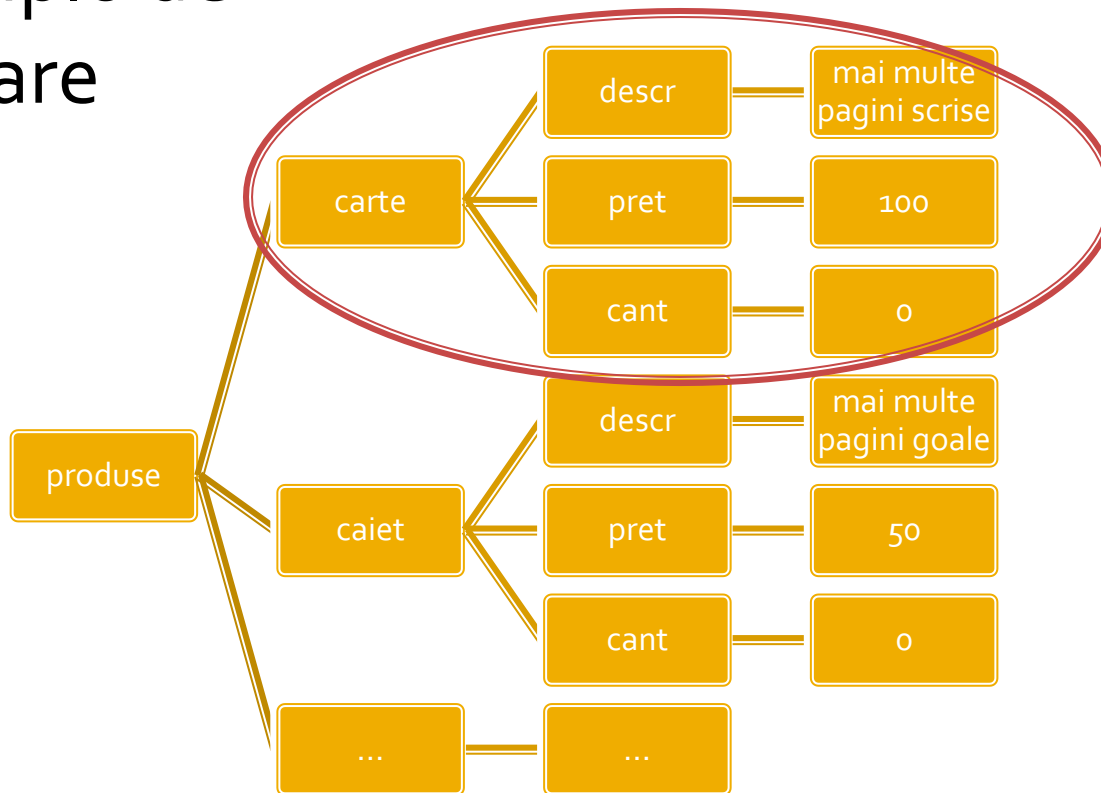


# Laborator 4 – Mod de lucru

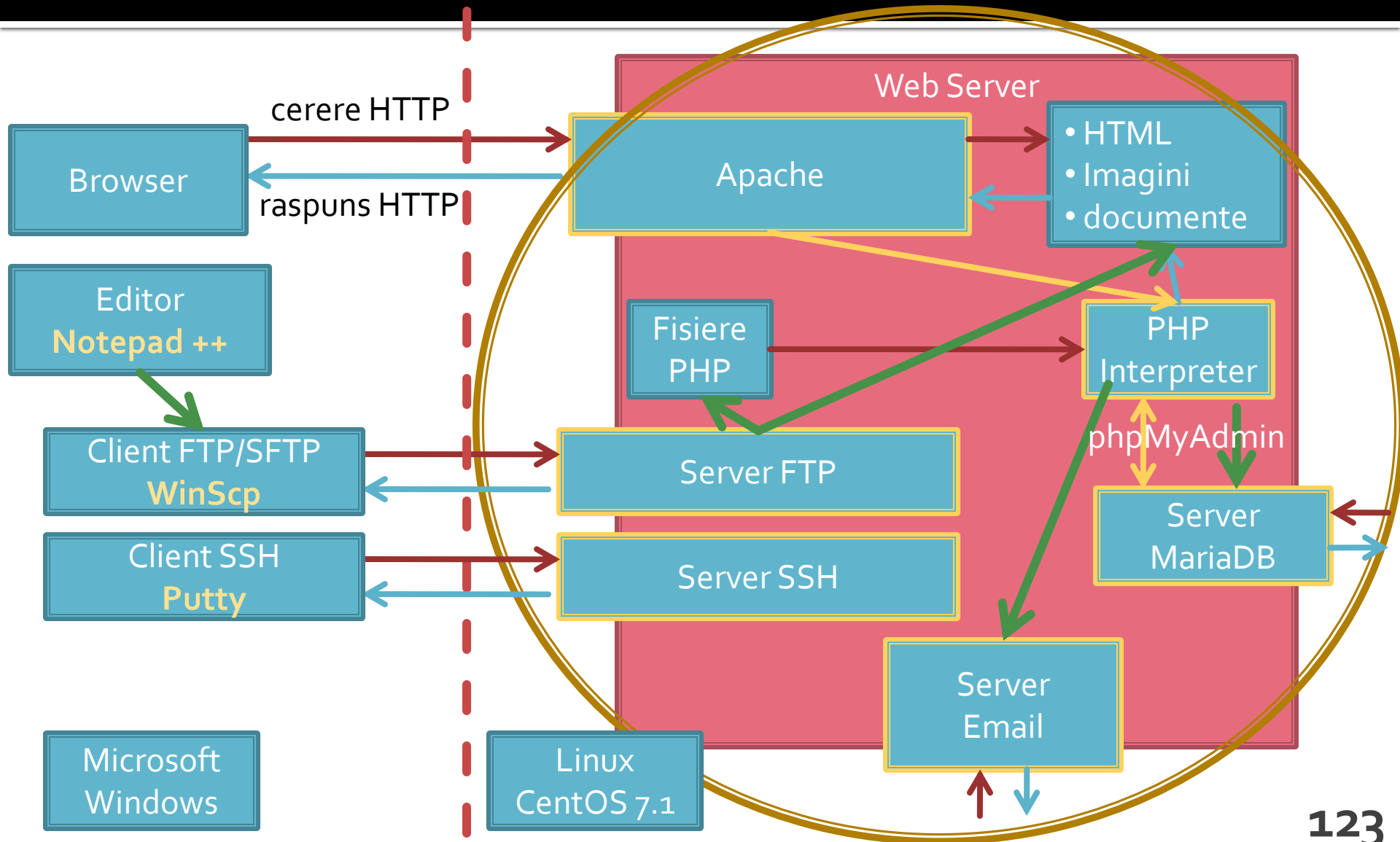
- 1. Se introduce in surse facilitatea template (S61)
- 2. Se modifica sursele pentru lucru cu matrici (S101)
- 3. Se modifica sursele pentru a citi datele de pe disc
  - **anterior** se creaza fisierul text sau:
  - **o singura data** se salveaza datele (S116)
- 4. Optional: Se creaza o pagina care sa permita modificarea fisierului
  - numai pret/descriere, fara adaugare/schimbare produse

# Laborator 4 – Matrice produse

- exemplu de grupare



# Utilizare LAMP



# Utilizare LAMP

**Magazin online Firma X SRL**

**Lista Produse**

Nr.	Produs	Pret
1	Carti	100
2	Caiete	50
3	Penare	150
4	Stilouri	125
5	Creioane	25

Comanda

**Magazin online Firma X SRL**

**Realizati comanda**

Nr.	Produs	Pret	Cantitate
1	Carti	100	<input type="text" value="1"/>
2	Caiete	50	<input type="text" value="2"/>
3	Penare	150	<input type="text" value="1"/>
4	Stilouri	125	<input type="text" value="0"/>
5	Creioane	25	<input type="text" value="0"/>

Trimite

**Magazin online Firma X SRL**

**Rezultate comanda**

Pret total (fara TVA): 350

Pret total (cu TVA): 416.5

Comanda receptionata la data: 17/03/2010 ora 08:24

<input name="x" ..

\$\_POST['x']

\$\_GET['x']

Web Server

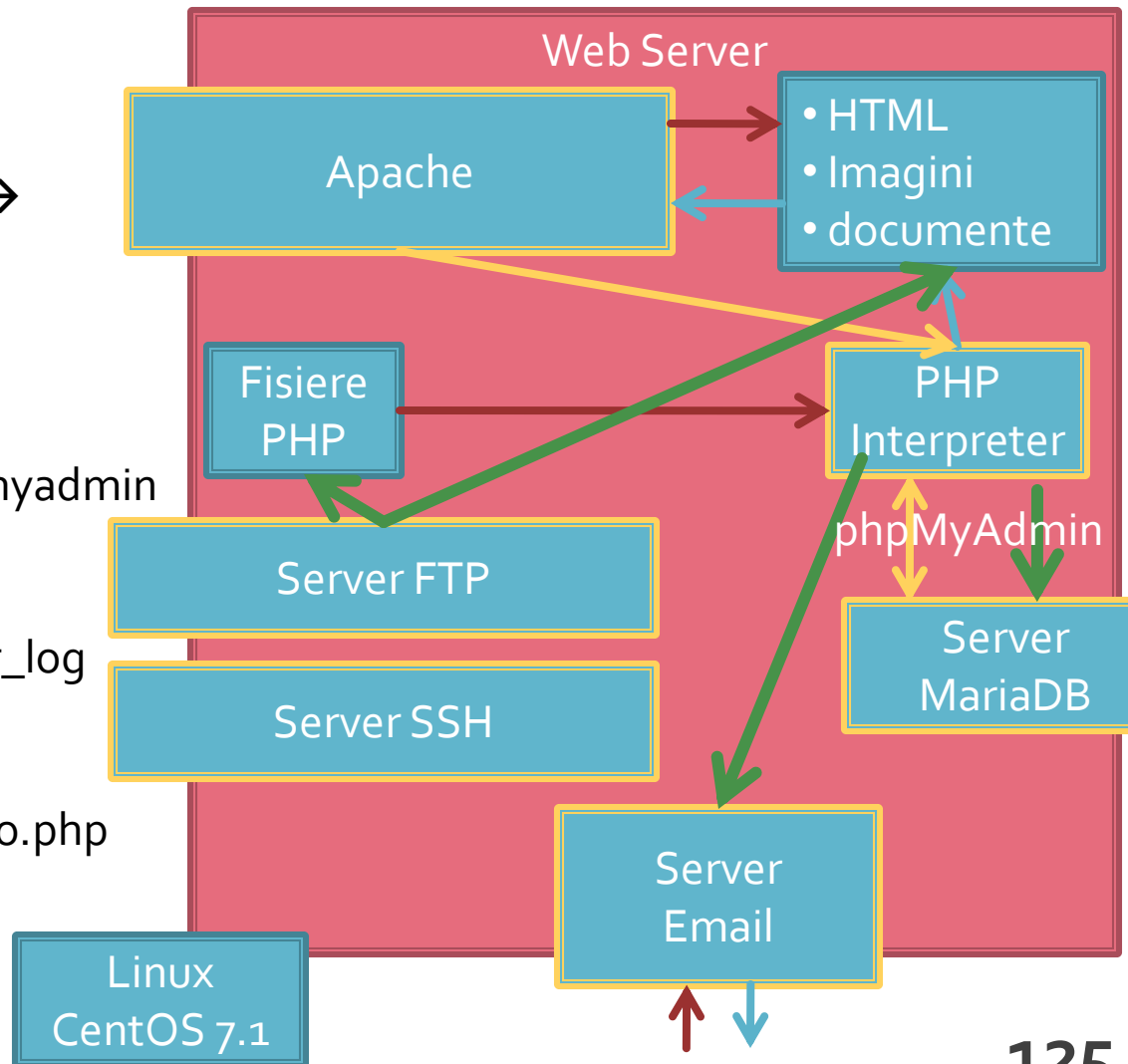
Apache

Server FTP

Server SSH

# Utilizare LAMP

1. login → root:masterrc
2. ifconfig → 192.168.30.5
3. putty.exe → 192.168.30.5 → SSH → root:masterrc (remote login)
4. [alte comenzi linux dorite]
5. FTP → Winscp → SFTP → student:masterrc@192.168.30.5
6. MySql → http://192.168.30.5/phpmyadmin → root:masterrc
7. Apache Error Log →
  - 7a. putty → nano /var/log/httpd/error\_log
  - 7b. http://192.168.30.5/logfile.php (nonstandard)
8. PHP info → http://192.168.30.5/info.php



# Client / Server

```
<input name="nume" ....>
```

```
echo $_POST['nume']; //ceva  
echo $_GET['nume']; //ceva  
echo $_REQUEST['nume']; //ceva
```

ceva

Trimite

get  
post

Interpretor PHP primeste  
\$\_POST  
\$\_GET  
\$\_REQUEST

# Depanare

```
echo "<pre>";  
print_r($_POST);  
echo "</pre>";
```

```
<p>temp <?php echo  
"a=";echo $a; ?> </p>
```

# Contact

- Laboratorul de microunde si optoelectronica
- <http://rf-opto.etti.tuiasi.ro>
- [rdamian@etti.tuiasi.ro](mailto:rdamian@etti.tuiasi.ro)