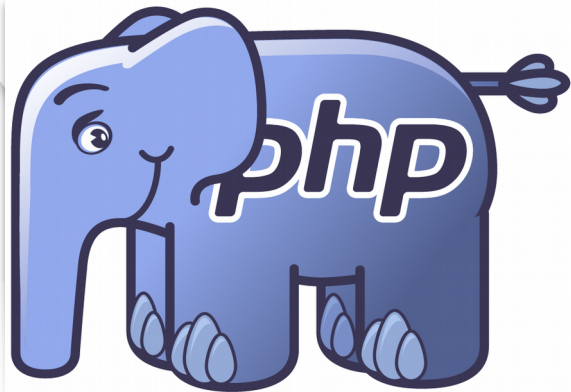




Univerza v Ljubljani  
Fakulteta za računalništvo  
in informatiko

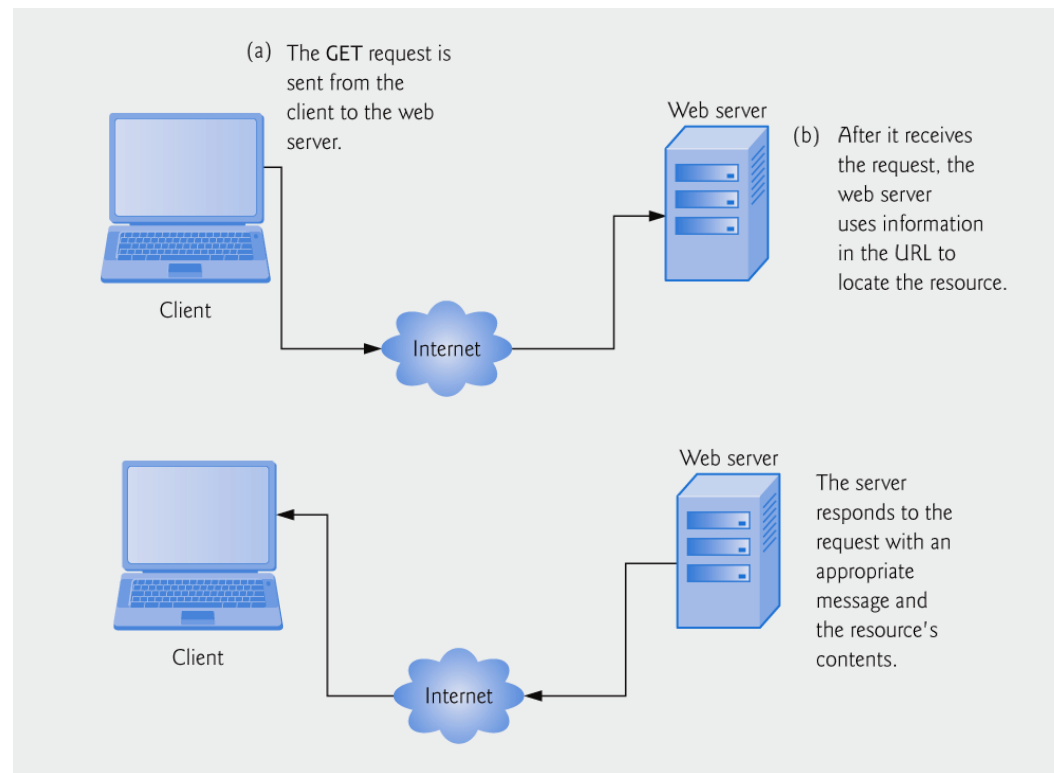


# PHP



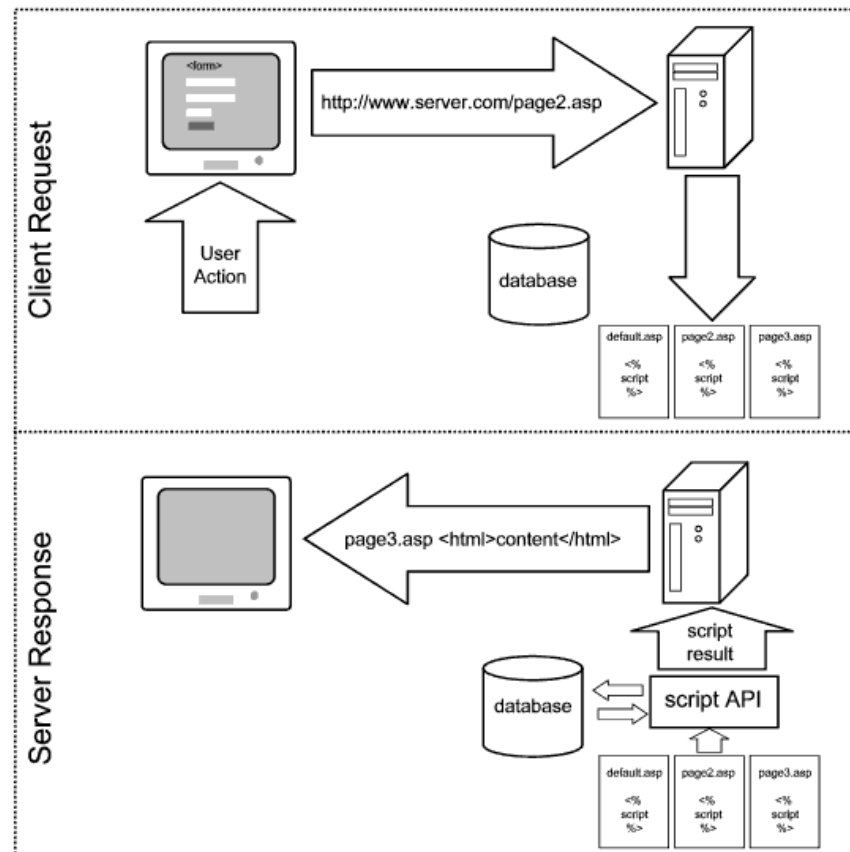
# Statična vsebina s strani strežnika

- ▶ odjemalec izvede zahtevek HTTP (metoda GET ali metoda POST)
- ▶ strežnik poišče zahtevano stran (resurs)
- ▶ strežnik odgovori s pošiljanjem zahtevane strani (resursa)
- ▶ odjemalec prejme zahtevano stran (resurs)
- ▶ odjemalec prikaže prejeto stran



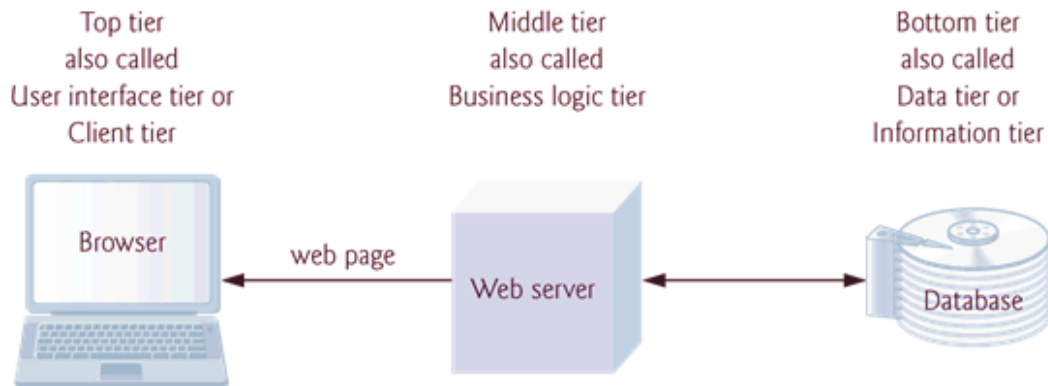
# Dinamična vsebina s strani strežnika

- ▶ izvedba zahtevka HTTP (metoda GET ali metoda POST)
  - ▶ strežnik dinamično generira stran
  - ▶ strežnik odgovori
- 
- ▶ odjemalec prejme unikatno stran:
    - odjemalec ne vidi strežnikove kode, ki je generiral stran
    - stran, ki jo prejme klient lahko vsebuje tudi kodo, ki se bo izvedla na njegovi strani (JavaScript)
  - ▶ odjemalec prikaže dobljeno stran



# Tronivojska aplikacija

- ▶ zaradi kompleksnosti dodan še en sloj
- ▶ spodnji: podatkovni sloj (podatki aplikacije)
- ▶ sredinski: poslovna logika (interakcija med odjemalcem in podatki; pravila za integriteto podatkov, pravice za dostop)
- ▶ zgornji: uporabniški vmesnik (prikaz podatkov, interakcija z uporabnikom)
- ▶ sloji so lahko porazdeljeni ali lokalni





# Spletni strežniki

- ▶ Microsoft IIS (Internet Information Services)
  - Microsoft WebMatrix - razvojno orodje za izdelavo aplikacij v jezikih PHP in ASP.NET
- ▶ Apache HTTP Server
- ▶ XAMPP: večplatformni paket (X), ki vsebuje Apache server (A), MySQL (M), modul PHP (P), modul Perl (P)
  - <http://www.apachefriends.org/en/xampp.html>





# PHP – nastavitev strežnika

- ▶ privzeto strežnik ne interpretira PHP kode, treba je namestiti pakete:
  - ubuntu/debian
    - php, libapache2-mod-php, php-mysql, php-pgsql, ...
  - windows
    - xamp
- ▶ ustrezne module je treba omogočiti
  - ubuntu/debian
    - `sudo a2enmod php7.0`
    - `sudo phpenmod pdo_mysql pdo_pgsql`
  - windows
    - odkomentirati vrstico kjer se naloži modul `php_module`



# Zemljevid predmeta

## Osnovni gradniki



vsebina



oblika



podatki

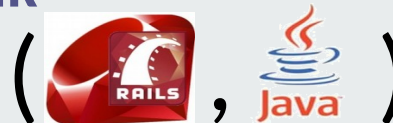
## Tehnologije na strani odjemalca in na strani strežnika



odjemalec



strežnik





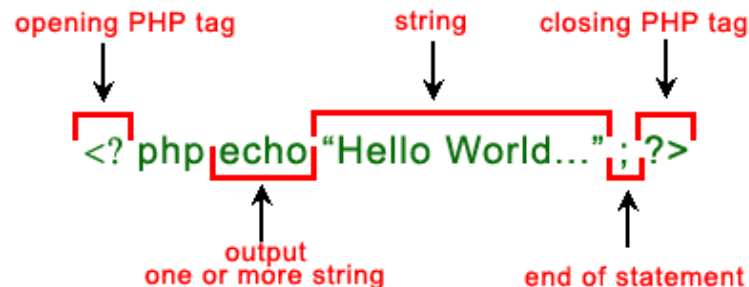
# PHP

- ▶ PHP => PHP: Hypertext Preprocessor (1994, Lerdorf)
- ▶ skriptni jezik na strani strežnika
- ▶ skriptna koda je interpretirana **preden** se pošlje odjemalcu
- ▶ namen:
  - procesiranje obrazcev
  - branje datotek
  - dostop do podatkovnih baz
- ▶ PHP je podoben JavaScriptu
  - dinamično tipiziran – tip spremenljivke se določi med izvajanjem
  - šibko tipiziran – implicitne konverzije spremenljivk
  - interpretiran (novejše izvedbe omogočajo prevajanje za kompleksne skripte)
  - spremenljivke so občutljive na velike/male črke, rezervirane besede pa ne
  - rezultat izvajanja skripte je HTML sintaksa



# PHP - osnovna sintaksa

- ▶ značka `<?php ... ?>` v dokumentu HTML
- ▶ vse spremenljivke se začnejo z znakom `$`
- ▶ imena spremenljivk se začnejo s podčrtajem ali črko
- ▶ imena spremenljivk lahko vsebujejo črke, številke in podčrta
- ▶ vrstice se zaključijo s podpičjem (`;`)
- ▶ za spremenljivke v dvojnih navednicah (npr. `"$ime"`) se izvede interpolacija – spremenljivko zamenja njena vrednost
- ▶ konkatencija nizov z operatorjem `.` (`"a"."b" -> "ab"`)
- ▶ komentarji: `// ...` in `# ...` (enovrstični), `/* ... */` (večvrstični)
- ▶ sestavljeni stavki `{ ... }`
- ▶ koda je lahko shranjena v več datotekah, `include(...)`
- ▶ končnica datoteke `.php`





# Primer

```
1 <?php print( '<?xml version = "1.0" encoding = "utf-8"?>' ) ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 23.1: first.php -->
6 <!-- Simple PHP program. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <?php
9     $name = "Harvey"; // declaration and initialization
10  ?><!-- end PHP script -->
11   <head>
12     <title>Using PHP document</title>
13   </head>
14   <body style = "font-size: 2em">
15     <p>
16       <strong>
17         <!-- print variable name's value -->
18         Welcome to PHP, <?php print( "$name" ); ?>!
19       </strong>
20     </p>
21   </body>
22 </html>
```

začetek skripte PHP

konec skripte PHP

deklaracija in  
inicializacija  
spremenljivke

interpolacija in izpis  
vrednosti  
spremenljivke v  
XHTML dokument

# Tipi spremenljivk

Tip	Opis
int, integer	cela števila
float, double, real	realna števila
string	besedilo v enojnih ( ' ') ali dvojnih ( " ") navednicah
bool, boolean	true ali false
array	skupina elementov
object	skupina med seboj povezanih lastnosti in metod
resource	zunanji vir – npr. podatek iz baze
NULL	brez vrednosti

- `unset(..)`, `IsSet(..)`, `is_tip(..): is_int(..)`, `is_double(..)`, ...
- `gettype(..)`: vrne tip, `settype(.., "tip")`: eksplicitna pretvorba tipa (v isto spremenljivko)
- `(tip) ...`(eksplicitna pretvorba v novo spremenljivko)

# Izpis

- ▶ **print** - za preprost neformatiran izpis
  - **print** (**niz**);
  - lahko se uporabi interpolacija spremenljivk
  - primer:

```
$day = "Tuesday";  
$high = 79;  
print ("The high on $day was $high\n");  
print "The high on $day was $high\n";
```
- ▶ **printf** - za formatiran izpis, sintaksa tega ukaza taka kot v C
  - **printf**(literal\_string, param1, param2, ...);
  - primer:

```
$day = "Tuesday";  
$high = 79;  
printf ("The high on %s was %d\n", $day, $high);
```

# Kontrola toka programa

- ▶ kontrolni ukazi imajo podoben obliko kot v C
  - operatorji: and, or, xor, &&, ||, !
  - relacijski operatorji - enako kot v JavaScriptu  
>, <, >=, <=, !=, ==: običajen pomen  
===: ista vrednost in tip, !==: ni ista vrednost oziroma tip
  - ```
if (pogoj1) {...}  
[elseif (pogoj2) {...}]  
[else {...}]
```
  - ```
switch ($vrednost){  
    case "...": ...; break;  
    default: ...  
}
```
  - ```
while (pogoj){  
    ...  
}
```
  - ```
do {  
    ...  
}while (pogoj)
```



# Kontrola toka programa

- `for (inicializacija; pogoj; sprememba vrednosti){  
 ...  
}`
- `foreach (array_expression as $value){  
 ...  
}`
- `foreach (array_expression as $key => $value) {  
 ...  
}`
- `break`, `continue` se lahko uporabita za prekinitev/preskok izvajanja `for`, `foreach`, `while` ali `do-while` konstruktov

# Polja (arrays)

- ▶ polja so kombinacija polj tipičnih jezikov in asociativnih polj
- ▶ vsak element polja sestavlja dva dela: ključ in vrednost
- ▶ če ima polje logično strukturo kot v drugih jezikih, so ključi nenegativna števila, v naraščajočem zaporedju
- ▶ če ima polje obliko asociativnih polj (hash), potem so ključi nizi, vrstni red pa določa funkcija za mapiranje (hash)
- ▶ izdelava polja

```
$list[0] = 17; //vrednost prvega elementa
```

```
$list[1] = "Prvi vnos"
```

```
$list[] = "Naslednji vnos" //izpuščen indeks naslovi naslednji element
```

```
$list = array()
```

```
$list = array(17, 24, 45, 91)
```

```
$list = array(1 => 17, 2 => 24, 3 => 45, 4 => 91)
```

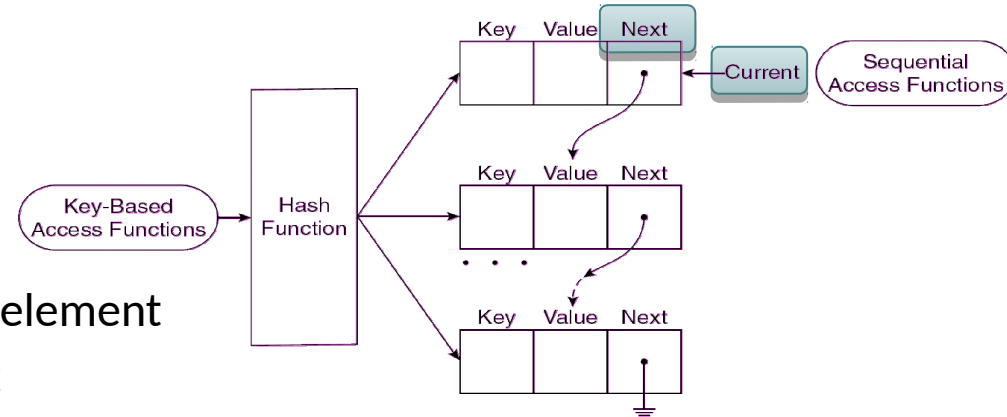
```
$list = array("Joe" => 42, "Mary" => 12, "Jack" => 33)
```

```
$list = array(5, 3 => 7, 5 => 10, "month" => "May");
```

# Iteriranje po elementih - funkcije

- ▶ funkcije:
  - `current()`: kaže trenutni element
  - `next()`: premakne kazalec na naslednji element
  - `prev()`: premakne kazalec na predhodni element
  - `reset()`: postavi kazalec na prvi element
  - `end()`: postavi kazalec na zadnji element
  - npr. `current($seznam)` //vrne trenutno vrednost
- ▶ `each` (vrne **ključ** in **vrednost**, prestavi kazalec naprej)

```
while ($tmp = each($colors)) {  
    print ("Ključ: $tmp['key'], vrednost: $tmp['value']");  
}
```







# Iteriranje po elementih

- ▶ **foreach (array\_name as scalar\_name) { ... }**

```
foreach ($colors as $color) {  
    print "Is $color your favorite color? <br />";  
}
```

- ▶ **foreach (array\_name as \$key => \$value) { ... }**

```
$ages = array("Bob" => 42, "Mary" => 43);  
foreach ($ages as $name => $age)  
    print("$name is $age years old <br />");
```

- ▶ **array\_push(...), array\_pop(...)**



# Sortiranje polj

- ▶ urejanje se izvede "in-place", popravi se originalno polje, ne dobimo popravljenih kopij → funkcije za urejanje ne vračajo rezultata
  - primer klica: `sort ($seznam)`
- ▶ tri+tri glavne funkcije:
  - `sort`: uredi po vrednosti, ključne nadomesti z 0, 1, 2, ..., nizi v abecednem redu na začetku, numerične vrednosti v numeričnem redu na koncu
  - `asort`: uredi po vrednosti tako kot `sort`, ključev ne spremeni
  - `ksort`: urejanje po ključih
  - `rsort`, `arsort`, `krsort`: urejanje vrednosti v obratnem vrstnem redu



# Regularni izrazi

- ▶ temeljijo na Perl (podobno kot v JavaScriptu)
  - navedemo v poševnicah, npr. `/student/`
  - posebni znaki: `.`, `*`, `+`, `?`, `{n}`, `[a-zA-Z]`, `^[a-zA-Z]`, `/^abc/`, `/abc$/`
- ▶ **`preg_match(regex, str [,array])`** *//iskanje niza*

Primer:

```
if (preg_match("/^PHP/", "PHP rocks!!!"))  
    print "String begins with PHP"
```

- ▶ **`preg_split(regex, niz)`** *//deljenje niza, podobno kot explode funkcija nad nizi*

Primer:

```
$fruits = "apple : orange : banana";  
$fruits = preg_split("/ : /", $fruits);  
// $fruits ima vrednost ["apple", "orange", "banana"]
```



# Funkcije

- ▶ splošna oblika funkcije v PHP:

```
function name ([formal_parameters]){  
    .....  
    [return val;]  
}
```

- ▶ definicija funkcije ni potrebna pred njeno uporabo
- ▶ število dejanskih parametrov je lahko različno od števila formalnih parametrov
- ▶ vidljivost spremenljivk v funkciji je lokalna, globalne spremenljivke moramo v funkciji deklarirati kot take:

```
global $aVariable;
```

- ▶ privzeto je klicanje po vrednosti
- ▶ klicanje po referenci:

```
function aFunction(&$name){...}
```

ali

```
afunction (&$name);
```



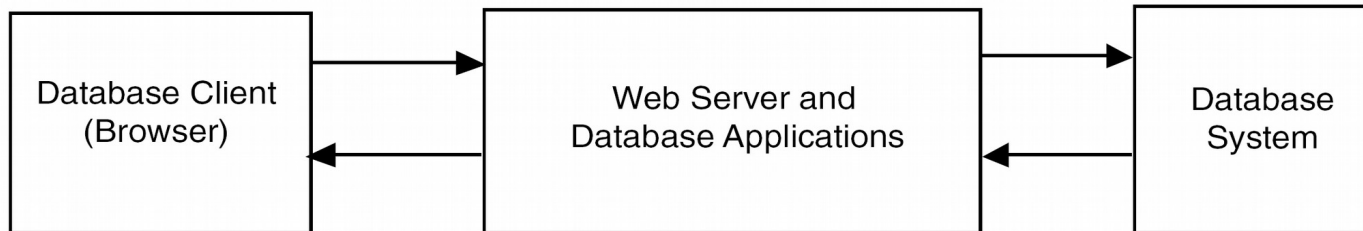
# Obrazci

- ▶ skripta podana z atributom `action` značke `form`
- ▶ atribut `method` obrazca določa način prenosa vrednosti (GET ali POST) strežniku
- ▶ PHP hrani vrednosti elementov obrazca v spremenljivkah `$_GET` in `$_POST` (vsebujeta ključe, ki ustrezajo imenom elementov obrazca in vrednost, ki so bile vnešene: npr. `$_POST["phone"]`)
- ▶ funkcija `extract($array)` priredi vrednosti v `$_GET` in `$_POST` lokalnim spremenljivkam (odsvetovani pristop, prepisemo lahko obstoječe spremenljivke!)
- ▶ druge globalne spremenljivke (superglobal arrays):

Ime spremenljivke	Opis
<code>\$_SERVER</code>	Podatki o strežniku
<code>\$_ENV</code>	Podatki o okolju uporabnika
<code>\$_GET</code>	<b>Podatki, poslani strežniku v zahtevku GET</b>
<code>\$_POST</code>	<b>Podatki, poslani strežniku v zahtevku POST</b>
<code>\$_COOKIE</code>	<b>Podatki v piškotku na uporabnikovem računalniku</b>
<code>\$_REQUEST</code>	<b>Podatki po pošiljanju obrazca HTML</b>
<code>\$GLOBALS</code>	Polje vseh globalnih spremenljivk

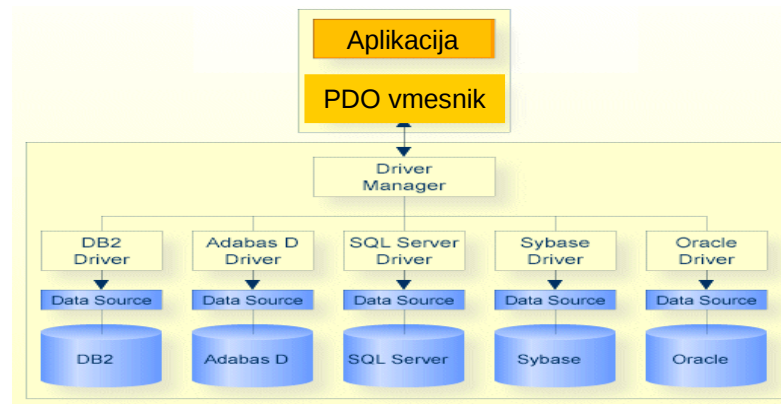
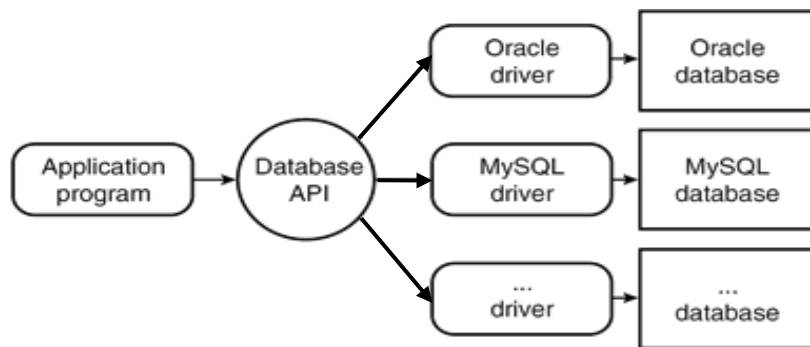
# Arhitektura za dostop do baze

- ▶ odjemalec-strežnik
- ▶ tronivojska:
  - odjemalec (brskalnik)
  - spletni strežnik z dostopom do podatkovne baze
  - podatkovna baza



# Dostop do baze iz PHP

- ▶ dve možnosti:
  - **v 1 koraku:** povezovanje direktno z bazo preko namenskega gonilnika za uporabljeno bazo (API)
  - **v 2 korakih:** z uporabo PHP DatabaseObject (PDO), ki je vmesnik za povezovanje z različnimi bazami. Za posamezno bazo je potreben gonilnik PDO .
- ▶ mogoča je uporaba SQL poizvedb ali "Object-Relational Mappers" (ORM)



# Namestitev MySQL in kreiranje uporabnika

- ▶ `$ sudo apt install mysql-server`
- ▶ `$ sudo /etc/init.d/mysql stop`
- ▶ `$ sudo mkdir /var/run/mysqld`
- ▶ `$ sudo chown mysql /var/run/mysqld/`
- ▶ `$ sudo mysqld_safe --skip-grant-tables &`
- ▶ `mysql> update user set authentication_string=PASSWORD('mysql') where user='root';`
- ▶ `mysql> flush privileges;`
- ▶ `$ sudo /etc/init.d/mysql start`
- ▶ `$ mysql -h 127.0.0.1 -u root -p`
- ▶ `mysql> GRANT ALL PRIVILEGES ON *.* TO 'aless'@'localhost' IDENTIFIED BY 'password';`
- ▶ `mysql> mysql -u aless -h 127.0.0.1 -p`
- ▶ `mysql> CREATE DATABASE tehnologije;`
- ▶ `mysql> USE tehnologije;`
- ▶ `mysql> CREATE TABLE sample (id smallint not null, name varchar(255));`
- ▶ `mysql> INSERT INTO sample (id, name) VALUES (1, 'Danko');`
- ▶ `mysql> SELECT * FROM sample;`



# Povezava s podatkovno bazo

```
$db = mysqli_connect(host, user, pass) # povezava do baze  
mysqli_select_db("cars", $db);      # izbira sheme  
$query = "SELECT * from States";    # generiranje stavka SQL  
$result = mysqli_query($query, $db); # izvedba poizvedbe → bool, rezultat  
if ($result) {mysqli_fetch_row($result);} # »branje« rezultata  
mysqli_close($db)      # zapiranje povezave
```

varianete funkcije:

- `mysql(i)_connect` # `i` – improved, novejša različica (za MySQL 4.1.3 in višje)  
`p` – persistent

pomožne funkcije:

- `mysqli_error` # sporočilo baze o napaki
- `mysqli_fetch_row` # vrne naslednjo vrstico rezultata



# Delo s podatkovno bazo: MySQLi

```
<!DOCUMENT html>
<html>
<head>
<title>MySQLi</title>
<meta charset="utf-8">
</head>
<body>
.....
</body>
</html>
```

```
<?php
    $file=fopen("connectData.txt","rt");
    $data=fread($file, filesize("connectData.txt"));
    $connD=explode(" ", trim($data));
    $conn=new mysqli($connD[0], $connD[1], $connD[2],$connD[3]);

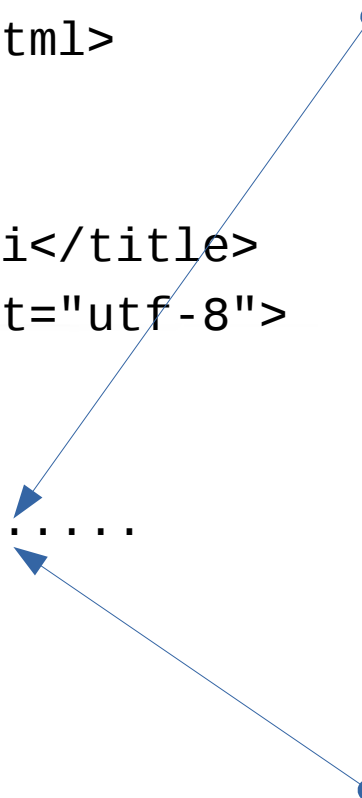
    if ($conn->connect_error){
        print("<h1>povezava NI uspela</h1>"); }
    else {print("<h1>povezava JE uspela</h1>");}
    $sql = "SELECT id,name FROM sample";
    $result = $conn->query($sql);
    if ($result->num_rows > 0){
        while ($row = $result->fetch_assoc()){
            print("<h2> id: ".$row["id"]." Name: ".$row["name"]);
        }
    }
    $conn->close();
?>
```

# Delo s podatkovno bazo: pgsql

```
<!DOCUMENT html>
<html>
<head>
<title>MySQLi</title>
<meta charset="utf-8">
</head>
<body>
. . . . .
</body>
</html>
```

```
?php
$file=fopen("connectData.txt","rt");
$data=fread($file, filesize("connectData.txt"));
$connD=explode(" ", trim($data));
$cS="host=$connD[0] dbname=$connD[3] user=$connD[1] password=$connD[2]";
$conn= pg_connect($cS);

if (!$conn){
    print("<h1>povezava NI uspela</h1>");
}
else {
    print("<h1>povezava JE uspela</h1>");
}
$sql = "SELECT id,name FROM sample";
$result = pg_query($conn, $sql);
if ($result != false){
    while ($row = pg_fetch_assoc($result)){
        print("<h2> id:".$row["id"]." Name: ".$row["name"]);
    }
}
pg_close($conn);
?>
```



# Delo s podatkovno bazo: PDO

```
<!DOCUMENT html>
<html>
<head>
<title>MySQLi</title>
<meta charset="utf-8">
  <style>
    table, th, td
    {border: 1px solid blue;
     font-size: 1.5em;}
  </style>
</head>
<body>
  .....
</body>
</html>
```

```
<?php
$file=fopen("connectData.txt","rt");
$data=fread($file, filesize("connectData.txt"));
$connD=explode(" ", trim($data));
try{
    $conn=new PDO("mysql:host=$connD[0];dbname=$connD[3]", $connD[1], $connD[2]); ALI
    $conn=new PDO("pgsql:host=$connD[0];dbname=$connD[3]", $connD[1], $connD[2]);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    print("<h1>povezava JE uspela</h1>");
    $stmt = $conn->prepare("SELECT primer_id, ime, priimek FROM primercek");
    $stmt->execute();
    $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
    print("<table><thead><tr><th>ID<th>Ime<th>Priimek</thead>");
    print("<tbody>");
    foreach ($result as $row){
        print("<tr><td>".$row["primer_id"]."<td>".$row["ime"]."<td>".$row["priimek"]);
    }
    print("</tbody>");
}
catch(PDOException $e) {
    print("<h1>povezava NI uspela</h1>");
}
$conn = null;
?>
```



# Delo z ORM - Doctrine

- ▶ namestite ustrezne knjižnice
  - v ubuntu: doctrine-orm
  - `sudo apt install php-doctrine-orm composer`
- ▶ za uporabo je potrebno ustvariti projekt

- kreiraj direktorij za svoj projekt
- v tem direktoriju definiraj `composer.json`
- izvedi `composer install`
- naredi direktorije

- `src`,
- `config`,
- `config/yaml`,
- `config/xml`

```
{  
  "require": {  
    "doctrine/orm": "2.*",  
    "symfony/yaml": "2.*"  
  },  
  "autoload": {"psr-0": {"": "src/"}}  
}
```



# Delo z ORM - Doctrine

► ...

- naredi datoteko bootstrap.php

```
<?php
use Doctrine\ORM\Tools\Setup
use Doctrine\ORM\EntityManager
require_once "vendor/autoload.php"
$isDevMode=true
$config=Setup::createAnnotationMetadataConfiguration(array(__DIR__."src",
                                                         $isDevMode, null, null, false);

$conn = array('driver' => 'pdo_mysql', ... );
$entityManager = EntityManager::create($conn, $config);
?>
```

- naredi datoteko cli-config.php

```
<?php
require_once "bootstrap.php"
return
    Doctrine\ORM\Tools\Console\ConsoleRunner::createHelperSet($entityManager);
?>
```



# Delo z ORM - Doctrine

- ▶ ...
  - ustvari shemo podatkovne baze:
    - `vendor/bin/doctrine orm:schema-tool:create`
    - (med popravljanjem baze je potrebno popraviti tudi sheme)  
`vendor/bin/doctrine`  
`orm:schema-tool:update --force --dump-sql`
  - v začetnem koraku se ne zgodi nič
  - potrebno je še ustvariti razrede za entitete

# Delo z ORM - Doctrine

► ...

- razred Primercek:
- potrebno je dodati anotacije
- omogočajo pravilno preslikavo med podatki v bazi in lastnostmi objektov v PHP
- ob spremembi razreda je potrebno pognati posodobitev baze

```
<?php
use Doctrine\ORM\Mapping as ORM;
/**
 * @ORM\Entity @ORM\Table(name="primercek")
 */
class Primercek{
    /**
     * @var int
     */
    /** @ORM\Id @ORM\Column(type="integer") @ORM\GeneratedValue */
    protected $primer_id;
    /**
     * @var string
     */
    /** @ORM\Column(type="string") */
    protected $ime;
    /**
     * @var $priimek
     */
    /** @ORM\Column(type="string") */
    protected $priimek;
    public function getId() {return $this->primer_id;}
    public function getIme() {return $this->ime;}
    public function getPriimek() {return $this->priimek;}
    public function setTime($ime) {$this->ime = $ime;}
    public function setPriimek($priimek)
        {$this->priimek = $priimek;}
}
?>
```



# Delo z ORM - Doctrine

- ▶ ...
  - delo s podatki v bazi se lahko izvaja nad objekti v PHP.
  - kreiranje novega PHP objekta omogoča enostavno dodajanje nove vrstice v bazo
  - preko EntityManager objekta lahko beremo posamezno vrstico iz baze

```
<?php
require_once "bootstrap.php";
if (sizeof($argv)<3){
    echo "Premalo parametrov, koncujem\n";
    exit (1);
}
$prm = new Primercek();
$prm->setIme($argv[1]);
$prm->setPriimek($argv[2]);
$entityManager->persist($prm);
$entityManager->flush();
echo "Oseba: ". $prm->getId(). "\n";
?>
```

```
<?php
require_once "bootstrap.php";
$id = $argv[1];
$person = $entityManager->find('Primercek', $id);
if ($person == null){exit(1);}
echo "Oseba: " . $person->GetIme() . " " . $person->getPriimek()."\n";
?>
```

# Delo z ORM - Doctrine

- ▶ ...
  - možno je prebrati tudi celotno tabelo v bazi
  - popravljane vrstice v bazi se lahko izvede kot popravljane objekta
  - potrebno je eksplicitno zahtevati pisanje podatkov v bazo

```
<?php
require_once "bootstrap.php";
$perRep = $entityManager->getRepository('Primercek');
$persons = $perRep->findAll();
foreach ($persons as $p){
    echo sprintf("%d %s %s\n", $p->getId(),
                    $p->getIme(), $p->getPriimek());
}
?>
```

```
<?php
require_once "bootstrap.php";
$person=$entityManager->find('Primercek', $argv[1]);
if ($person == null){ exit(1); }
$person->setIme($argv[2]);
$entityManager->flush();
?>
```

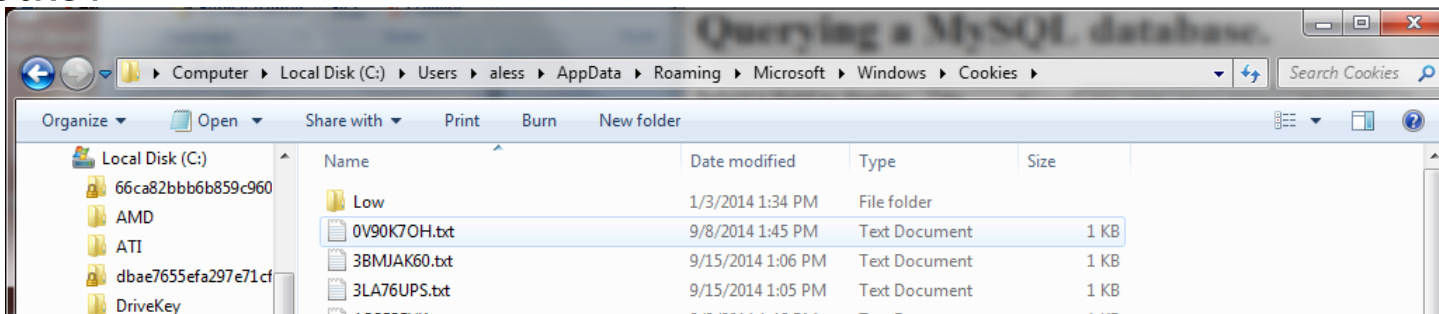


# Hranjenje informacije o uporabnikih

- ▶ spletna stran pozna istega uporabnika na različnih računalnikih
  - uporabnik se pač prijavi na spletno stran, ni panike
- ▶ spletna stran pozna uporabnika
  - ko pridem na spletno stran dobim personalizirano sporočilo brez predhodne prijave
- ▶ spletna stran pozna uporabnikove navade
  - ko odprem iskalnik Google, mi med možnostmi iskanja ponudi iskanja, ki sem jih v preteklosti že izvedel, brez prijave
  - ko pridem v spletno trgovino mi med priporočene izdelke uvrsti izdelke, ki sem jih kupil ali pa sem si jih ogledoval, brez prijave
- ▶ kako je to mogoče?

# Piškotki (cookies)

- ▶ protokol HTTP ne hrani stanja, kar bi bilo koristno pri določenih aplikacijah (oglaševanje, nakupovanje, personalizacija)
- ▶ piškotki hranijo pare ime/vrednost, ki se prenesejo v **glavi HTTP**. Zato je v splošnem potrebno piškotke izdelati pred kakršnim koli HTTP izpisom!
  - (izjema: nastavitev `output_buffering` v `php.ini` oziroma `ob_start()` v skripti)
- ▶ strežnik lahko dostopa samo do podatkov, ki jih je sam shranil (varnost!)
- ▶ piškotki lahko posegajo v zasebnost uporabnika, zato lahko uporabnik zavrne uporabo piškotkov



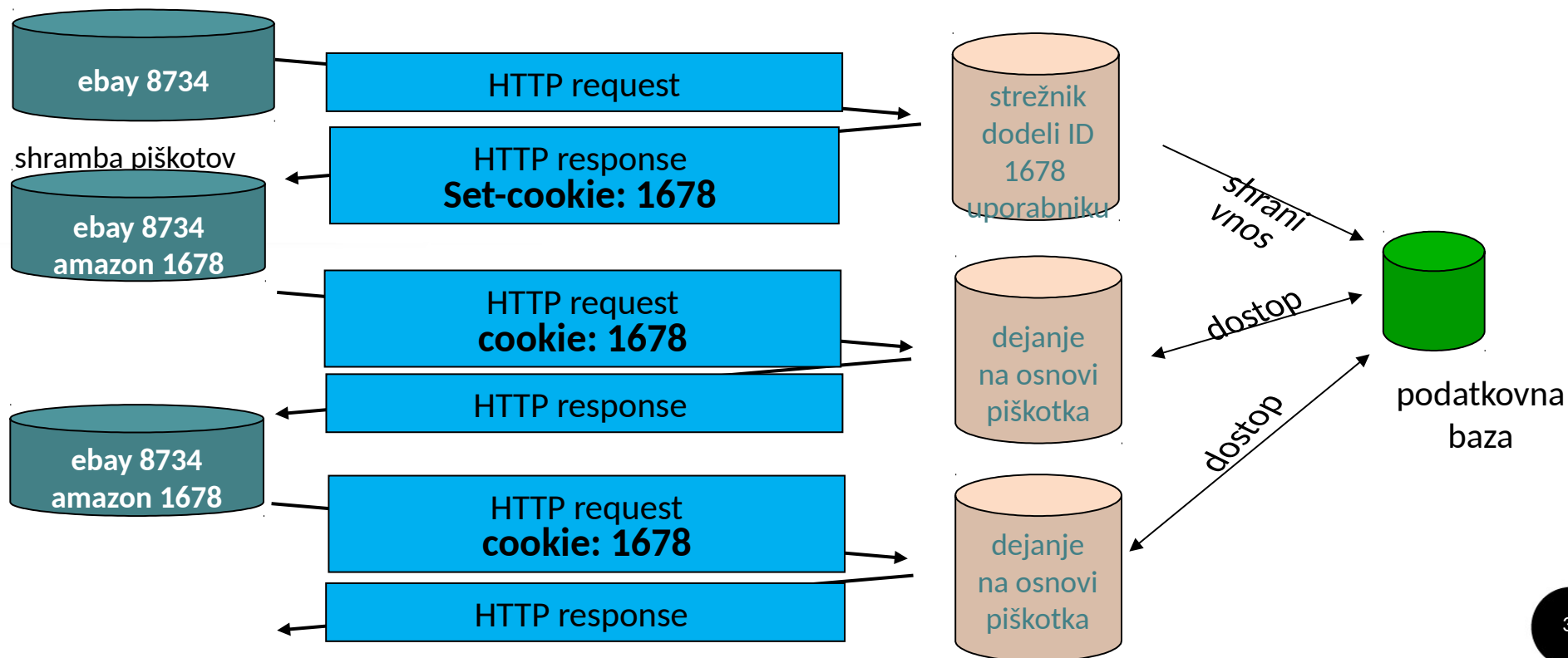
# Piškotki (cookies)

- ▶ lahko se shrani informacija o uporabniku
- ▶ majhna količina podatkov, tekstovni niz
- ▶ hrani jo brskalnik na strani odjemalca
- ▶ podatke lahko bere samo strežnik, ki je piškotek tudi ustvaril
- ▶ potrebno je obveščanje uporabnika o piškotkih
- ▶ možnost zavrnitve piškotkov, ki niso nujni za funkcionalnosti spletne strani
- ▶ uporabljajo se za:
  - avtentikacijo
  - sledenje uporabnika
  - urejanje nakupovalnega vozička
  - ...





# Uporaba piškotov





# Piškotki (cookies)

## ► sintaksa:

`setcookie(name, value, expire, path, domain, secure, httponly)`

### – obvezno

- name → ime piškota
- value → vsebina do 4096 znakov

### – opcijsko

- expire → čas do poteka veljavnosti
- path → za katero poddrevo v domeni je piškot dostopen
- domain → v kateri domeni je piškot
- secure → ali je piškot dostopen samo preko https povezave
- httponly → dostopnost z JavaScript (true → ni dostopen z JS)

# Piškotki (cookies)

- ▶ primer piškotka, ki se izbriše po enem dnevu:  
`setcookie("name", "Danko", time() + 24*60*60);`
- ▶ po preteku časovne veljavnosti se piškoti zbrišejo
  - če veljavnost ni podana, je veljavnost do konca seje
- ▶ piškote beremo v PHP iz polja `$_COOKIE` ali `$_REQUEST`  
`if (isset($_COOKIE[name])) $uname=$_COOKIE[name];`
- ▶ piškote brišemo z enakimi parametri, kot so bili ustvarjeni, datum kreiranja se postavi v preteklost  
`setcookie("name", "Danko", time() - 24*60*60);`
- ▶ imena piškotov morajo biti polje imen, ki je na voljo v skripti PHP



# Piškotki (cookies)

- ▶ primer piškotka, ki se izbriše po enem dnevu:  
`setcookie("name", "Danko", time() + 24*60*60);`
- ▶ po preteku časovne veljavnosti se piškoti zbrišejo
  - če veljavnost ni podana, je veljavnost do konca seje
- ▶ piškote beremo v PHP iz polja `$_COOKIE` ali `$_REQUEST`  
`if (isset($_COOKIE[name])) $uname=$_COOKIE[name];`
- ▶ piškote brišemo z enakimi parametri, kot so bili ustvarjeni, datum kreiranja se postavi v preteklost  
`setcookie("name", "Danko", time() - 24*60*60);`
- ▶ imena piškotov morajo biti polje imen, ki je na voljo v skripti PHP

# Piškotki (cookies)

```
<?php
    $val="Jaz sem piskotek";
    setcookie("testPiskotek", $val);
?>
<!doctype html>
<html>
    <head>
        <title>Piskotki</title>
    </head>
    <body>
        <?php
            echo "Nastavil sem piskotek";
            if (isset($_COOKIE["testPiskotek"])) echo "<h2>Piskotek setiran</h2>";
        ?>
    </body>
</html>
```

- ▶ ob prvem dostopu se piškotek nastavi, izpiše se, da je nastavljen
- ▶ ob ponovnem nalaganju pa se obstoječ piškotek prebere in se izpiše njegova vrednost

# Seje (sessions)

- ▶ seje so druga možnost za vodenje uporabnikove identitete
- ▶ imajo več funkcij
  - določitev enoličnega identifikatorja seje
  - uporabljajo se za pridobivanje informacij povezanih s konkretno sejo
- ▶ spremenljivke povezane s sejo, dostopne preko spremenljivke `$_SESSION`
- ▶ stanje seje se shrani v datoteko ali podatkovno bazo s klicem funkcije `session_state_save_handler()`
- ▶ seja ostane veljavna, dokler je brskalnik aktiven (se ne zapre) oziroma dokler se seja programsko ne zaključi



# Seje (sessions)

- ▶ delo s seji
  - začetek seje → `session_start()`  
oziroma nadaljevanje s prejšnjo sejo glede na identifikator, ki se ga pridobi preko zahtevka ali piškotka
  - dostop do identifikatorja seje → `session_id()`
  - končanje seje → `session_destroy()`
  - spremenljivke se shranjujejo v `$_SESSION`  
`$_SESSION['varName']=varValue;`



`$_SESSION`

# Seje (sessions) - kreiranje

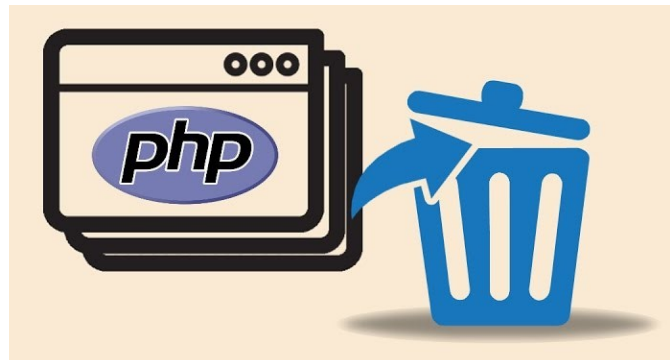
```
<!DOCTYPE html>
<html>
  <head><meta charset="UTF-8"></head>
  <body>
    <?php
      session_start();
      if (!isset($_SESSION['accCount'])) { $_SESSION['accCount']=1; }
      else{ $_SESSION['accCount']++; }
    ?>

    <?php
      echo "<h2>ID seje je " . session_id() . "</h2>";
      echo "<h3>Do strani si dostopal " . $_SESSION['accCount'] . "</h3>";
    ?>
  </body>
</html>
```



# Seje (sessions) - uničenje

```
<!DOCTYPE html>
<html>
  <head><meta charset="UTF-8"></head>
  <body>
    <?php
      session_start();
      if (!isset($_SESSION['accCount'])) { $_SESSION['accCount']=1; }
      else{ $_SESSION['accCount']++; }
    ?>
    <?php
      echo "<h2>ID seje je " . session_id() . "</h2>";
      echo "<h3>Do strani si dostopal " . $_SESSION['accCount'] . "</h3>";
      if ($_SESSION['accCount']>=5) {
        echo "Koncujem sejo";
        session_destroy();}
    ?>
  </body>
</html>
```



# Seje (sessions) – za spletne trgovine

- ▶ z uporabo seje je enostavno implementirati nakupovalno košaro
- ▶ vsakič, ko izberem produkt in ga dodam v košarico, se doda spremenljivka z ustrezno vrednostjo v sejo → seja predstavlja nakupovalno košarico
- ▶ produkte pridobimo tako, da preberemo vse spremenljivke iz seje in si zapomnimo vrednosti
- ▶ nakupovalne košarice so ponavadi implementirane z uporabo sej

