



Principi softverskog inženjerstva

Vežbe - VI - VIII nedelja PHP - osnovni kurs

Dražen Drašković, asistent
Elektrotehnički fakultet
Univerziteta u Beogradu

Aplikacija



- Aplikacija je bilo koji program koji je napravljen da bi bio pojednostavljen ili obavljen neki zadatak.
- Klasična aplikacija je instalirana na klijentskoj mašini.
- Klasična aplikacija se pokreće izvršavanjem određenog izvršnog fajla (exe).

Veb aplikacija



- Veb aplikacija je aplikacija koja se nalazi na veb-u i kojoj se pristupa putem internet-a.
- Veb aplikacija je instalirana na nekom udaljenom serveru.
- Veb aplikacija se pokreće zahtevom odgovarajućeg URL-a u veb pregledaču, koji predstavlja interfejs veb aplikacija.

Aplikacije i baze podataka



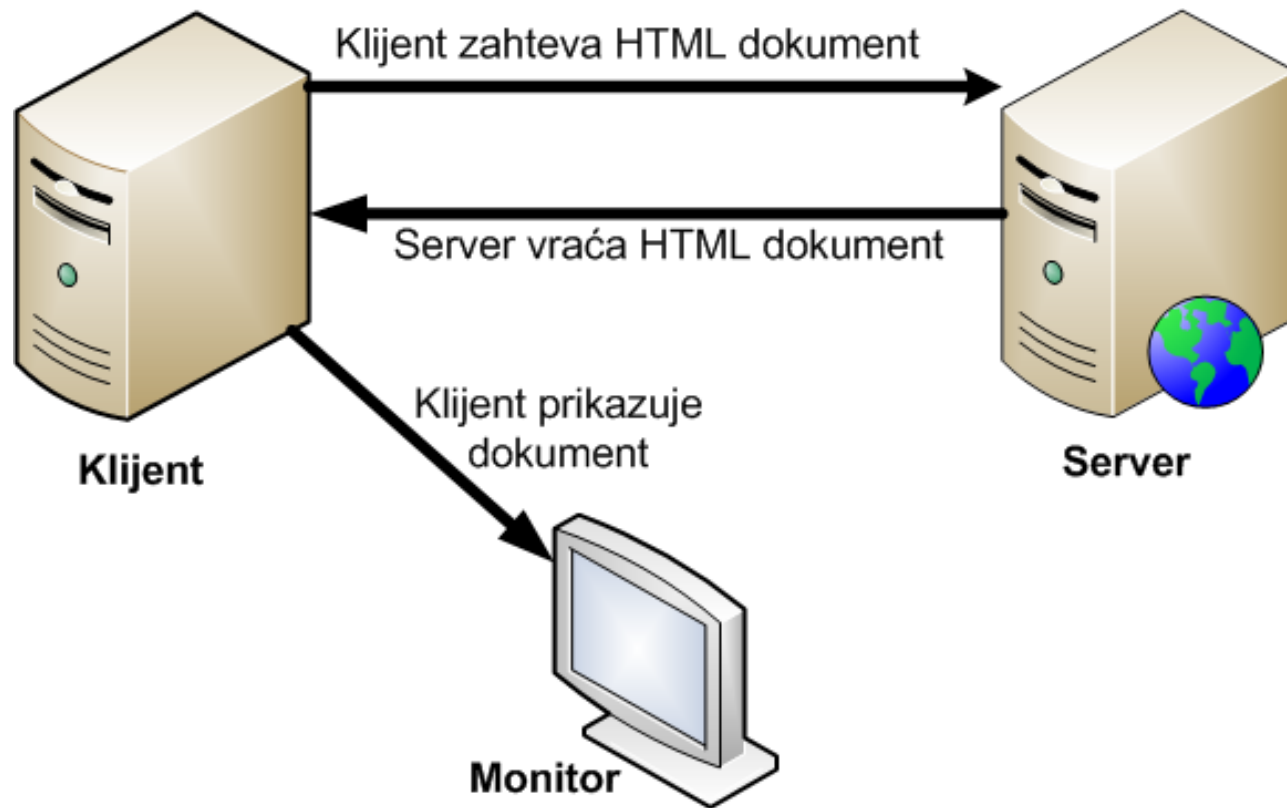
- Baza podataka se koriste za skladištenje podataka, ako klasična ili veb aplikacija rade sa većom količinom podataka.
- Podaci u bazi podataka su potpuno nezavisni od aplikacije.
- Posebnim naredbama aplikacija čita podatke iz baze podataka i ažurira podatke u bazi podataka (dodaje, briše, menja).

Statički veb sajtovi



- Statički sajtovi se ne menjaju sve dok sam autor nešto ne promeni
- Omogućavaju slanje informacija ka korisnicima
- Korisnici nemaju mogućnost interakcije i ne mogu neki zadatak da izvršavaju na programabilan način
- Statičke veb sajtove čine obične statičke HTML stranice (skup HTML fajlova)

Zahtev za HTML dokumentom

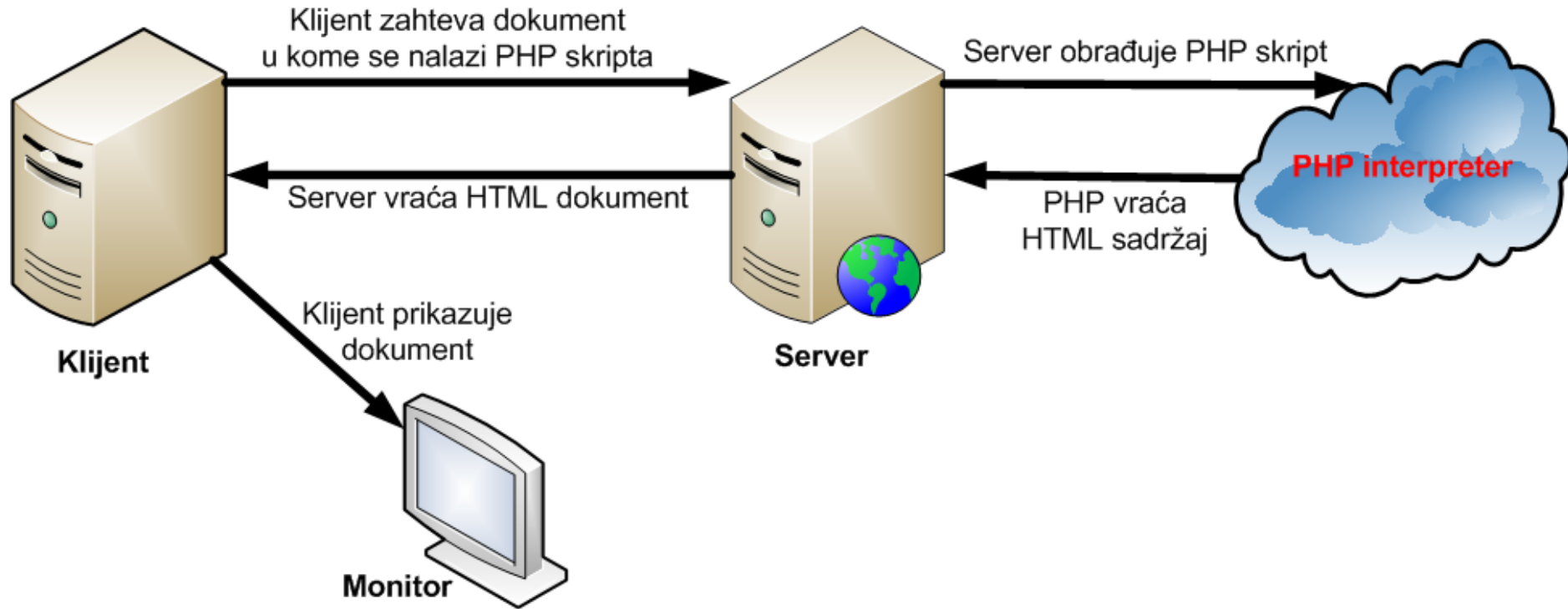


Dinamički veb sajtovi



- Dinamički veb sajtovi omogućavaju interakciju sa korisnikom.
- Dinamičke veb sajtove čine dinamičke stranice (skup php, asp, jsp... fajlova) koje takođe koriste HTML za komunikaciju sa klijentom.

Zahtev za PHP skriptom

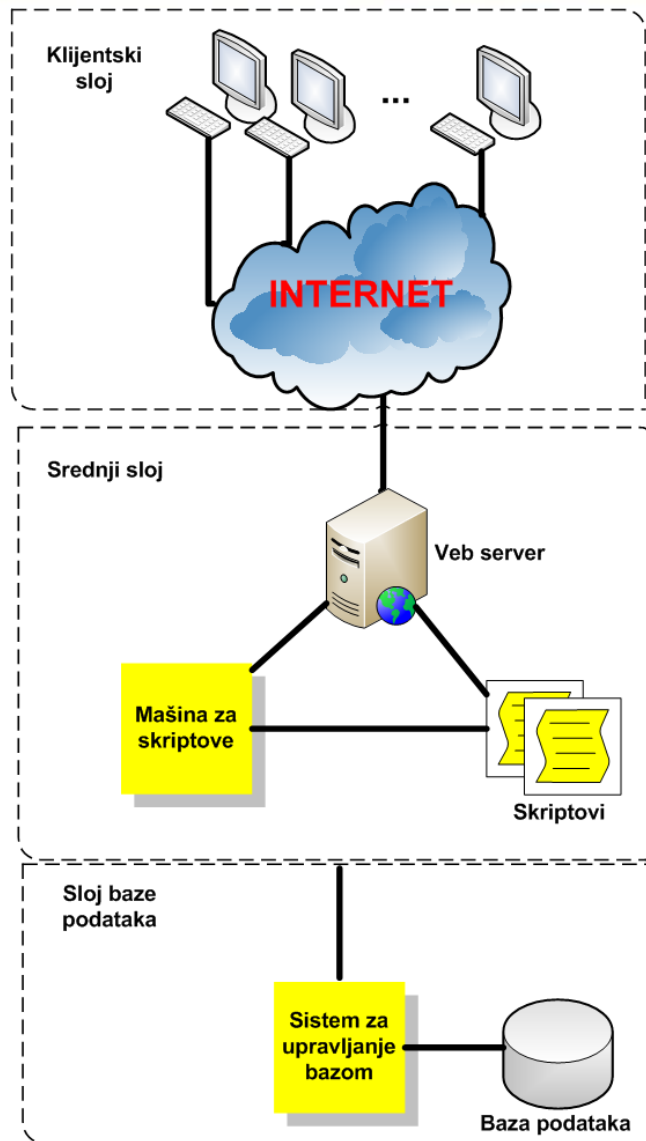


Troslojna arhitektura



- Većina veb aplikacija koje rade sa bazama podataka poseduje tzv. troslojnu arhitekturu koju čine sledeći slojevi:
 - klijentski sloj
 - aplikacioni sloj
 - sloj baze podataka

Arhitektura troslojne veb aplikacije



Kako su povezani slojevi?



- Sam veb obezbeđuje mrežu i protokole koji povezuju klijentski sloj sa srednjim slojem, a to je dominantno HTTP (HyperText Transfer Protocol).
- Komunikacija između srednjeg sloja i sloja baze podataka se obavlja pomoću protokola zavisnog od tipa programskog jezika i tipa baze podataka.
- Komunikacija preko HTTP-a dominira mrežnim saobraćajem na internetu (oko 80%).
- Za izradu veb aplikacija sa bazama podataka nije potrebno detaljno poznavati HTTP, ali je vrlo važno znati kakve probleme može da izazove HTTP u veb aplikacijama koje rade sa bazama podataka.

Osnove HTTP-a



- HTTP je standard koji omogućava prosleđivanje i deljenje dokumenata na veb-u.
- HTTP je jednostavan:
klijent (koji je u većini slučajeva veb pregledač) šalje HTTP serveru zahtev za određenim resursom, a server mu šalje svoj odgovor, u kome je sadržan i sam resurs - najčešće HTML dokument.

Čuvanje stanja



- Klasične aplikacije koje rade sa bazama podataka čuvaju stanje (korisnici se prvo prijavljuju, zatim izvršavaju pojedine transakcije, i na kraju kad obave posao odjavljuju se).
- HTTP ne čuva stanje tj. svaka interakcija između veb pregledača i veb servera je nezavisna od bilo koje druge interakcije.
- Dakle treba na neki način obezbediti čuvanje tekućeg stanja. Za čuvanje tekućeg stanja PHP obezbeđuje kolačiće i sesije.

Klijentski sloj



- Klijentski sloj u modelu troslojne arhitekture je veb pregledač (engl. *web browser*).
- Veb pregledač šalje HTTP zahteve za resursima, obrađuje HTTP odgovore i prikazuje HTML resurse.
- Prednosti upotrebe veb pregledača kao tankog klijenta:
 - jednostavan razvoj
 - nezavisnost od platforme na klijentskoj strani
- Najpopularniji veb pregledači:
Internet Explorer, Mozilla Firefox,
Google Chrome, Opera,...

Veb pregledači (čitači)



- Zajedničke odlike pregledača (čitača) veb-a:
 - Svi veb pregledači su HTTP klijenti koji šalju zahteve i prikazuju odgovore dobijene od veb servera (obično u nekom grafičkom okruženju)
 - Svi čitači tumače sadržaj HTML stranica
 - Neki čitači prikazuju slike, reprodukuju filmove i zvuk i vizuelizuju druge tipove objekata
 - Mnogi čitači mogu da izvršavaju JavaScript kod ugrađen u HTML stranice (npr. za proveru ispravnosti unetih podataka u HTML formi)
 - Nekoliko čitača može da primenjuje kaskadne liste stilova (Cascading Style Sheets, CSS)
- Postoje manje razlike u načinima na koji pojedini čitači veb-a prikazuju HTML stranice. Npr. neki ne prikazuju slike ili ne izvršavaju Java Script kod itd.

URL



- URL-ovi (Uniform Resource Locator) koriste se na veb-u kao primarni način za imenovanje i adresiranje resursa.
- Drugim rečima, URL omogućava da se jedinstveno identifikuje adresa nekog dokumenta/fajla na veb-u.

Struktura URL



- URL se može podeliti na tri osnovna dela:
 - identifikator protokola
 - identifikator ciljnog računara i servisa na njemu (port)
 - putanja (koja može da sadrži još i parametre)
- *Struktura URL-a:*
scheme://host:port/path?parameter=value#anchor
- Primer:
<http://en.wikipedia.org/w/wiki.phtml?title=Train&action=history>
- Sam HTTP standard ne ograničava dužinu URL-a, ali to ipak čine neki stariji veb čitači i posrednički serveri.

Scheme - Identifikator protokola

- Prvi deo URL-a (scheme) služi da se identifikuje aplikacioni protokol.
- Najčešće korišćeni protokoli:
 - **http** - HyperText Transfer Protocol (protokol za prenos hiperteksta)
 - **https** - HyperText Transfer Protocol with SSL (protokol za prenos hiperteksta putem veza zaštićenim SSL protokolom)
 - **ftp** - File Transfer Protocol (protokol za prenos fajlova)

Host - identifikator ciljnog računara

- Drugi deo URL-a (*host*) identifikuje ciljni računar (server).
- **host** se može zadati preko imena ili kao IP adresa.
- Primeri:
www.w3.org
18.29.1.35

Port - identifikator servisa na ciljnom računaru

- Treći deo URL-a (port) specificira TCP port number i služi za identifikaciju servisa na ciljnom računaru.
- Na internetu postoji konvencija da se dobro poznati priključak koristi na serveru koji pruža dobro poznati tip usluge (HTTP server očekuje zahteve na priključku 80, FTP server na priključku 21, itd).
- TCP priključak (port) nije fizički uređaj, već identifikator koji služi TCP softveru i omogućava uspostavljanje više virtuelnih veza sa istom mašinom za različite aplikacije.

Path - identifikator resursa na ciljnom računaru

- Četvrti deo URL-a (path) služi za identifikovanje resursa na ciljnom računaru, i to je putanja (najčešće stvarna putanja do ciljne datoteke na serveru).
- Ako se ime ciljne datoteke izostavi podrazumeva se index.html (ovo je po defaultu za Apache, mada se može i drugačije podesiti u fajlu httpd.conf).
- Primeri:
rfc/rfc1738.html
clanovi/novi_c/ ≡ clanovi/novi_c/index.html

Dodatni parametri u URL-u

- Peti deo URL-a (parameter=value) se koristi za prosleđivanje vrednosti parametara ciljnom resursu (npr. php fajlu) putem URL-a.
- U slučaju prosleđivanja više parova parametar - vrednost, isti se razdvajaju znakom &.
- Primeri
p1=Beograd
p1=Beograd&p2=2003

Anchor - identifikator fragmenta

- Šesti deo URL-a (anchor) predstavlja identifikator fragmenta se koristi za pozicioniranje na određeno mesto u okviru dokumenta.
- Ova mogućnost se koristi u slučaju velikih dokumenata.
- Mesto "skoka" mora biti registrovano u okviru dokumenta.

Aplikacioni sloj (#1)



- Aplikacioni sloj sadrži aplikacionu logiku koja se uglavnom realizuje preko skriptova .
- Skriptovi obavljaju sledeće funkcije:
 - Obavljaju određene zadatke karakteristične za svaku aplikaciju pojedinačno (izračunavanja,...) .
 - Komuniciraju sa DBMS-om na strani veb servera.
 - Generišu HTML kod potreban za prezentaciju podataka u korisnikovom veb pregledaču.

Aplikacioni sloj (#2)



- Najveći deo aplikacione logike nalazi se u aplikacionom sloju.
- Aplikacioni sloj obavlja većinu zadataka pomoću kojih se objedinjuju ostali slojevi:
 - Obraduje podatke dobijene od korisnika pretvarajući ih u upite za čitanje ili upisivanje u bazu podataka.
 - Upravlja se strukturom i sadržajem podataka koji se prikazuju korisniku.
 - Omogućava upravljanje stanjem uz upotrebu HTTP protokola.
- PHP se nametnuo kao komponenta mnogih dinamičkih veb aplikacija srednjeg i velikog obima.

Osnovni pojmovi - PHP



- Šta je PHP?

Jezik za pisanje skriptova koji rade na serveru, namenski projektovan za upotrebu na webu.

- Početna verzija PHP-a napravljena 1994. godine.

- Aktuelna verzija: PHP ver. 5.4 (od 01.03.2012.).

- PHP je proizvod otvorenog koda.

- Personal Home Page, **PHP Hypertext Preprocessor**

- <http://www.php.net>

Osnovni pojmovi - MySQL



- Šta je MySQL?

Brz i robustan sistem za upravljanje relacionim bazama podataka.

- MySQL je višekorisnički i višenitni sistem.

- SQL (Structured Query Language) je standardni jezik za upite u bazi podataka.

- Zašto koristimo PHP i MySQL?

Mogu da rade pod svakim poznatijim operativnim sistemom, čak i kod onih manje popularnih.

Prednosti PHP-a



- Visoke performanse
- Povezivanje s velikim brojem sistema za upravljanje bazama podataka
- Ugrađene biblioteke za obavljanje velikog broja poslova
- Niska cena
- Lako se uči i upotrebljava
- Dobra podrška za objektno orijentisano programiranje
- Prenosivost
- Izvorni kod dostupan svima
- Dobra podrška u slučaju problema
- Glavni konkurenti: Perl, ASP.NET, JSP, ColdFusion

Šta je novo u PHP-u 5.0 ?



- Bolja podrška za objektno orijentisano programiranje.
- Izuzeci, koji omogućavaju skalabilnu i jednostavnu obradu grešaka.
- Komponenta SimpleXML, koja omogućava jednostavan rad sa podacima u XML formatu.

Sloj baze podataka (#1)



- Sloj baze podataka se sastoji od sistema za upravljanje bazom podataka (Database Management System - DBMS) .
- DBMS omogućava:
 - Čitanje podataka iz baze
 - Ažuriranje podataka u bazi (unos, brisanje i izmena)

Sloj baze podataka (#2)



- **Organizacija prema objektima i odnosima koje postoje u sistemu na koji se baza podataka odnosi.**
- **Integrisanost i kontrolisana redudansa:**
Krajnji cilj integrisanosti je minimalna redudansa (visešestruko pojavljivanje) podataka;
ponekad svesno želimo da ponavljamo određene podatke radi bržeg rada sa bazom podataka.

Sloj baze podataka (#3)



- **Organizacija prema potrebama korisnika:**
Podrazumeva mogućnost definisanja izvedenih slogova sa podacima.
- **Sigurnost:**
Podrazumeva efikasnu kontrolu pristupa podacima, u smislu ko može da pristupi bazi podataka, kojim podacima i šta može da radi sa tim podacima.

Sloj baze podataka (#4)



- **Konkurentnost:**

Podrazumeva mogućnost sinhronizovanog rada više korisnika istovremeno.

- **Integritet:**

Podrazumeva automatski oporavak od nasilnih prekida u toku rada koji dovode do tzv. nekonzistentnih stanja usled delimično izvršenih ažuriranja (unos, izmena i brisanja) podataka.

Sloj baze podataka (#5)



- **Import podataka:**

Baza se često uvodi kao zamena za neki postojeći sistem, i tada mora postojati mogućnost preuzimanja tih podataka.

- **Eksport podataka:**

Često se javlja potreba da se postojeća baza zameni nekom još savremenijom bazom i tada mora postojati mogućnost predaje podataka bazi podataka na koju se prelazi.

Sloj baze podataka (#6)



- **Performanse:**

Baza podataka mora da obezbedjuje maksimalni učinak u smislu najveće brzine rada uz najmanje zauzeće računarskih resursa.

- **Ekonomičnost:**

Odnos učinak-cena treba da je što niži.

- **Standardizacija:**

Standardni način opisa organizacije i operacija nad bazom obezbeđuje maksimalnu nezavisnost i trajnost korisničkog programa za rad sa bazom podataka u odnosu na promenu baze podataka.

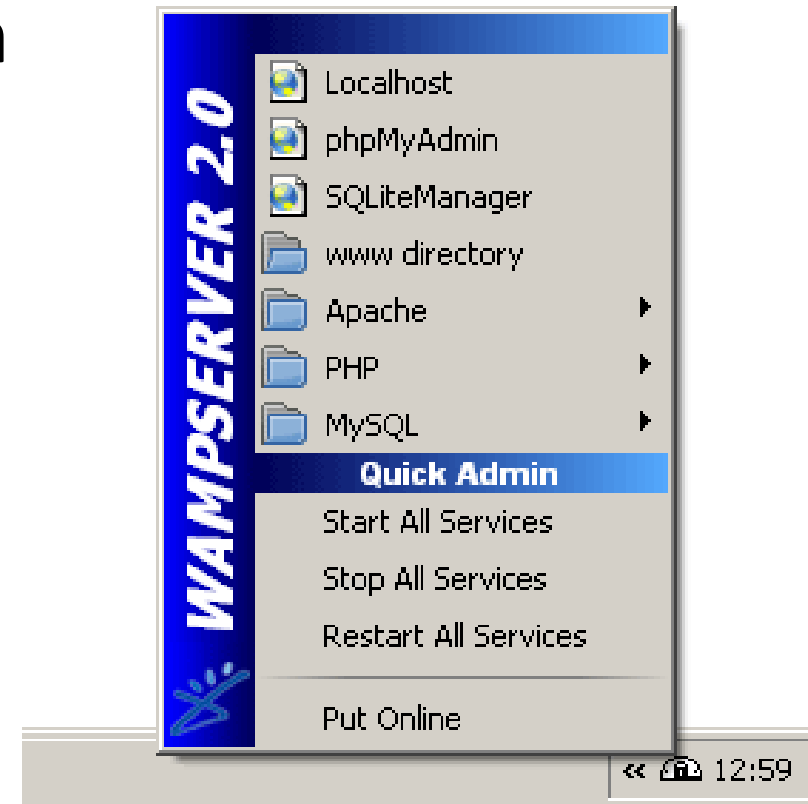
Prednosti MySQL



- Visoke performanse
<http://web.mysql.com/benchmark.html>
- Niska cena
- Lako se konfiguriše i uči
- Prenosivost
- Izvorni kod je javno dostupan
- Široko dostupna podrška u slučaju problema

Integrisane instalacije - WAMP/XAMPP

- Integrisana = sve u jednom
 - WAMP ili XAMPP
- Razdvojena:
 - Apache
 - PHP
 - MySQL



- Download Wamp: <http://www.wampserver.com/en/>
- Download XAMPP: <http://xampp.en.softonic.com/>


Pristupanje preko http://localhost

Accueil WAMP5 - Mozilla Firefox

Echier Édition Affichage Historique Marque-pages Outils ? Manuel PHP

http://localhost/

Rechercher RS Orthographe Paramètres

 Version 2.0 English Version

Configuration Serveur

Version de Apache: 2.0.61
Version de PHP: 5.2.5
Version de MySQL: 5.0.45

Extensions Chargées:

bcmath	calendar	com_dotnet	ctype
session	filter	ftp	hash
iconv	json	odbc	pcntl
Reflection	date	libxml	standard
tokenizer	zlib	SimpleXML	dom
SPL	wddx	xml	xmlreader
xmlwriter	apache2handler		

Outils

- phpinfo()
- phpmyadmin
- sqlitemanager

Vos Projets

- v2
- v2_demo

Vos Alias

- phpmyadmin
- sqlitemanager

WampServer - Powered by Anaska

Terminé

Print/A

Razdvojena instalacija



- Instalacija Apache
- Podešavanje Apache servera
- Instalacija PHP
- Podešavanje PHP konfiguracije
- Instalacija MySQL

Instalacija Apache servera (#1)

Apache HTTP Server 2.0 - Installation Wizard

Server Information

Please enter your server's information.

Network Domain (e.g. somenet.com)

Server Name (e.g. www.somenet.com):

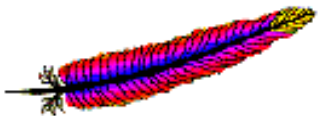
Administrator's Email Address (e.g. webmaster@somenet.com):

Install Apache HTTP Server 2.0 programs and shortcuts for:

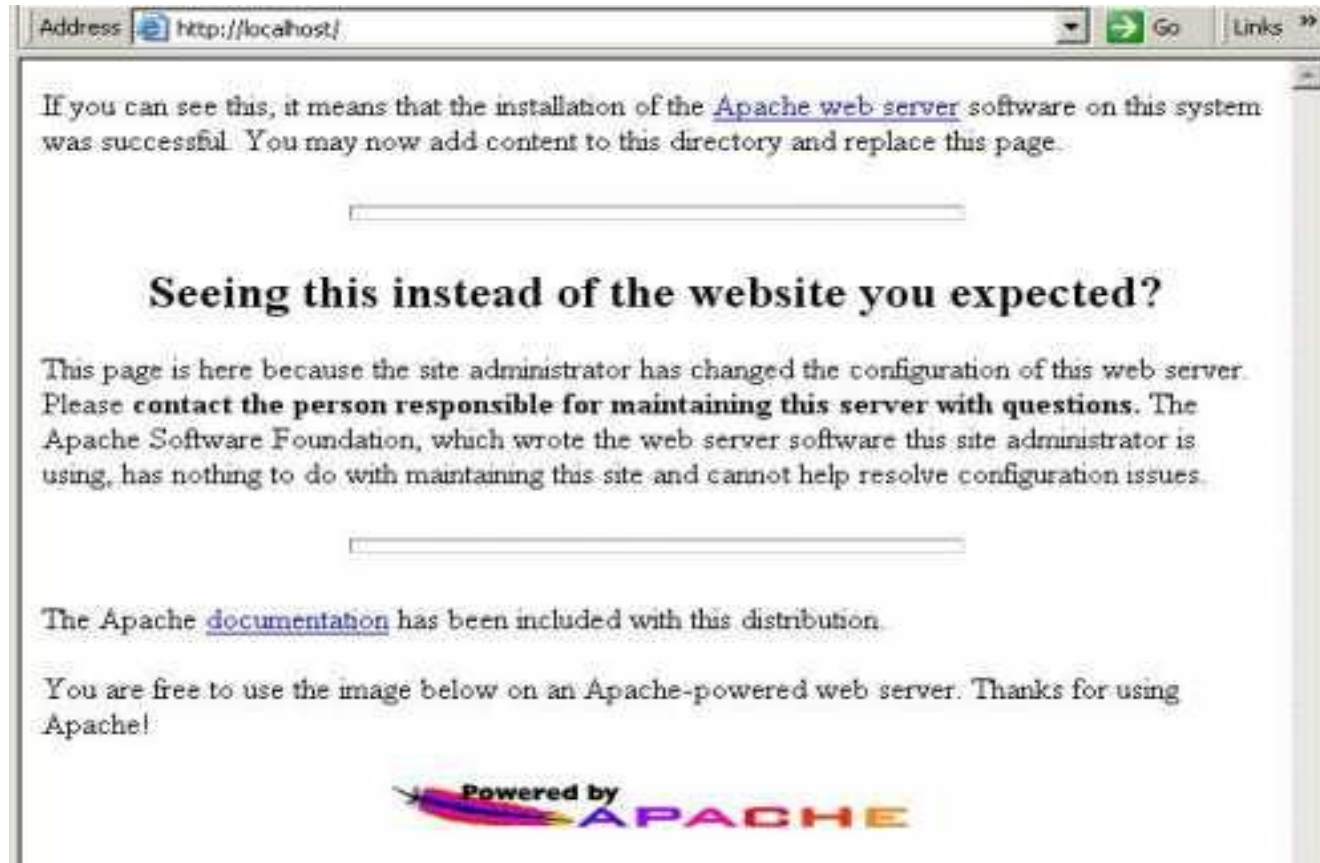
for All Users, on Port 80, as a Service -- Recommended.

only for the Current User, on Port 8080, when started Manually.

InstallShield



Instalacija Apache servera (#2)



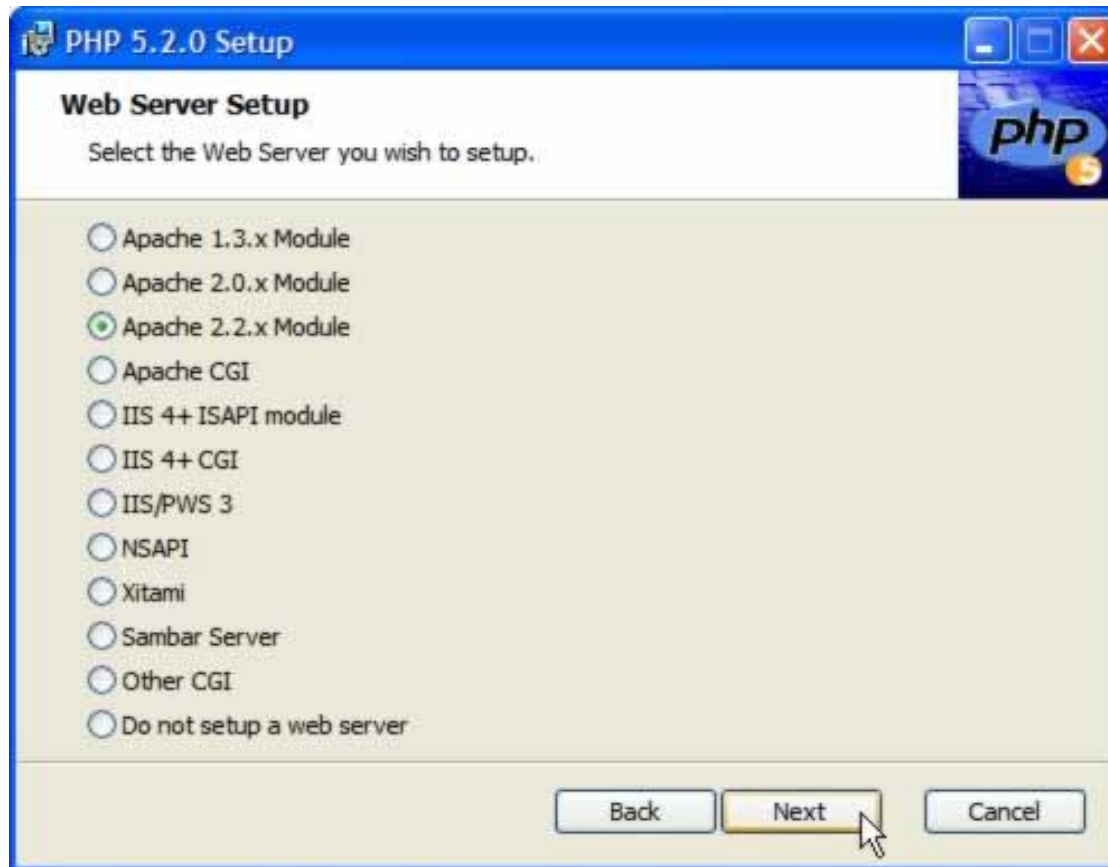
Instalacija Apache servera (#3)

- C:\Program Files\Apache Group\Apache2\conf\httpd.conf
- DocumentRoot "C:/Program Files/Apache Group/Apache2/htdocs"
- DocumentRoot "C:/www"
- Obratiti pažnju na korišćenje / i \
- DirectoryIndex index.html index.php main.php
- Restart Apache servera nakon podešavanja je obavezan.

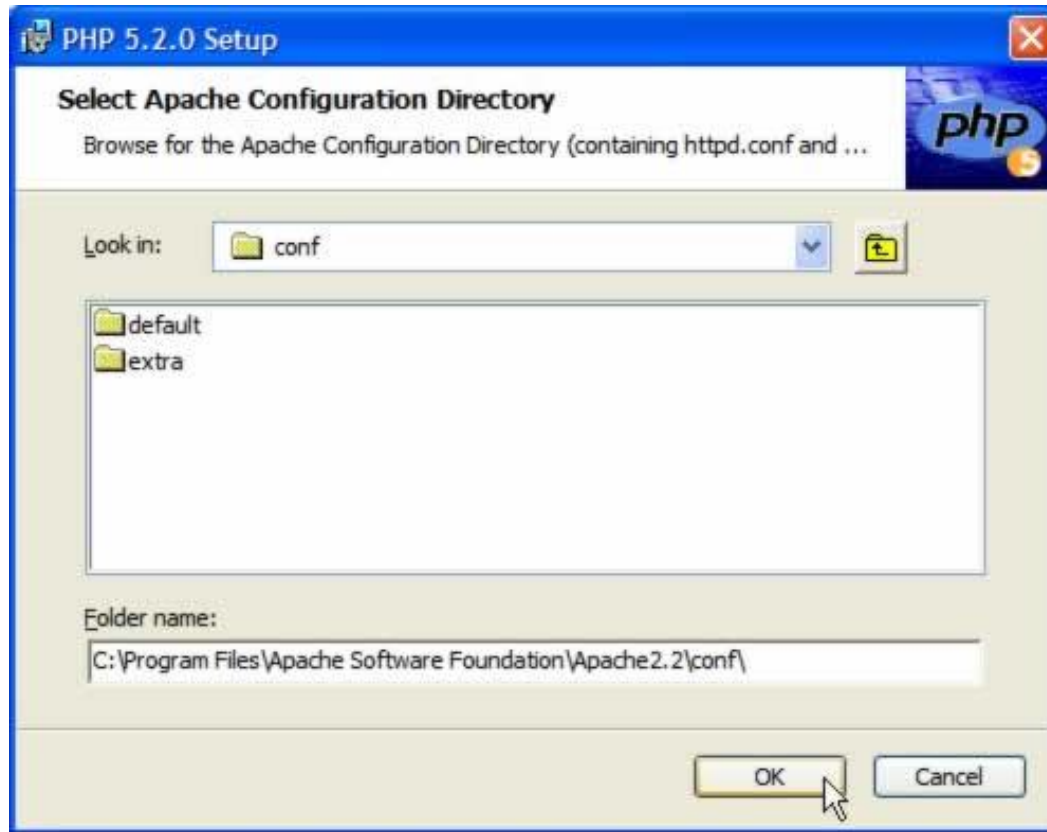
Instalacija PHP-a (#1)



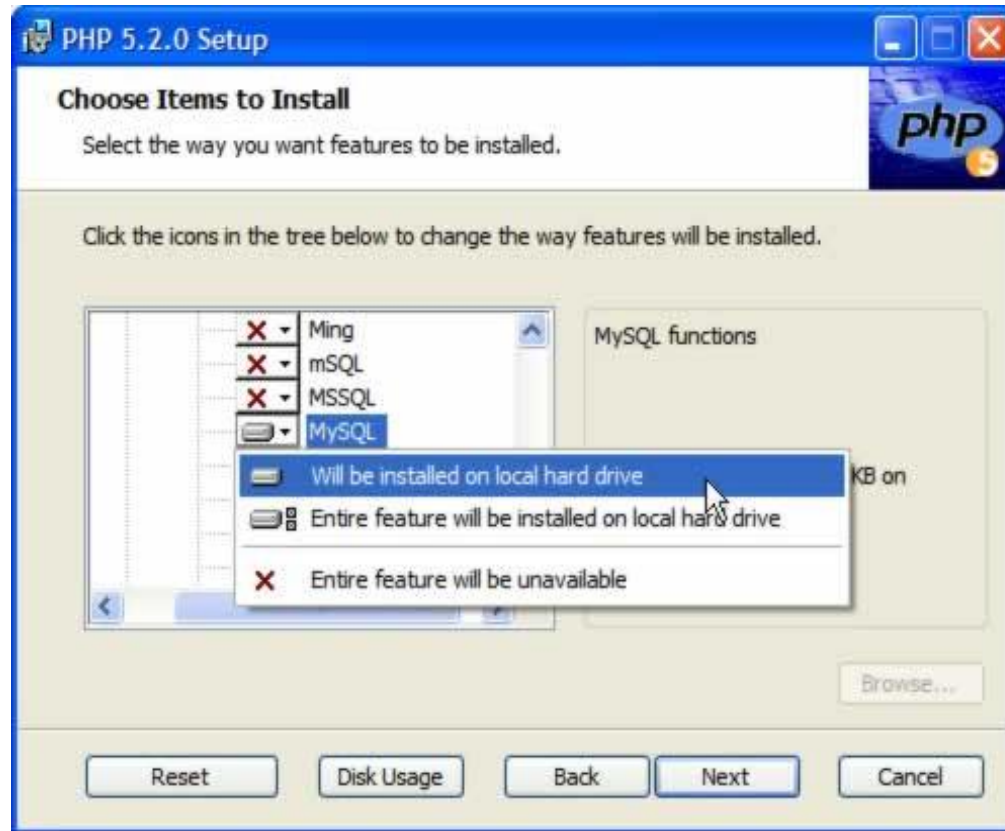
Instalacija PHP-a (#2)



Instalacija PHP-a (#3)



Instalacija PHP-a (#4)



Podešavanje Apache i PHP

For PHP 4 do something like this:

```
LoadModule php4_module "c:/php/php4apache2.dll"
```

Don't forget to copy the php4apache2.dll file from the sapi directory!

```
AddType application/x-httpd-php .php
```

For PHP 5 do something like this:

```
LoadModule php5_module "c:/php/php5apache2.dll"
```

```
AddType application/x-httpd-php .php
```

configure the path to php.ini

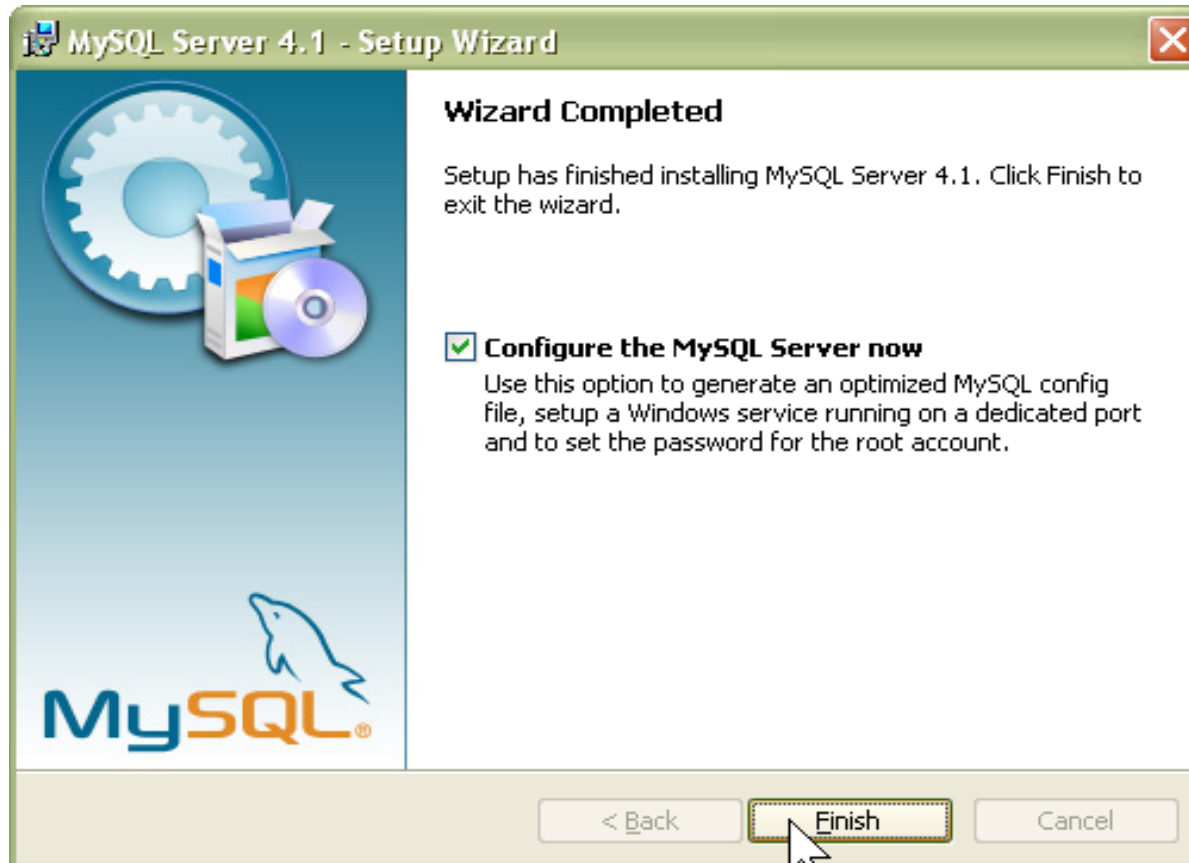
```
PHPIniDir "C:/php"
```

Podešavanje PHP-a

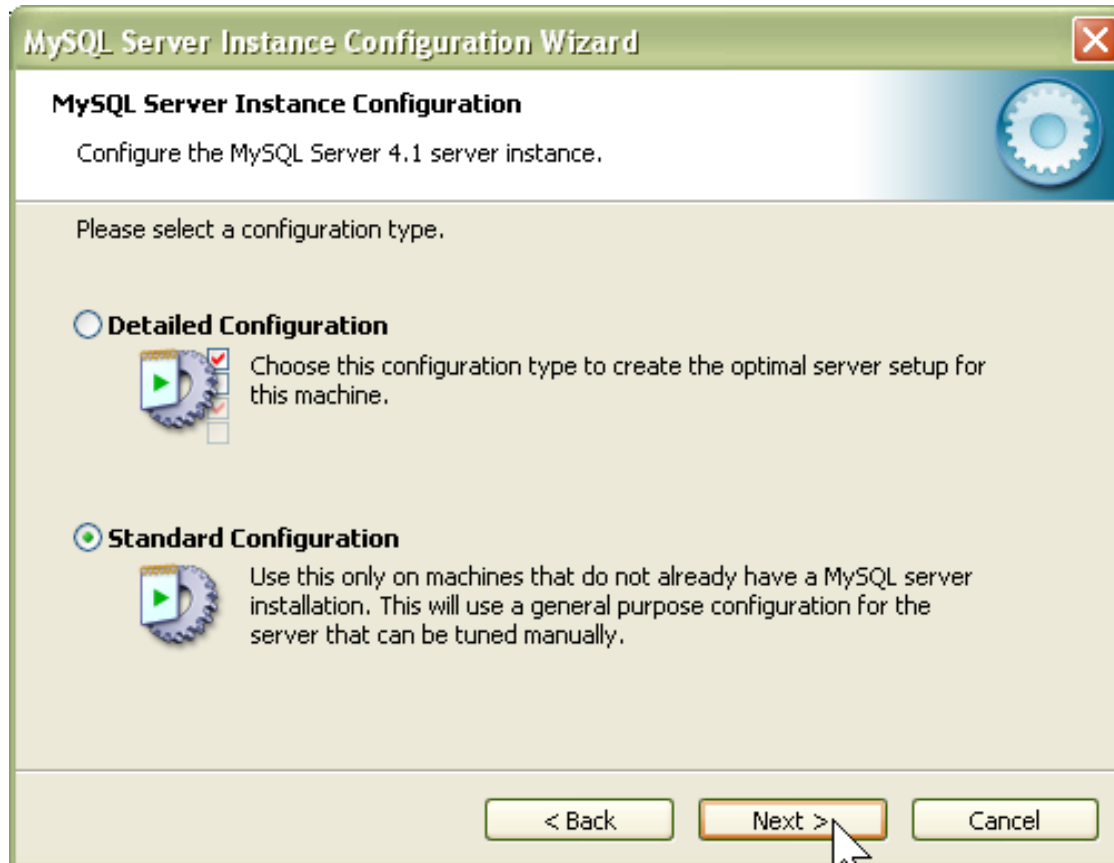


- PHPIniDir "C:/php"
- register_globals :OFF
- error_reporting
 - Razvoj: E_ALL
 - Produkcija: E_NONE
- display_errors: ON
- extension and extension_path
 - Putanja do ekstenzija
- session.save_path
 - Putanja gde da čuva podatke o sesijama
- max_execution_time
 - Default: 30

Instalacija MySQL-a (#1)



Instalacija MySQL-a (#2)



Okruženja za pisanje koda

- Eclipse Classic + Aptana Studio

- Eclipse, verzija 3.7.2:

Download:

<http://www.eclipse.org/downloads/>



- Aptana Studio, verzija 3
Eclipse Plug-in



aptana®

Download:

<http://aptana.com/downloads/start#>

- Druga opcija: Adobe Dreamweaver CS4

Instalacija plug-in u Eclipse-u

- Iz help menija odaberite **Install New Software...** da otvorite dijalog za instaliranje plug-in.
- Iskopirajte URL adresu sajta sa koga skidate plug-in u polje **Work With** i pritisnuti **Enter**.
- U datoj tabeli čekirati polje pored naziva plug-in i kliknuti na **Next**.
- Kliknuti na **Next** dugme da biste prešli na stranu za licenciranje.
- Izaberite opciju da prihvatate uslove licence i kliknite na **Finish** dugme.
- Potrebno je da resetujete Eclipse da biste nastavili korišćenje i uspešno završili proces instalacije.

1) Uvod u PHP



- PHP elementi
- Ugradnja PHP koda u HTML kod
- Promenljive i superglobalne promenljive
- Rad sa stringom i naredbe za štampanje
- Operatori
 - aritmetički, dodele, za nadovezivanje stringova, reference, operatori poređenja, logički operatori,...
- Uslovne strukture
 - if, if-else, switch
- Petlje
 - while, for, foreach, do... while

Primer 1 - HTML kod



● Neka je dat HTML kod:

```
<html>
<head>
  <title>Univerzitet u Beogradu</title>
</head>
<body>
  <h1>Elektrotehnički fakultet</h1>
  <h2>Odsek za softversko inženjerstvo</h2>
  <p>Neki tekst napisan u HTML kodu</p>
</body>
</html>
```

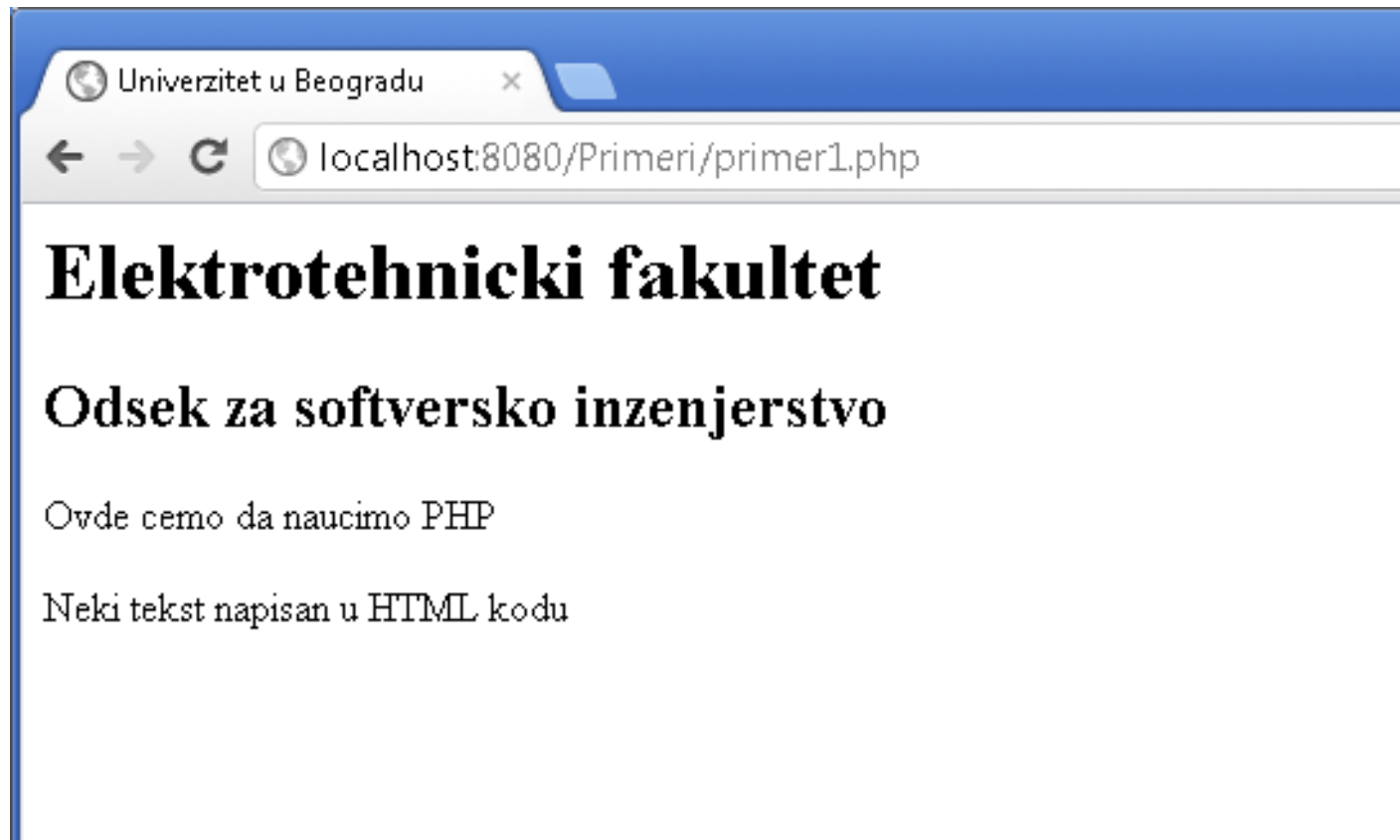
Primer 1 - PHP kod



- Sada ćemo ispod naslova, odnosno iza HTML taga `</h2>` da dodamo jedan PHP skript i da pokrenemo fajl u pregledaču veba:

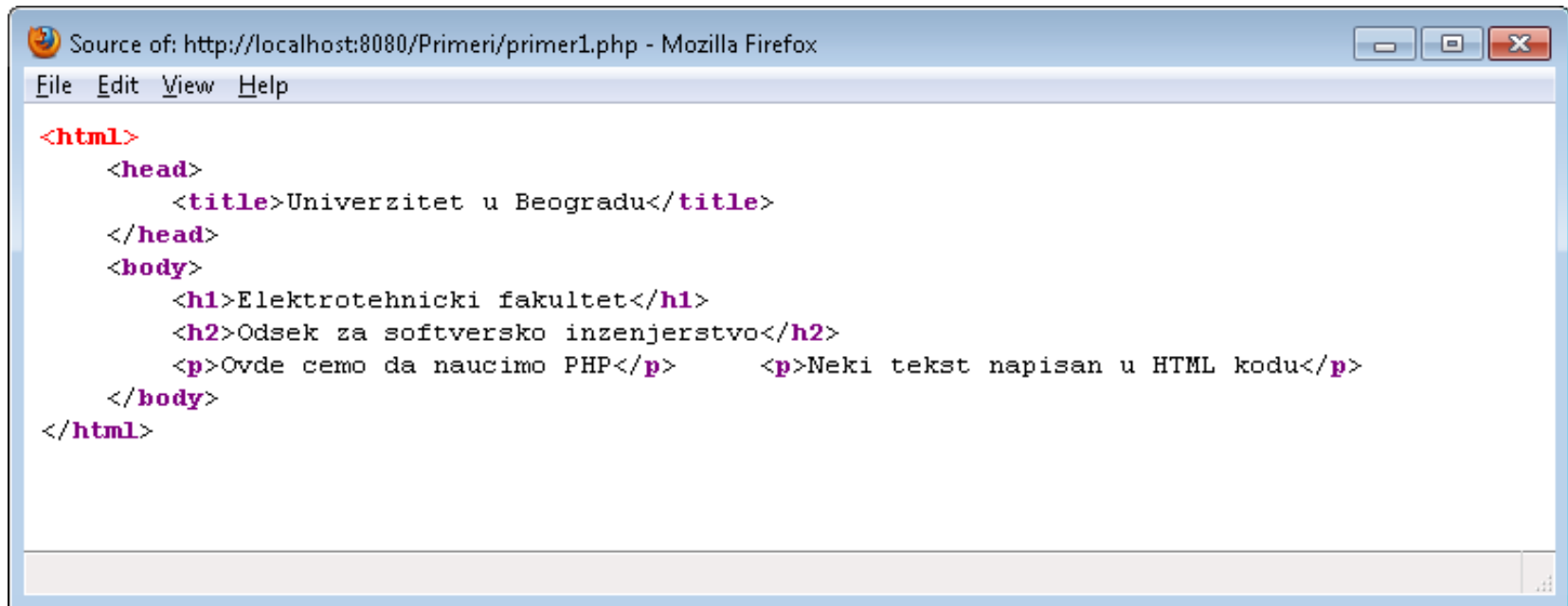
```
<?php
    echo '<p>Ovde cemo da naucimo PHP</p>';
?>
```

Primer 1 - Prikaz



Primer 1 - Izvorni kod

- Source code: PHP kod se ne vidi! Samo HTML...



```
<html>
  <head>
    <title>Univerzitet u Beogradu</title>
  </head>
  <body>
    <h1>Elektrotehnicki fakultet</h1>
    <h2>Odsek za softversko inzenjerstvo</h2>
    <p>Ovde cemo da naucimo PHP</p>    <p>Neki tekst napisan u HTML kodu</p>
  </body>
</html>
```

Ugradnja PHP koda u HTML kod

- PHP naredbe se ne vide, zato što je interpretator PHP koda zamenuje naredbe rezultatom.
- PHP se dakle dobija kao čisti HTML kod (zaključak: pregledač ne mora da razume PHP!)
- Skript jezici se razlikuju:
 - JavaScript = izvršava se kod klijenta,
 - PHP = izvršava se na veb serveru.

Elementi u PHP fajlu



U PHP fajlu mogu da postoje:

- HTML oznake (HTML tagovi),
 - PHP oznake (PHP tagovi),
 - PHP iskazi,
 - Beline i komentari.
-
- U primeru 1 najveći deo koda je HTML kod.

PHP oznake/tagovi



- Početni tag: `<?php`
označava početak PHP koda
- Završni tag: `?>`
označava kraj PHP koda
- Ovi tagovi pokazuju veb serveru šta je PHP kod, i sve između `<?php` i `?>` smatra se PHP kodom (izvan ovih oznaka veb pregledač smatra da je običan HTML kod)

Drugi PHP tagovi

- Postoje četiri stila pisanja PHP tagova:

- XML stil (standardni)

```
<?php echo 'ETF'; ?>
```

- Skraćeni stil (administratori nekad isključivo korišćenje ove oznake - opcija `short_tags`)

```
<? echo 'ETF'; ?>
```

- stil SCRIPT (slično kao JS)

```
<SCRIPT LANGUAGE='php'> echo 'ETF'; </SCRIPT>
```

- ASP stil (samo u okruženjima ASP i ASP.NET, podrazumevano ovaj stil je isključen - opcija `asp_tags`)

```
<% echo 'ETF'; %>
```

PHP iskazi

- PHP iskazi (engl. *statements*) nalaze se između početnog i završnog taga i određuju šta interpretator PHP koda treba da uradi.
- U primeru 1, komandom `echo` štampamo (ispisujemo) jedan paragraf, i rezultat izvršavanja u veb pregledaču predstavlja tekst koji smo napisali u paragrafu.
- Napomena:
Na kraju iskaza `echo` nalazi se znak `;` (tačka zarez). Taj znak razdvaja izraze u PHP-u (kao u C-u ili Javi). Izostavljanje `;` je česta sintaksna greška, koja se lako pravi, ali se lako i otkriva 😊



- Beline ili znakovi za razdvajanje: novi red, znaci razmaka i tabulacija.
- Pregledači veća zanemaruju beline i u HTML kodu i u PHP kodu.

- **Primeri istog PHP koda:**

```
echo '<h1>Dobrodosli u PHP</h1>';
```

```
i
```

```
echo '<h1>Dobrodosli      u          PHP      </h1> ';
```

```
echo 'Dobrodosli ';
```

```
echo 'na ETF! ';
```

```
i
```

```
echo 'Dobrodosli ';echo 'na ETF!';
```

Komentari



- Komentari služe kao obaveštenje osobama koje čitaju programski kod.
- Šta se piše najčešće u komentarima?
- PHP podržava komentare slično kao C i C++

- **Jednoredni komentar:**

```
echo '<h1>Dobrodosli u PHP</h1>'; // komentar1  
echo '<h1>Dobrodosli u PHP</h1>'; # komentar2
```

- **Višeredni komentar:**

```
echo '<h1>Dobrodosli u PHP</h1>';  
/* Primer komentara  
   Drazen Draskovic  
   Izmenjen fajl: 01.03.2012.  
   Opis: Skript opisuje uvodnu lekciju.  
*/
```


Promenljive



- Svaka promenljiva (engl. *variable*) u PHP-u počinje znakom za dolar \$
- Nemojte da se nervirate, izostavljanje \$ je uobičajena greška kod početnika na PHP-u 😊

String



- Predstavlja niz karaktera
- Deklaracije
 - apostrofi 'Primer1'
 - navodnici "Primer2"
 - pomoćnih dokumenata <<< When_Will_It_End
- Bilo koja promenljiva se unutar stringa sa navodnicima pretvara se u string i izvršava, dok se kod apostrofa ništa ne dešava!

```
$a=17;  
echo "Presli smo $a slajdova"; //stampa se 17  
echo 'Presli smo $a slajdova'; //stampa se $a
```

Štampanje - echo, print



- **Mogu se koristiti:**

```
echo 'zdravo';
```

```
echo 'zdravo', 'pozdrav';
```

```
echo ('zdravo');
```

```
print 'zdravo';
```

```
print ('zdravo');
```

Štampanje heredoc sintaksa (<<<)

- Od PHP4 dodata je heredoc sintaksa, koja je poznata korisnicima Perla.
- Heredoc sintaksa omogućava zadavanje dužih znakovnih podataka u lako razumljivom obliku, tako što se zada i graničnik kraja:

```
echo <<<kraj
    prvi red
    drugi red
    treci red
kraj
```

- Graničnik može biti bilo šta, bitno je samo da se ne pojavljuje u tekstu!

Identifikatori

- Identifikatori su imena promenjivih, ali i imena funkcija i klasa.
- Identifikatori mogu da budu bilo koje dužine i mogu da se sastoje od slova, brojeva i _
- Identifikatori ne mogu da počinju cifrom.
- U PHP-u pravi se razlika između malih i velikih slova u imenima identifikatora (\$pr i \$Pr nije isto).
- Izuzetak:
Imena funkcija mogu se pisati i malim i velikim slovima!!!

Deklaracije



- U PHP-u ne treba da se deklarirše promenljiva, pre nego što se upotrebi.
- Promenljiva će biti napravljena kada joj prvi put dodelimo vrednost:
`$ime = "Jasna";`

Tipovi promenljivih



- *integer* - celobrojni tip, koristi se za cele brojeve
- *double* ili *float* - pokretni zarez dvostruke preciznosti, koristi se za realne brojeve
- *string* - znakovni tip, koristi se za znakovne podatke
- *boolean* - logički, koristi se za podatke tipa tačno ili netačno
- *array* - niz, koristi se za čuvanje više podataka istog tipa
- *object* - objekat, koristi se za čuvanje primerka (instance) jedne klase

Specijalni tipovi podataka



- Dva specijalna tipa:
 - NULL - promenljive kojima nije dodeljena vrednost, koje su nedefinisane ili kojima je izričito dodeljena vrednost NULL.
 - Resurs - neke ugrađene funkcije (npr. za rad sa bazama podataka) vraćaju promenljive tipa resurs, koje predstavljaju spoljne resurse (veza sa bazom podataka).

Provera tipa podataka



- PHP je jezik sa slabom proverom tipa podataka.
- Tip podataka se ne navodi eksplicitno kod promenljivih u PHP-u, već se određuje na osnovu vrednosti koja joj je dodeljena:

```
$ime = "Jasna";           //ime je string  
$broj = 10;               //broj je integer  
$broj = "Desetka";       //broj je sada string
```

Konverzija tipa podataka



- Operator za konverziju omogućava privremenu promenu tipa promenljive ili vrednosti. Postupak se odvija identično kao u C-u.

- **Primer:**

```
$broj = 10; //integer  
$realbroj = (double) $broj; //double
```

Promenljiva promenljive

- PHP podržava tip - promenljiva promenljive (engl. *variable variable*).
- Takve promenljive omogućavaju da ime promenljive menjamo dinamički.

- **Primer:**

```
$ocena = "Deset";  
$$ocena = 10;
```

što je ekvivalentno izrazu:

```
$Deset = 10;
```

Deklarisanje i upotreba konstanti

- Konstanta predstavlja određenu vrednost, isto kao i promenljiva, ali se ta vrednost zadaje jednom i potom se više ne može menjati.
- Primer:

```
define ('BEOGRAD', 11000);
```
- Kako razlikujemo konstante i promenljive?
 - Imena konstanti pišu se velikim slovom! (kao u C)
 - Ispred konstante se ne piše simbol dolar \$
- Upotreba konstante - samo se pozove njeno ime:

```
echo BEOGRAD; //izlaz: 11000
```

Opseg važenja promenljive (1)

- Izraz opseg važenja (engl. *scope*) odnosi se na mesta u skriptu na kojima je data promenljiva vidljiva.
- U PHP-u postoji 6 mogućih opsega važenja:
 - Ugrađene globalne promenljive
- vidljive u celom skriptu.
 - Konstante imaju globalnu vidljivost,
pa se upotrebljavaju i unutar i izvan funkcija.
 - Globalne promenljive, deklarirane u skriptu, vidljive su svuda u skriptu, ali ne i unutar funkcija.

Opseg važenja promenljive (2)

(nastavak)

- Promenljiva koju napravite unutar funkcije, a ima isto ime kao promenljiva koja je deklarirana kao globalna, upućuje na globalnu promenljivu istog imena.
- Promenljive koje napravite unutar funkcije, a deklarirate kao statičke promenljive, ne vide se izvan funkcija, ali čuvaju svoju vrednost između dva izvršavanja funkcije.
- Promenljive koje napravite unutar funkcije lokalne su za funkciju i prestaju da postoje kada prestane izvršavanje funkcije.

Opseg važenja promenljive (3)

- Od PHP-a 4.1 nadalje, za nizove `$_GET` i `$_POST`, i za neke druge posebne promenljive, važe drugačija pravila za opseg važenja.
- To su superglobalne promenljive, koje se svuda vide, i u funkcijama i van njih.

Superglobalne promenljive

- **\$GLOBALS** - niz svih globalnih promenljivih
- **\$_SERVER** - niz serverskih promenljivih
- **\$_GET** - niz promenljivih prosleđenih skriptu metodom GET
- **\$_POST** - niz promenljivih prosleđenih skriptu metodom POST
- **\$_COOKIE** - niz kolačića
- **\$_FILES** - niz promenljivih koje se odnose na fajlove poslate sa klijentskog računara
- **\$_ENV** - niz promenljivih okruženja
- **\$_REQUEST** - niz svih promenljivih koje je korisnik poslao
- **\$_SESSION** - niz promenljivih sesije

Aritmetički operatori jezika PHP

- Aritmetički operatori su znakovi za računske operacije.

Operator	Ime	Primer
+	sabiranje	$\$a + \b
-	oduzimanje	$\$a - \b
*	množenje	$\$a * \b
/	deljenje	$\$a / \b
%	modulo (mod)	$\$a \% \b

Primer

`$a = 16;`

`$b = 10;`

`$rezultat = $a + $b; // rezultat je 26`

`$rezultat = $a % $b; // rezultat je 6`

Operatori nadovezivanja



- Operator nadovezivanja znakovnih vrednosti (.) može se koristiti za spajanje dve ili više znakovnih vrednosti (slično kao sabiranje dva ili više brojeva).

- **Primer**

```
$a = "Drazen ";
```

```
$b = 'casove';
```

```
$rezultat = $a." drzi ".$b;
```

Promenljiva rezultat će u ovom slučaju sadržati znakovnu vrednost: "Drazen drzi casove".

Operatori dodele



- Osnovni operator dodele (=)
- Čita se kao “dobija vrednost”
- Na primer: `$ukupno=3; //ukupno dobija vred.3`
- Za sve operatore dodele važi sledeće pravilo:
vrednost celog iskaza dodele je vrednost koja je dodeljena operandu na levoj strani.
- Na primer: `$b=6+($a = 5); //b dobija vred. 11`
- Zagrade se upotrebljavaju da se poveća prioritet izračunavanja podizraza. Princip je isti kao u matematici!!!

Kombinovani operatori dodele

Operator	Upotreba	Ekvivalentan izrazu
+=	$\$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	$\$a = \$a . \$b$

Kombinovani operatori dodele postoje za svaki aritmetički operator i za operator nadovezivanja znakovnih vrednosti

Operatori ++ i --



- Prefiksno uvećanje ++ i umanjeње --
- Primer (prefiksno uvećanje)
\$a = 4;
echo ++\$a;
// rezultat je 5, a=5
- Sufiksno uvećanje ++ i umanjeње --
- Primer (sufiksno uvećanje)
\$a = 4;
echo \$a++;
// rezultat je 4, a=5
- Slično se ponašaju i operatori umanjeња

Reference



- Operator referenca (&, ampersand) se može koristiti u kombinaciji sa operatorima dodele.

- Primer:

```
$a = 5;
```

```
$b = $a;
```

```
$a = 7; // b je i dalje 5
```

```
- - - - -
```

```
$a = 5;
```

```
$b = &$a;
```

```
$a = 7; // i a i b su sada 7
```

- Poništavanje definicije jedne od promenljivih
`unset($a);` Promenljiva `b` ne menja vrednost, ali prekida se veza između `$a` i `7`.

Operatori poređenja



Operator	Ime	Upotreba
==	jednako	$\$a == \b
===	identično	$\$a === \b
!=	različito	$\$a != \b
!==	nije identično	$\$a !== \b
<>	različito	$\$a <> \b
<	manje od	$\$a < \b
>	veće od	$\$a > \b
<=	manje od ili jednako	$\$a <= \b
>=	veće od ili jednako	$\$a >= \b

Logički operatori



Operator	Ime	Upotreba	Rezultat
!	negacija	!\$b	Vraća <i>true</i> ako je \$b <i>false</i> i obrnuto
&&	konjunkcija	\$a && \$b	Vraća <i>true</i> ako su \$a i \$b <i>true</i>
	disjunkcija	\$a \$b	Vraća <i>true</i> ako su \$a i \$b ili oba <i>true</i>
and	konjunkcija	\$a and \$b	Isto kao &&, ali nižeg prioriteta
or	disjunkcija	\$a or \$b	Isto kao , ali nižeg prioriteta

Na primer koristimo pri utvrđivanju da li je vrednost promenljive \$a između 0 i 100. To omogućava operator logičke konjunkcije (AND) :

```
$a >= 0 && $a <= 100
```


Operatori nad bitovima



Operator	Ime	Upotreba	Rezultat
&	konjunkcija	$\$a \& \b	Bitovi koji su aktivni u $\$a$ i $\$b$
	disjunkcija	$\$a \b	Bitovi koji su aktivni u $\$a$ ili $\$b$
~	negacija	$\sim \$a$	Bitovi koji su aktivni u $\$a$ nisu u $\$b$ i obrnuto
^	isključiva disj.	$\$a \wedge \b	Bitovi aktivni ili u $\$a$ ili u $\$b$, ali ne u oba
<<	pomeranje ulevo	$\$a \ll \b	Pomera bitove $\$a$ ulevo za $\$b$ mesta
>>	pomeranje udesno	$\$a \gg \b	Pomera bitove $\$a$ udesno za $\$b$ mesta

Još neki operatori



- Uslovni operator (slično kao if-else)
uslov ? uslov_true : uslov_false
(\$ocena > 5 ? ' Polozio ' : ' Pao ');

- Operator zanemarivanja greške
\$a = @(27/0)

Bez operatora @, izvršno okruženje bi generisalo upozorenje “delite nulom”.

Ako upotrebite ovaj operator, greška se zanemaruje (ali kad se zanemaruju upozorenja o greškama, trebalo bi da napišete kod za obradu grešaka).

Operatori za rad s nizovima



Operator	Ime	Upotreba	Rezultat
+	unija	$\$a + \b	niz od svih elemenata $\$a$ i $\$b$
==	jednako	$\$a == \b	<i>true</i> , ako imaju jednake elemente
===	identično	$\$a === \b	<i>true</i> , ako imaju jednake elemente u jednakom redosledu
!=	različito	$\$a != \b	<i>true</i> , ako je $\$a$ različit od $\$b$
<>	različito	$\$a <> \b	<i>true</i> , ako je $\$a$ različit od $\$b$
!==	nije identično	$\$a !== \b	<i>true</i> , ako $\$a$ nije identičan $\$b$

Operator za utvrđivanje tipa - **instanceof** :

```
class nekaKlasa { };  
nekiObjekat = new nekaKlasa();  
if (nekiObjekat instanceof nekaKlasa) echo "nekiObjekat je primerak klase  
nekaKlasa";
```

Prioriteti



- Svaki operator ima određeni prioritet, ili redosled kojim se izračunava.
- Svaki operator ima i asocijativnost (redosled izračunavanja za operatore jednakih prioriteta):
 - leva asocijativnost (slevo nadesno)
 - desna asocijativnost (zdesna nalevo)
 - nije bitna asocijativnost ($n \setminus b$)

Prioriteti - Tabela (1)



Asocijativnost	Operatori
leva	,
leva	or
leva	xor
leva	and
desna	print
leva	= += -= *= /= ,= %= &= = ^= ~= <<= >>=
leva	?:
leva	
leva	&&
leva	
leva	^
leva	&

najmanji
prioritet



nastavak tabele prioriteta na sledećem slajdu

Prioriteti - Tabela (2)



Asocijativnost	Operatori
n\b	== != ===
n\b	< <= > >=
leva	<< >>
leva	+ - .
leva	* / %
desna	! ~ ++ -- (int) (double) (string) (array) (object) @
desna	[]
n\b	new
n\b	()



najveći
prioritet

Funkcije za rad s promenljivama

- `string gettype (promenljiva)`
 - Utvrđuje tip promenljive koju joj prosledite i vraća znakovnu vrednost s imenom tipa (npr. boolean, integer, double, string array, resource, object ili NULL).
- `int settype (promenljiva, tip)`
 - Menja tip promenljive koju joj prosledite u novi tip naveden u obliku znakovne vrednosti kao drugi argument.

Funkcije za proveru tipova podataka

● Funkcije za ispitivanje određenih tipova podataka:

- `is_array`
- `is_double, is_float, is_real` //ista f-ja
- `is_long, is_int, is_integer` //ista f-ja
- `is_string`
- `is_object`
- `is_resource()`
- `is_null()`
- `is_scalar()`
- `is_numeric()`
- `is_callable` /*Ispita da li vrednost promenljive ima postojeće funkcije.*/

Ispitivanje stanja promenljive

- `boolean isset (promenljiva)`
 - Vraća `true`, ako je promenljiva definisana, u suprotnom vraća `false`
- `void unset (promenljiva)`
 - Poništava definiciju promenljive koju joj prosledite
- `boolean empty (var)`
 - Ispituje da li postoji promenljiva i da li ima vrednost koja nije nula, odnosno nije prazna i vraća `true` ili `false`

Uslovne strukture - iskaz if

- Ako je uslov ispunjen, izvršava se blok koda koji sledi iza if, u suprotnom se preskače

```
if(uslov $ocena == 5)  
echo 'Pali ste ispit!<br/>';  
//ako je uslov true biće izvršen  
//iskaz echo
```

Blok naredbi



- Često unutar jednog uslovnog iskaza, npr if, treba da se izvrši više iskaza:

```
if ($ocena > 5)
{
    echo '<font color=red>';
    echo 'Polozili ste ispit!<br/>';
    echo '</font>';
}
```

blok

Uvlačenje koda



- Napomena:

Kao što smo naučili PHP zanemaruje praznine u kodu.

Radi bolje čitljivosti, trebalo bi da uvlačite iskaze. Uvlačenje se obično primenjuje da biste lakše uočili redove koji će biti izvršeni ako su ispunjeni uslovi, iskaze koji su grupisani u blokove i iskaze koji su deo petlji ili funkcija.

Iskazi else, elseif

- ```
if ($uslov == 0) {
```

- ```
...
```

- ```
 } else {
```

- ```
...
```

- ```
}
```

- ```
if ($kolicina < 10)
```

- ```
 $popust = 0;
```

- ```
elseif ($kolicina >=10 && $kolicina<=99)
```

- ```
 $popust = 10;
```

- ```
elseif ($kolicina >100)
```

- ```
 $popust = 20;
```

**Reč elseif može da se kuca i ovako i sa razmakom (else if), oba oblika su ispravna!**

# Iskaz switch



- Slično kao if, osim što kod switch, uslov može imati više različitih vrednosti, koje moraju biti skalarnog tipa (integer, string ili float).

```
switch ($navijac) {
 case 'c': echo '<p>Crvena Zvezda</p>';
 break;
 case 'p': echo '<p>Partizan</p>';
 break;
 default: echo '<p>Ne navija za klub</p>';
 break;
}
```

# Petlje



- Ponavljanje nekih akcija (0 ili više puta)
- Petlja WHILE
- Petlje FOR i FOREACH
- Petlja DO .. WHILE

# Petlja while



```
while (uslov) izraz;
```

- Slično kao IF, samo se blok ne izvršava kad je uslov ispunjen, nego dokle god je uslov ispunjen.

```
$brojac=0;
while ($brojac<=5) {
 echo $brojac."
";
 $brojac++;
}
```



# Petlje for i foreach



```
for (izraz1; uslov; izraz2) izraz3;
for ($i=1; $i<=$br_ljudi; $i++) {
 $temp="ime$i";
}
```

- Petlja foreach se koristi za rad sa nizovima.

# Petlja do...while



```
do
```

```
 izraz;
```

```
while (uslov);
```

- **uvek se izvrši najmanje jednom (nezavisno od uslova)**

- `$brojac=100;`

```
do {
```

```
 echo $brojac."
"; }
```

```
while ($brojac < 1);
```

# Izlazak iz upravljačke strukture

- Postoje tri načina da se prekine izvršavanje bloka koda:
  - break - iskakanje iz petlje
  - continue - iskakanje na novu iteraciju petlje
  - exit - prekidanje izvršavanja celog PHP skripta

```
if ($broj_ispita == 0)
{
 echo 'Nemate ispite za prijavljivanje!';
 exit;
}
```

# Alternativni oblici sintakse

- Za sve navedene upravljačke strukture postoji alternativni oblik sintakse:
  - Početna vitičasta zagrada ( { ) zamenjuje se dvotačkom ( : )
  - Završna vitičasta zagrada ( } ) zamenjuje se novom ključnom rečju, koja će biti `endif`, `endswitch`, `endwhile`, `endfor` ili `endforeach`, u zavisnosti koja upravljačka struktura je u pitanju.

```
if ($stanje == 0) {
 echo "Nemate dovoljno novca."
 exit;
}
```

```
if ($stanje == 0) :
 echo "Nemate dovoljno novca."
 exit;
endif;
```

# Primer 2

- Napraviti narudžbenicu lekova u online apoteci, kao na slici:

| Artikal                               | Kolicina                                                                                                    |
|---------------------------------------|-------------------------------------------------------------------------------------------------------------|
| Andol                                 | <input type="text"/>                                                                                        |
| Aspirin                               | <input type="text"/>                                                                                        |
| Vitamin C                             | <input type="text"/>                                                                                        |
| Kako ste saznali za nasu apoteku?     | <input type="text" value="Ja sam redovan kupac"/>                                                           |
| <input type="button" value="Naruci"/> | <ul style="list-style-type: none"><li>Ja sam redovan kupac</li><li>TV reklama</li><li>Halo oglasi</li></ul> |

# Primer 2



- Kada kupac odredi količine i naruči lekove, kao potvrda treba da mu se odštampa fiskalni račun sa sledećim podacima:
  - a) datum i vreme kada su lekovi naručeni  
**(DINAMIČKI SADRŽAJ!!!)**
  - b) ukupna količina naručenih lekova
  - c) količina po stavkama
  - d) ukupna cena računa bez poreza i sa porezom; stopa poreza je 8%;
  - e) “Hvala! Dođite nam ponovo!” ako nije redovni kupac

# Primer 2 - Rezultat



## **Apoteka - narudzbina**

### **Fiskalni racun**

Roba narucena u 18:47, 29th April

Porucili ste:

Ukupna kolicina: 6

1 andol

2 aspirin

3 vitamin C

Ukupno bez poreza: 285.00 dinara

Ukupno sa porezom: 307.80 dinara

Hvala! Dodjite nam ponovo!

# Kako pristupati elementima forme?

● Postoje 3 načina, koji nemaju naziv:

● kratki stil\* `$kolicina`

● srednji stil `$_POST[' kolicina ']`

● dugi stil\*\* `$HTTP_POST_VARS[' kolicina']`

\* ovaj stil je praktičan, ali zahteva da se parametar `register_globals` postavi na `on` (da li ima standardno vrednost `on` ili `ne`, zavisi od verzije PHP-a, od verzije 4.2.0 ovaj parametar ima vrednost `off`, zato je bolje koristiti srednji stil (upotrebljiv tek od verzije 4.1.0)

\*\* najopširniji - zastareo stil



# \$\_POST, \$\_GET i \$\_REQUEST

- \$\_POST i \$\_GET su **superglobalni nizovi**.

- Jedan od nizova, ili \$\_POST ili \$\_GET, sadržiće vrednosti svih polja forme.

Koji niz će biti upotrebljen zavisi od metode POST/GET koja se koristi u formi:

```
<FORM NAME="forma" METHOD="POST" ACTION="prva.php">
```

```
<FORM NAME="forma" METHOD="GET" ACTION="prva.php">
```

- Svim poljima na `prva.php` se može pristupiti preko:

```
$_POST['naziv_polja'] ili
```

```
$_GET['naziv_polja']
```

- U nizu \$\_REQUEST biće dostupni svi podaci, bilo da su poslani preko POST ili GET metode.

# Rešenje (1)



```
<?php
 // kreiranje kracih imena varijabli
 $kolicina1 = $_POST['kol1'];
 $kolicina2 = $_POST['kol2'];
 $kolicina3 = $_POST['kol3'];
 $nadji = $_POST['nadji'];
?>
<html>
<head>
 <title>Online apoteka</title>
</head>
<body>
```

# Rešenje (2)

```
<h1>Apoteka - narudzbina</h1>
```

```
<h2>Fiskalni racun</h2>
```

```
<?php
```

```
echo '<p>Roba narucena u ';
```

```
echo date('H:i, jS F');
```

```
echo '</p>';
```

```
echo '<p>Porucili ste: </p>';
```

```
$ukupno = 0;
```

```
$ukupno = $kolicina1 + $kolicina2 + $kolicina3; //sabiranje
```

```
echo 'Ukupna kolicina: '.$ukupno.'
';
```

# Upotreba funkcije date()



- Funkcija `date()` očekuje kao argument znakovni niz, koji predstavlja format rezultata koji želite.
- Svako slovo u argumentu predstavlja jedan element datuma i vremena. Na primer:
  - H je oznaka za sat u 24-časovnom formatu
  - i je oznaka za minute, sa vodećom cifrom 0 za jednocifrene (0-9)
  - j je dan u mesecu, bez vodeće cifre 0
  - S predstavlja redni sufiks na engleskom (st/nd/rd/th)
  - F je pun naziv meseca

# Rešenje (3)



```
if($ukupno == 0) {
 echo 'Niste kupili nista!
'; }
else {
 if ($kolicina1>0)
 echo $kolicina1.' andol
';
 if ($kolicina2>0)
 echo $kolicina2.' aspirin
';
 if ($kolicina3>0)
 echo $kolicina3.' vitamin C
';
 echo '
';
}
$ukupna_cena = 0.00;
```

# Rešenje (4)



```
define('ANDOLCENA', 10);
define('ASPIRINCENA', 100);
define('VITCCENA', 25);
$ukupna_cena = $kolicina1 * ANDOLCENA //mnozenje
 + $kolicina2 * ASPIRINCENA
 + $kolicina3 * VITCCENA;

echo 'Ukupno bez poreza: '.number_format($ukupna_cena,2).'
 dinara
';

$porez = 0.08; // porez je 8%
$ukupna_cena = $ukupna_cena * (1 + $porez); //mnozenje
echo 'Ukupno sa porezom: '.number_format($ukupna_cena,2).'
 dinara
';
```

# Rešenje (5)



```
if($nadji == 'a')
 echo '<p>HVALA!</p>';
else
 echo '<p>Hvala! Dodjite nam ponovo!</p>';
```

```
?>
</body>
</html>
```

## 2) Rad sa datotekama

- Snimanje i obrada datoteka
- Otvaranje datoteke
- Upisivanje u datoteku
- Zatvaranje datoteke
- Ostale korisne funkcije za rad sa datotekama
- Zaključavanje datoteka



# Snimanje i obrada datoteka

- Podatke možemo da skladištimo na 2 načina:
  - u običnu datoteku (engl. *flat file*)
  - u bazu podataka (engl. *database*)
- Obična datoteka najčešće tekstualna.
- Postupak upisivanja podataka u datoteku:
  1. Otvaranje datoteke. Ako ne postoji, treba je napraviti.
  2. Upisivanje podataka u datoteku.
  3. Zatvaranje datoteke.
- Postupak čitanja podataka datoteke:
  1. Otvaranje datoteke.
  2. Čitanje podataka iz datoteke.
  3. Zatvaranje datoteke.

# Otvaranje datoteke (1)



## ● Funkcija fopen

```
$fp = fopen("$DOCUMENT_ROOT/../../www/02/naruciti.txt", 'w');
```

## ● Na početku skripta mora da postoji:

```
$DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
```

## ● Prvi parametar: lokacija datoteke

- označava roditeljski direktorijum osnovnog direktorijuma za dokumente (iz bezbednosnih razloga)

## ● Drugi parametar: režim u kome se datoteka otvara

- označava način na koji ćemo koristiti otvorenu datoteku (omogućava operativnom sistemu da utvrdi kako da odgovori na zahtev drugih korisnika ili skriptova za pristup datoteci)

# Otvaranje datoteke (2)



- Prilikom otvaranja morate odgovoriti na 3 pitanja:
  - Da li otvarate datoteku samo za čitanje, samo za upisivanje ili i za jedno i za drugo?
  - Ako upisujete: da li upisujete podatke preko postojećeg sadržaja ili dodajete novi sadržaj na kraj datoteke?
  - Ako pokušavate da upišete podatke u datoteku na sistemu koji razlikuje binarne i tekstualne fajlove, koji tip ćete koristiti?

# Otvaranje datoteke (3)



## ● Režimi za otvaranje:

- r (otvara datoteku za čitanje, od početka)
- r+ (otvara datoteku za čitanje i upisivanje, od početka)
- w (otvara datoteku za upisivanje, od početka; ako datoteka postoji, postojeći sadržaj se briše, ako ne postoji, sistem pokušava da je napravi)
- w+ (otvara datoteku za upisivanje i čitanje)
- x (otvara datoteku za upisivanje; kada datoteka postoji sistem je ne otvara, funkcija vraća false, a PHP generiše upozorenje)
- x+ (otvara datoteku za upisivanje i čitanje; ukoliko datoteka postoji, sistem je ne otvara, funkcija vraća false, a PHP generiše upozorenje)
- a (otvara datoteku samo za dodavanje od kraja postojećeg sadržaja)
- a+ (otvara datoteku za dodavanje i čitanje od kraja postojećeg sadržaja ako ga ima; ako datoteka ne postoji, sistem pokušava da je napravi)
- b (Binary - koristi se u kombinaciji sa nekim od ostalih režima)
- t (Text - koristi se u kombinaciji sa nekim od ostalih režima)

# Upisivanje u datoteku



- Koristimo sledeće 2 funkcije:

`fwrite()` = file write ili `fputs()` = file put string

- Poziv funkcije `fwrite`: `fwrite ($fp, $tekst);`

upisuje znakovnu vrednost iz promenljive `$tekst` u datoteku, na koju upućuje promenljiva `$fp`.

- Treći parametar *int duzina* nije obavezan, a predstavlja maksimalan broj bajtova koje treba upisati.
- Kada pravite datoteku sa podacima, sami određujete format u kojem će podaci biti snimljeni.

# Zatvaranje datoteke



- Funkcija `fclose`: `fclose($fp) ;`
  - Vraća *true*, ako je datoteka uspešno zatvorena, odnosno *false* ako nije.

# Ostale korisne funkcije (1)

- feof() - da li se došlo do kraja datoteke

```
while (!feof($fp))
```

- Čitanje red po red:

- `fgets($fp, 999)` - čitanje teksta red po red, dok ne naiđe na kraj reda, datoteke ili dok ne pročita 998 bajtova

- `fgetss($fp, int duz, string [dozvoljene_oznake])`

- `fgetc($fp, int duz [,string granicnik])`

- Čitanje cele datoteke:

- `readfile("$DOCUMENT_ROOT/ ../prodaja/racun.txt")`  
otvara datoteku i prosleđuje njen sadržaj  
direktno na standardni izlaz

# Ostale korisne funkcije (2)

- `$filearray=file("$DOCUMENT_ROOT/../prodaja/racun.txt")` - učitava datoteku u niz i svaki red datoteke smešta u zaseban element niza
- `fpassthru($fp);` - vraća true ako uspešno prođe fopen
- Datoteka može da se čita i znak po znak:  
`$char = fgetc($fp) //fp je pokazivač na datoteku`
- Čitanje niza proizvoljne dužine `fread($fp, int duzina)` učitava iz datoteke zadati broj bajtova ili koliko ih je preostalo do kraja datoteke
- Provera da li datoteka postoji:  
`file_exists("$DOCUMENT_ROOT/../prodaja/racun.txt");`



# Ostale korisne funkcije (3)



## ● Utvrđivanje veličine datoteke:

```
filesize("$DOCUMENT_ROOT/../../prodaja/racun.txt");
```

## ● Brisanje datoteke:

```
unlink("$DOCUMENT_ROOT/../../prodaja/racun.txt");
```

vraća false ako ne uspe da izbriše datoteku

## ● Kretanje po datoteci:

- `rewind()` - pomera pokazivač na početak datoteke
- `ftell()` - daje tekući položaj pokazivača u datoteci
- `fseek()` - pomera pokazivač za zadati broj bajtova u odnosu na referentni položaj

# Zaključavanje datoteka



- `bool flock (resurs fp, int operacija);`
- **Vrednosti operacije f-je `flock()`**
  - `LOCK_SH` (zaključavanje za čitanje)
  - `LOCK_EX` (zaključavanje za upisivanje)
  - `LOCK_UN` (otključava zaključanu datoteku)
  - `LOCK_NB` (sprečava blokiranje prilikom zaključavanja)

# 3) Upotreba nizova

- Nizovi sa numeričkim indeksima
- Nizovi sa drugačijim indeksima
- Operatori za rad sa nizovima
- Višedimenzionalni nizovi
- Sortiranje nizova
- Ostale radnje sa nizovima

# Šta je niz?



- *Niz (engl. array)* je promenljiva određenog imena koja sadrži skup vrednosti u nekom redosledu
- Vrednosti smeštene u nizu nazivaju se *elementi*
- Jedan niz može imati više elemenata, a svaki element po jednu vrednost (tekst, broj, drugi niz)

Lista u primeru apoteke >

- Svakom elementu niza pridružuje se *indeks (ključ)*
- Višedimenzionalni niz je niz nizova
- PHP podržava numerički indeksirane i asocijativne nizove

Aspirin

Brufen

Vitamin C

# Nizovi sa numeričkim indeksima

- PHP: indeksi uvek počinju sa nulom
- Inicijalizovanje: niz se pravi jezičkom konstrukcijom `array`
- Kopiranje jednog niza u drugi pomoću operatora `=`
- Funkcija `range()` automatski pravi niz rastućih brojeva

```
$lekovi=array('Andol', 'Brufen', 'VitaminC');
```

```
$brojevi = range(1,10);
```

```
$neparni = range(1,10,2);
```

//2 je korak za koji se povećava

```
$slova = range ('a', 'z');
```

# Pristupanje sadržaju niza (1)

- Sadržaju niza se pristupa pomoću imena promenljive i indeksa (ključa), koji se navodi u uglastim zagradama
- Primer:  
`$lekovi[0], $lekovi[1], $lekovi[2]`
- Isti sistem numerisanja kao u programskim jezicima C, C++, Java,... (indeks prvog u nizu je nula!)
- Sadržaj elementa niza menja se pomoću operatora =  
`$lekovi[0]='Ampicilin';`  
Može se dodati i novi element `$lekovi[3]='Sirup';`

# Pristupanje sadržaju niza (2)

- Nizovi se automatski prave prvi put kada ih upotrebljavate
- Veličina se dinamički povećava kad god mu dodate nov element
- Prikazivanje sadržaja niza:

```
for ($i = 0; $i<3; $i++)
 echo "$lekovi[$i] ";
```

- Za nizove se koristi i petlja `foreach`:

```
foreach ($lekovi as $tekuci_elem)
 echo $tekuci_elem.' ';
```

# Nizovi s drugačijim indeksima

- PHP podržava i nizove u kojima svakoj vrednosti možete da pridružite indeks koji želite
- Inicijalizovanje niza:

```
$lekovi=array ('aspirin'=>90, 'brufen'=>100,
'vitaminc'=>25);
```

Imena artikla su ključevi, a cene su vrednosti.

- Pristupanje elementima: `$lekovi['aspirin']`
- Može da se pravi i niz element po element:

```
$lekovi=array ('aspirin'=>90); //pravi se niz
$lekovi['brufen']=100;
$lekovi['vitaminc']=25;
```



# Funkcija each()



- Kada indeksi niza nisu brojevi, umesto for petlje koristi se petlja foreach i funkcije `list()` i `each()`
- Funkcija `each()` čita tekući element niza i pomera interni pokazivač na naredni element
- Funkcija `each()` redom vraća svaki element u nizu i zaustavlja se kada dođe do kraja niza
- **Primer:**

```
while ($element = each ($lekovi)) {
 echo $element ['key'];
 echo ' - ' ;
 echo $element ['value'];
 echo '
' ; }
}
```

# Funkcija list()




- Funkcija `list()` se upotrebljava da biste niz razdvojili na pojedinačne vrednosti
- `list($artikal, $cena) = each($stanje);`
- Funkcija `each()` iz niza `$stanje` izdvaja tekući element i interni pokazivač pomera na naredni element
- Funkcija `list()` pretvara elemente 0 i 1 iz niza koji je vratila funkcija `each()` u 2 nove promenljive - `$artikal` i `$cena`
- Ceo niz `$stanje` se može obraditi u petlji:

```
reset($stanje);
while (list($artikal, $cena) = each($stanje))
 echo "$artikal - $cena
";
```

# Operatori za rad sa nizovima

Operator	Ime	Primer
+	Unija	$\$a + \$b$
==	Jednako	$\$a == \$b$
===	Identično	$\$a === \$b$
!=	Različito	$\$a != \$b$
<>	Različito	$\$a <> \$b$
!==	Nije identično	$\$a !== \$b$



Operator unija pokušava da doda elemente niza  $\$b$  na kraj niza  $\$a$ . Ako u nizu  $\$b$  postoje elementi koji imaju iste ključeve kao neki elementi iz niza  $\$a$ , ti ključevi se izostavljaju iz novog niza.

# Višedimenzionalni nizovi



- Svaki element niza može da bude neki drugi niz
- Dvodimenzionalni nizovi = matrice

```
$matrica = array (array ('sifra1', 'aspirin', 90),
 array ('sifra2', 'brufen', 100),
 array ('sifra3', 'vitaminc', 25)
);
```

# Sortiranje nizova - funkcija `sort()`

- Funkcija `sort()` daje niz sortiran po abecednom redosledu

- Primer:

```
$lekovi = array ('brufen', 'sirup', 'aspirin');
sort($lekovi);
```

- Sortiranje može da se vrši i po numeričkom redosledu

- Drugi parametar funkcije `sort` je opcioni:

`SORT_REGULAR` (default), `SORT_NUMERIC`, `SORT_STRING`

# Sortiranje nizova



- Funkcija **asort()** sortira niz prema vrednosti svakog elementa
- Funkcija **ksort()** sortira niz po ključu

- Primer:

```
$lekovi=array ('aspirin'=>90, 'brufen'=>100,
'vitaminC'=>25);
```

```
asort($lekovi); //po cenama: 25,90,100
```

```
ksort($lekovi); //po kljucu tj abecedno
```

- Funkcije koje sortiraju niz po opadajućem redosledu:

```
rsort(), arsort(), krsort()
```

# Promena redosleda elemenata niza

- Funkcija **shuffle()** nasumično menja redosled elemenata u nizu
- Funkcija **array\_reverse()** pravi kopiju niza sa elementima u obrnutom redosledu
- Funkcija **array\_push()** dodaje po jedan novi element na kraj niza
- Funkcija **array\_pop()** uklanja poslednji element niza i vraća ga pozivajućem kodu

# Pokazni primeri

```
$brojevi = array ();
for ($i=10; $i>0; $i--)
 array_push ($brojevi, $i);
```

```
$brojevi = range(1,10);
$brojevi = array_reverse ($brojevi);
```



# Navigacija unutar nizova



- Kada napravimo novi niz, pokazivač upućuje na prvi element niza
- Rezultat `current ($niz)` je prvi element
- Funkcije `each` i `next` pomeraju pokazivač
- Rezultat `each ($niz)` je tekući element pre nego što se pokazivač pomeri
- Funkcija `next ($niz)` prvo pomera pokazivač unapred, a zatim vraća nov tekući element
- Funkcija `reset ($niz)` vraća pokazivač na prvi element u nizu
- Funkcija `end ($niz)` pomera pokazivač na kraj niza
- Funkcija `prev ($niz)` pomera pokazivač unatrag za jedno mesto, a zatim vraća nov tekući element

# Prebrojavanje elemenata u nizu

- Funkcija `count()` vraća ukupan broj elemenata u nizu, koji joj je prosleđen
- Funkcija `sizeof()` vraća ukupan broj elemenata u nizu, koji joj je prosleđen
- Funkcija `array_count_values($niz)` broji koliko jedinstvenih vrednosti ima u nizu; funkcija vraća asocijativni niz, koji sadrži tabelu učestanosti (ključ je element niza, vrednost je broj ponavljanja)

# 4) Rad sa znakovnim nizovima

- Formatiranje znakovnih vrednosti (stringa)
- Spajanje i razdvajanje znakovnih vrednosti
- Poređenje znakovnih vrednosti
- Regularni izrazi

# Obavezna polja u formi



- Ukoliko postoje obavezna polja u formi, preporučljivo je da se to proveri pomoću funkcije `isset()`

# Skraćivanje znakovnih vrednosti

- Funkcija `trim()` briše beline sa početka i kraja znakovnog podatka. Po definiciji briše znakove za povratak na početak reda (`\r`) i za prelazak u novi red (`\n`), horizontalne i vertikalne tabulatore (`\t`, `\x0B`), znakove za kraj znakovne vrednosti (`\0`) i razmake.
- Ovoj funkciji možete da prosledite i parametar koji sadrži eksplicitno zadat spisak znakova koje treba ukloniti umesto ovih navedenih.

# Formatiranje znakovnih vrednosti u oblik pogodan za prikazivanje

- Funkcija `n12br()` prihvata znakovnu vrednost kao ulazni parametar i sve znake za prelazak u novi red u njoj zamenjuje XHTML oznakama `<br/>`, ili HTML oznakom `<br>` pre PHP4.
- Ovo funkcija je korisna za formatiranje teksta.
- Funkcija `printf()` prosleđuje formatiranu znakovnu vrednost pregledaču veba, a funkcija `sprintf()` vraća formatiranu znakovnu vrednost.

```
printf ("Ukupno prodato: %.2f dinara", $ukupno);
// 2f oznacava broj sa dve decimale nakon zareza
```

# Više specifikacija konverzije

- U istom šablonu formata možete zadati više specifikacija konverzije. Svaka specifikacija biće zamenjena formatiranom vrednošću parametra, redosledom kojim su parametri navedeni.

```
printf ("Ukupan racun: %.3f dinara (sa porezom %.2f) ", $ukupno, $ukupno_pdv);
```

- Od verzije 4.0.6 moguće je numerisanje argumenata (tj. argumenti ne moraju da budu u istom redosledu kao specifikacije konverzije).

```
printf ("Ukupan racun: %2\$.3f dinara (sa porezom %1\$.2f) ", $ukupno_pdv, $ukupno);
```

# Tipovi specifikacija konverzije

Tip	Značenje
b	Vrednost se tumači kao ceo broj, a prikazuje kao binarni broj
c	Vrednost se tumači kao ceo broj, a prikazuje kao znakovni podatak
d	Vrednost se tumači kao ceo broj, a prikazuje kao decimalni broj
f	Vrednost se tumači kao broj sa pokretnim zarezom dvostruke preciznosti
o	Vrednost se tumači kao ceo broj, a prikazuje kao oktalni broj
s	Vrednost se tumači kao znakovna i prikazuje kao znakovna
u	Vrednost se tumači kao ceo broj, a prikazuje kao decimalni broj bez predznaka
x	Vrednost se tumači kao ceo broj, a prikazuje kao heksadecimalni broj sa malim slovima za cifre (a-f)
X	Vrednost se tumači kao ceo broj, a prikazuje kao heksadecimalni broj sa malim slovima za cifre (A-F)



# Pretvaranje malih i velikih slova

- Funkcije za menjanje malih i velikih slova u znakovnim podacima

Upotreba	Opis	Vrednost
\$ime		Idete li na Eurosong?
strtoupper(\$ime)	sva slova menja u velika	IDETE LI NA EUROSONG?
strtolower(\$ime)	sva slova menja u mala	idete li na eurosong?
ucfirst(\$ime)	menja u veliko slovo prvi znak ulaznog argumenta, ako je alfabetski	Idete li na Eurosong?
ucwords(\$ime)	menja u veliko slovo prvi znak svake reči koja počinje alfabetskim znakom	Idete Li Na Eurosong?

# Formatiranje znakovnih vrednosti u oblik pogodan za unošenje u bazu

- Određeni znakovi, koji su potpuno prihvatljivi u znakovnim vrednostima, mogu da izazovu probleme prilikom unošenja u bazu podataka, zato što baza može da ih protumači kao upravljačke znakove.
- Znakovi koji prave probleme:
  - navodnici (obični i polunavodnici)
  - obrnuta kosa crta ( \ )
  - znak NULL

# Funkcija `addslashes($string)`

- `$tekst = addslashes($tekst);`
- Pomoću ove funkcije treba formirati znakovni podatak, pre upisivanja u bazu.
- Argument ove funkcije je znakovna vrednost, a rezultat je formatirana znakovna vrednost.
- **Primer:**

```
$tekst = "Shaquille O'Neil";
```

```
$tekst = addslashes($tekst);
```

```
//Rezultat: Shaquille O\'Neil
```

# Funkcija `stripslashes($string)`

- Ova funkcija uklanja kose crte, koje je postavila funkcija `addslashes($string)` ;

# Magični navodnici



- Novije verzije PHP-a automatski uključuju direktivu `magic_quotes_gpc`.  
To znači da se automatski konvertuju sve znakovne promenljive iz GET, POST i COOKIE.
- Provera da li je direktiva uključena pomoću:  
`get_magic_quotes_gpc()`  
pa ako vraća rezultat `true`, to znači da se znakovne promenljive iz GPC konvertuju automatski, pa ćete za prikazivanje podataka korisniku morati da obradite podatke pomoću funkcije `stripslashes($string)`.

# Spajanje i razdvajanje znakovnih vr.

- `explode(string granicnik, string text)`
- Funkcija deli znakovnu vrednost `text` na mestima gde naiđe na znakovnu vrednost `granicnik`.
- Funkcija vraća niz.
- **Primer:**

```
$email_array = explode ('@', $email);
$email_array[0] = draskovic
$email_array[1] = etf.bg.ac.rs
```
- Suprotne ovoj funkciji su `implode()` i `join()`

# Funkcija substr()



- Ova funkcija omogućava izdvajanje dela ulazne znakovne vrednosti koja se nalazi između zadatog početnog i krajnjeg položaja.
- `string substr (string tekst, int pocetak [, duzina]);`
- **Primer:**  

```
$test = "Danas je Dan republike!";
echo substr($test, 5); //je Dan republike!
echo substr($test, 0, 5); //Danas
```

# Poređenje znakovnih vrednosti

- `int strcmp(string str1, string str2)`

Funkcija poredi dve znakovne vrednosti `str1` i `str2` i ako su jednake funkcija vraća 0. Ukoliko po abecednom redosledu vrednost `str1` dolazi iza `str2`, funkcija vraća broj veći od nule, u suprotnom vraća broj manji od nule.

- `int strcasecmp(string str1, string str2)`

Ista kao prethodna, samo ne pravi razliku između malih i velikih slova.



# Dužina znakovne vrednosti

- Dužina znakovne vrednosti (ukupan broj znakova) možete utvrditi pomoću funkcije `strlen()`  
`strlen('Hello')` je 5

# Nalaženje stringova unutar drugih stringova

- Funkcija `strstr()` omogućava traženje date znakovne vrednosti ili znaka unutar druge znakovne vrednosti.
- Identična ovoj funkciji je i funkcija `strchr()`
- Prototip funkcije `strstr()`:  
`string strstr (string tekst, string uzorak);`
- Funkcija `stristr()` je gotovo identična izvornoj funkciji, s tim što ne pravi razliku između malih i velikih slova u tekstu koji traži.

# Utvrđivanje položaja zadatog znakovnog podniza

- Funkcije `strpos()` i `strrpos()` su slične funkciji `strstr()`, ali ne vraćaju izdvojeni deo ulaznog paramtera, nego položaj parametra uzorak u parametru tekst.
- Prototip funkcije `strpos()`:  

```
int strpos(string tekst, string uzorak, int [pomak]);
//pomak je mesto odakle pocinje pretraga u tekstu
```
- Primer:  

```
$test = "Dobrodosli u Srbiju!";
echo strpos($test, 'o'); //rezultat: 1
echo strpos($test, 'do', 3); //rezultat: 5
```
- Funkcija `strpos()` radi brže od funkcije `strstr()`.

# Zamenjivanje delova znakovnog podatka (1)

- Funkcije `str_replace()` i `substr_replace()`
- Funkcionalnost je dobra za cenzurisanje pojedinih izraza, na primer na forumima.
- Prototip funkcije:

```
mixed str_replace (mixed uzorak, mixed zamena,
mixed tekst[, int &koliko_puta]);
```
- Rezultat funkcije je nova verzija prosleđenog teksta `tekst`, u kome je `uzorak` zamenjen vrednošću drugog parametra `zamena`.
- Opcioni parametar `koliko_puta`, kaže koliko puta treba da se izvrši zamena (tek od PHP 5).

# Zamenjivanje delova znakovnog podatka (2)

- Funkcija `substr_replace()` pronalazi i zamenjuje zadati podniz u zavisnosti od njegovog položaja.
- Prototip funkcije:  

```
string substr_replace (string tekst, string
zamena, int pocetak, int [duzina]);
```
- Ova funkcija zamenjuje deo parametra `tekst`, vrednošću drugog parametra `zamena`.
- Deo teksta u kome će biti izvršena zamena određen je trećim i opciono četvrtim parametrom.

# Primer 3

Napraviti formular za registrovanje korisnika na studentskom forumu koji će sadržati polja za ime, telefon, e-mail adresu i potvrdu e-mail adrese. Kada korisnik unese mail koji nije u formatu: *ime@domen* treba da se ispiše poruka da je pogrešan unos maila. Ako korisnik ne unese potvrdu maila treba da se ispiše da je pogrešno potvrđen mail. Kada korisnik potvrdi mail, ispisuje se poruka da će šifra korisniku *ime* na mail *ime@domen*.

## Otvaranje naloga za studentski forum

Ime i prezime	<input type="text" value="Drazen Draskovic"/>
Mobilni	<input type="text" value="069 234567"/>
Unesite e-mail adresu	<input type="text" value="drazen@ef.bg.ac.yu"/>
Potvrdite vasu e-mail adresu	<input type="text" value="drazen@"/>
<input type="button" value="Registruj me"/>	

# Rešenje (1)



```
<?php
 // pravljenje kratkih imena
 $ime = $_POST['ime'];
 $mob = $_POST['mob'];
 $email1 = $_POST['email1'];
 $email2 = $_POST['email2'];

 $DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
 //ovo mora da postoji

 $uporedi = strcmp($email1, $email2);
 $email_niz = explode ('@', $email1);
 $brojac = count($email_niz);
```

# Rešenje (2)



```
if (strlen($email_niz[1])==0)
 echo 'Niste lepo uneli e-mail adresu!!!
 Pokusajte ponovo.';
 else if ($uporedi!=0)
 echo 'Niste potvrdili mail adresu,
 pokusajte ponovo registraciju.';
 else
 echo 'Uspesna registracija naloga:
 '.$email_niz[0].'
 Sifra za forum
 ce uskoro stici na vas mail: '.$email2;
?>
```



# Primer 4

- Napravite formu sa checkbox-ovima gde korisnik može da između 4 muzičke grupe ili žanra odabere one koje sluša. Ono što sluša treba beležiti u niz, nepoznate veličine, a kada korisnik odabere i pritisne dugme potvrde potrebno je da se prikaže njegov izbor tj. da se isčita iz niza. U slučaju da korisnik ništa ne odabere, treba da se ispiše da ništa nije izabrano.

## **Izaberite muziku koju slusate**

Bon Jovi  Police  Madonna  RHCP

# Rešenje (1)



```
<html>
<body>
<h1>Izaberite muziku koju slusate</h1>
<?php
 if (!isset($_POST['submit'])) {
?>
 <form action="<?php echo $_SERVER['PHP_SELF'];
?>"method="POST">
 <input type="checkbox" name="muzika[]" value="Bon Jovi">Bon
 Jovi
 <input type="checkbox" name="muzika[]" value="Police">Police
 <input type="checkbox" name="muzika[]"
 value="Madonna">Madonna
 <input type="checkbox" name="muzika[]" value="RHCP">RHCP
 <input type="submit" name="submit" value="Izaberi">
 </form>
```

# Rešenje (2)



```
<?php
 }
 else {
 if (is_array($_POST['muzika'])) {
 echo 'Izabrali ste:
';
 foreach ($_POST['muzika'] as $nesto)
 {
 echo "<i>$nesto</i>
";
 }
 } else {
 echo 'Nista nije izabrano';
 }
 }
?>
</body> </html>
```

# 5) Višekratna upotreba koda

- Iskazi `require()` i `include()`
- Definisiranje sopstvenih funkcija
- Pozivanje po referenci i vrednosti
- Rekurzija

Jedan od ciljeva dobrih programera:

Koristiti što više napisan kod,  
umesto da pišemo novi kod.

# Upotreba require() i include()

- Pomoću iskaza require(), odnosno include() u PHP skript se može učitati datoteka. Ta datoteka može da sadrži iskaze PHP-a, tekst, HTML oznake, PHP-ove funkcije ili klase.

# Primer



Datoteka dodatak.php ima sledeći sadržaj:

```
<?php
 echo 'Ovo ce biti ubaceno.
';
?>
```

Datoteka main.php ima sledeći sadržaj:

```
<?php
 echo 'Ovo je glavni fajl.
';
 require('dodatak.php');
 echo 'Ovde je kraj!
'
?>
```

# Šta je rezultat?



- Izraz `require()` umeće u skript sadržaj datoteke `dodatak.php` koju smo zadali.
- Kada pokrenemo skript, prvo se iskaz 

```
require('dodatak.php');
```

 zamenjuje sadržajem navedene datoteke, pa se tek onda skript izvrši.

# Primer šablona za veb strane



Korisno je da ovu veb stranicu podelimo na delove:

- deo pre sadržaja - zaglavlje i meni (header.inc)
- dinamički deo sa sadržajem (home.php)
- deo iza sadržaja (footer.inc)

Datoteke header.inc i footer.inc sadrže kod koji ćemo upotrebljavati i na drugim stranicama.



# Primer šablona za veb strane

```
//home.php
<?php
 require('header.inc');
?>
 <p>Dobrodosli! Najjeftinije telefoniranje u Srbiji!!!</p>
<?php
 require('footer.inc');
?>
```

- Savet je da nastavak imena datoteka čiji se sadržaj umeće u druge datoteke bude `.inc` (skraćeno od include). Takođe se preporučuje da se datoteke za umetanje postavbe izvan stabla veb dokumenata kako bi se sprečilo neovlašćeno čitanje vašeg izvornog koda.

# Funkcije



- Funkcija je samostalan modul koji propisuje način pozivanja funkcije, obavlja zadatak i eventualno vraća rezultat.
- Pozivanje funkcija `moja_funkcija()` ;
- Pozivanje nedefiniranih funkcija => greška!
- Nema razlike između malih i velikih slova kod funkcija u PHP (Funk, FUNK, funk... je ista funkcija)

# Osnovna struktura funkcije

- Deklaracija počinje sa `function`, iza koje slede ime funkcije i parametri, a zatim i kod koji se izvršava kada je funkcija pozvana
- Imena funkcija:
  - Nova funkcija ne sme da ima isto ime kao neka postojeća
  - Ime funkcije može da bude sastavljeno od slova, cifara i donje crte `_`
  - Ime funkcije ne može počinjati cifrom
  - PHP ne dozvoljava preklapanje ugrađenih funkcija!!
- Može da postoji jedan ili više parametara, ali ne moraju svi biti obavezni.

# Primer - tabela



```
function napravi_tabelu($nesto)
{
 echo '<table border = 1>';
 reset($nesto); // Pokazivac na pocetak niza
 $vrednost = current($nesto);
 while ($vrednost) {
 echo "<tr><td>$vrednost</td></tr>\n";
 $vrednost = next($nesto);
 }
 echo '</table>';
}
$moj_niz = array('Prvo polje.', 'Drugo polje.', 'Trece
 polje.');
```

```
napravi_tabelu($moj_niz);
```

# Opseg važenja



- Opseg važenja promenljive određuje gde je promenljiva vidljiva i upotrebljiva.
- Promenljive deklarisanе unutar funkcije čiji je opseg od mesta deklarisanja do kraja funkcije (lokalne promenljive - funkcijski opseg važenja).
- Promenljive deklarisanе izvan funkcije čiji je opseg od mesta deklarisanja do kraja datoteke (globalne promenljive - globalni opseg važenja).
- Pomoću reči `global` može se ručno zadati globalni opseg važenja za promenljivu definisanu unutar funkcije.

# Prenos parametara po vrednosti

- Engl. *pass by value*
- Kada funkciji prosledite vrednost parametra, unutar funkcije se pravi nova promenljiva koja sadrži prosleđenu vrednost - kopija originala.
- Može slobodno da se menja, dok izvorna promenljiva ostaje nepromenjena.

# Prenos parametara po referenci

- Engl. *pass by reference*
- Kada se parametar prosledi funkciji, ona ne pravi novu promenljivu već preuzima referencu na originalnu promenljivu.
- Referenca se ponaša kao promenljiva, upućuje na original i nema sopstvenu vrednost (svaka izmena reference se odnosi i na original).

```
function inkrement(&$vr, $ink=1) {
 $vr = $vr + $ink;
}
```

```
$a = 10;
echo $a.'
>';
inkrement($a);
echo $a. '
>';
```

# Povratak iz funkcije

- Izvršenje funkcije se prekida sa `return`

```
function veci($x, $y) {
 if(!isset($x)||!isset($y)) {
 echo 'Prosledite dva broja';
 return;
 }
 if($x>=$y)
 echo $x;
 else
 echo $y;
}
```

**isset()** utvrđuje da li je promenljiva napravljena i da li ima vrednost

```
$a = 1;
$b = 2.5;
$c = 1.9;
veci($a, $b);
veci($c, $a);
veci($d,$a);
```



# Rekurzija



- Rekurzivne funkcije - one koje pozivaju samu sebe
- Prednosti: kraći kod, elegantnije rešenje
- Nedostaci: opterećenje memorije

```
function obrnuto_r($str)
{
 if (strlen($str)>0)
 obrnuto_r(substr($str, 1));
 echo substr($str, 0, 1);
 return;
}
```

```
obrnuto_r('Zdravo');
obrnuto_r('dravo');
obrnuto_r('ravo');
obrnuto_r('avo');
obrnuto_r('vo');
obrnuto_r('o');
obrnuto_r('');
```

# 6) Objektno orijentisano programiranje na PHP-u

- Izrada klasa, objekata, atributa
- Polimorfizam, nasleđivanje, redefinisavanje
- Modifikatori pristupa
- Interfejsi

# Klase i objekti



- Klasa je osnovna jedinica programiranja na OO jezicima
- Klase sadrže:
  - Attribute (svojstva ili promenljive koje opisuju objekat)
  - Operacije (metode, radnje ili funkcije koje objekat može da izvršava)
  - Mehanizme za stvaranje objekata na osnovu definicije (konstruktoze)
- Enkapsuliranje (skrivanje podataka)
  - Pristup podacima unutar datog objekta moguć samo pomoću operacija tog objekta
- Šta je objekat?
  - Jedan primerak (instanca, pojava) klase

# Polimorfizam, nasleđivanje

- OO jezici podržavaju polimorfizam: svaki objekat izvedene klase izvršava operaciju onako kako je to definisano u njegovoj (izvedenoj) klasi
- Nasleđivanje: klasa + jedna ili više potklasa
- Potklasa (izvedena klasa, dete) nasleđuje attribute i operacije od svoje natklase (roditeljske klase)
- Nasleđivanje omogućava nadgradnju i proširenje postojećih klasa:

```
class Vozilo
class Auto extends Vozilo
class Kamion extends Vozilo
```

# Struktura klase, konstruktori

```
class ime {
 var $atribut1;
 var $atribut2;
 function operacija1() {}
 function operacija2($par1, $par2) {}
}
```

- Konstruktor se poziva prilikom pravljenja objekata date klase
- Konstruktor se deklarira kao druge operacije, ali ima specijalno ime `construct()`

```
class ime {
 function __construct($par) {
 echo "pozvan je konstruktor sa parametrom $par";
 }
}
```

# Pravljenje objekata



- Nov objekat se pravi pomoću rezervisane reči `new`

- Primer:

```
$a = new ime('Prvi');
```

```
$b = new ime('Drugi');
```

```
$c = new ime();
```

- Rezultat:

pozvan je konstruktor s parametrom Prvi

pozvan je konstruktor s parametrom Drugi

pozvan je konstruktor s parametrom

# Upotreba atributa klase



- Pokazivač `$this` upućuje na tekući objekat
- Ako tekući objekat ima atribut `$atr`, možete da mu pristupite pomoću imena `$this->atr`
- Primer:

```
class ime {
 var $atr;
 function operacija($par) {
 $this->atr = $par;
 echo $this->atr;
 }
}
```

# Pristupne funkcije



## ● Primer:

```
class ime {
 var $atr;
 function __get($imeatributa) {
 return $this->$imeatributa; }
 function __set($imeatributa, $nova_vred) {
 $this->$imeatributa=$nova_vred; }
}
```

## ● Ove 2 funkcije se pozivaju implicitno

```
$a = new ime();
$a->atr = 5; // poziva se funkcija __set
```



# Modifikatori pristupa



- public (javni) - elementima koji se deklarišu kao javni može se pristupati i unutar i izvan klase;
- private (privatni) - elementima koji se deklarišu kao privatni može se pristupati samo unutar klase; ne mogu da se nasleđuju;
- protected (zaštićen) - označava da elementu klase može da se pristupi samo unutar klase, ali se taj element nasleđuje u svim potklasama;

```
class ime {
 public $atribut;
}
```

# Pozivanje operacije klase



```
$a = new ime();
```

```
//pristup kao atributu tog objekta
```

```
$a->operacija1();
```

```
$a->operacija2(11, 'proba');
```

```
//ako operacija vraca neku vrednost
```

```
$x = $a->operacija1();
```

```
$y = $a->operacija2(11, 'proba');
```

# Nasleđivanje - primer



```
class B extends A {
 var $atribut2;
 function operacija2() {}
}
class A {
 var $atribut1;
 function operacija1() {}
}
```

Klasa A nema funkciju `operacija2()`  
i atribut `$atribut2` !!!

# Redefinisanje (eng. *overriding*)

- Promena postojeće funkcionalnosti natklase u nasleđenoj klasi

```
class A {
 var $atribut = "staravrednost";
 function operacija() {
 echo 'nesto
';
 echo "vrednost je $this->atribut"; }
}

class B extends A {
 var $atribut = "novavrednost";
 function operacija() {
 echo 'nesto drugo
';
 echo "vrednost je $this->atribut"; }
}
```

# Redefinisanje - Primer



```
$a = new A();
$a->operacija();
```

Izlaz:

```
nesto
vrednost je staravrednost
```

```
$b = new B();
$b->operacija();
```

Izlaz:

```
nesto drugo
vrednost je novavrednost
```

Poziv u potklasi B:

```
parent::operacija();
```

```
//nesto
```

```
//vrednost je novavrednost
```



# final



- Služi za sprečavanje nasleđivanja
- Kada se postavi ispred deklaracije funkcije, ta se funkcija ne može zameniti istoimenom ni u jednoj potklasi:

```
final function operacija() { ... }
```

- Upotrebom `final` se može sprečiti i nasleđivanje određene klase:

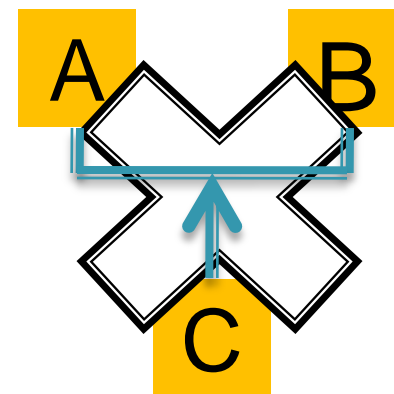
```
final class A()
{ ... }
```

# Interfejsi

- PHP ne podržava višestruko nasleđivanje.
- Interfejs deklariše određen broj funkcija koje moraju biti realizovane u svim klasama koje realizuje taj interfejs.

```
interface Prikazuje {
 function prikazi ();
}
```

- Interfejs se koristi kao “zaobilazni” način da se realizuje višestruko nasleđivanje



# Naprednije OO funkcionalnosti

- **const** - konstanta klase, koja se koristi bez obaveze da se napravi objekat:

```
class Matematika {
 const pi = 3.14159;
}
echo 'Pi je 'Matematika::pi;
```

- **static** - kada je zadata ispred deklaracije metode, omogućava da se poziva metoda bez pravljenja instance klase:

```
static function kvadrat($broj) {
 return $broj*$broj; }
echo 'Kvadrat od 8 = 'Matematika::kvadrat(8);
```



# Primer sa statičkom funkcijom (1)

```
<?php //Student.php
class Student {
 public $firstName;
 public $lastName;
 public function __construct($firstName, $lastName = '') {
 //Optional parameter
 $this->firstName = $firstName;
 $this->lastName = $lastName;
 }
 public function greet() {
 return "Hello, my name is " . $this->firstName . " " .
 $this->lastName . ".";
 }
 public static function staticGreet($firstName, $lastName) {
 return "Hello, my name is " . $firstName . " " .
 $lastName . ".";
 }
}
?>
```

# Primer sa statičkom funkcijom (2)

```
<?php //index.php
 require ("Student.php");
 $demon1 = new Student('Stefan', 'Kostic');
 $demon2 = new Student('Mina', 'Reljic');
 $demon3 = new Student('Filip Zivkovic');

 echo $demon1->greet();
 echo '
';
 echo $demon2->greet();
 echo '
';
 echo $demon3->greet();
 echo '
';
 echo Student::staticGreet('Balsa', 'Bojic');
?>
```

# 7) Obrada izuzetaka na PHP-u

- Koncepti obrade izuzetaka
- Klasa Exception
- Izuzeci koje korisnik definiše

# Obrada izuzetaka



- Kod se izvršava unutar `try` bloka:

```
try {
 //kod
}
```

- Unutar `try` bloka izuzetak se izaziva:

```
throw new Exception('poruka', broj_greske);
```

- Svakom bloku `try` mora da sledi barem jedan blok `catch`:

```
catch (Exception $e) {
 //obrada izuzetka
}
```

- Objekat koji se prosleđuje bloku `catch` (da bi ga blok presreo) jeste objekat prosleđen iskazu `throw` koji je generisao izuzetak.

# Obrada izuzetaka - Primer



```
<?php
 try
 {
 throw new Exception('Greska!!!', 42);
 }
 catch (Exception $e)
 {
 echo 'Exception '. $e->getCode(). ': '.
 $e->getMessage(). '
'. ' u fajlu '. $e->getFile().
 'na liniji '. $e->getLine(). '
';
 }
?>
```

# Klasa Exception



- Konstruktor te klase prihvata 2 parametra: tekst poruke o grešci i broj greške
  - getCode() - vraća broj greške koji je bio prosleđen konstruktoru
  - getMessage() - vraća tekst poruke koja je bila prosleđena konstruktoru
  - getFile() - pozivajućem kodu vraća punu putanju datoteke u kojoj je generisan izuzetak
  - getLine() - vraća broj reda koda u kome je generisan izuzetak
  - getTrace() - vraća niz s podacima o stablu pozivanja koji omogućavaju utvrđivanje mesta na kome je generisan izuzetak
  - getTraceAsString () - vraća iste podatke kao getTrace(), formatirane kao znakovni podaci
  - \_\_toString() - omogućava da se iskazu echo direktno prosledi ceo sadržaj objekta Exception, sa svim podacima koje daju navedene metode

# Napredne tehnike u PHP-u

- Korisnička autentifikacija
- Kolačići i sesije
- Rad sa fajlovima i serverom
- Upotreba mrežnih funkcija i protokola
- Rad sa datumima i vremenima
- Ostale korisne mogućnosti
  - Funkcija eval()
  - Prekid izvršavanja skripta
- Zaštita aplikacije

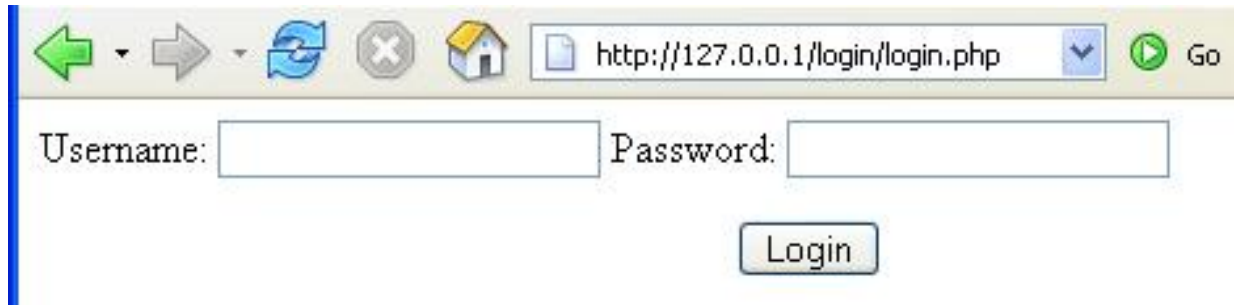


# *Korisnička autentifikacija*



# Forma za prijavljivanje korisnika

- Tekstualna polja za unos:
  - korisničkog imena
  - lozinke



A screenshot of a web browser window showing a login form. The browser's address bar displays the URL `http://127.0.0.1/login/login.php`. Below the address bar, there are two text input fields: one labeled "Username:" and one labeled "Password:". Below these fields is a "Login" button. The browser's navigation bar includes back, forward, refresh, stop, and home icons.

# login.php (1)



- Na počátku skripta stavljamo lokalne promjenljive:

```
$uname = "";
$pword = "";
$errorMessage = "";
$num_rows = 0;
```

- **\$errorMessage** ćemo koristiti, ako želimo da sačuvamo neku poruku koju ćemo vratiti login stranici.

# login.php (2)



- Sledeći kod prikazuje SQL proveru, kako bi se sprečio SQL injection napad:

```
if ($_SERVER['REQUEST_METHOD'] == 'POST')
{

}
```

- Nakon ovoga znamo da li je forma prosleđena ili nije (da li smo kliknuli na Submit)

# login.php (3)



- Prvo ćemo prihvatiti podatke koji su prosleđeni i smestiti ih u lokalne promenljive:

```
$uname = $_POST['username'];
$pwd = $_POST['password'];
```

- Nakon ovoga znamo da li je forma prosleđena ili nije (da li smo kliknuli na Submit)

- Zatim treba da uklonimo i neželjeni HTML kod (skript napad):

```
$uname = htmlspecialchars($uname);
$pwd = htmlspecialchars($pwd);
```

# login.php (4)



- Pokušamo da ostvarimo konekciju sa bazom:

```
$user_name = "root";
$pass_word = "";
$database = "login";
$server = "127.0.0.1";
```

```
$db_handle = mysql_connect($server, $user_name,
$pass_word);
$db_found = mysql_select_db($database, $db_handle);
```

- Ako je baza pronađena, varijabla nazvana \$db\_found će biti true:

```
if ($db_found) {
}
else {
 $errorMessage = "Error logging on";
}
```

# login.php (5)



- Sada treba tekst koji imamo da sredimo od neželjenih znakova:

```
$uname = quote_smart($uname, $db_handle);
$pword = quote_smart($pword, $db_handle);
```

- Funkcija `quote_smart` sprečava SQL injection :

```
function quote_smart($value)
{
 $value = stripslashes($value);
 if (!is_numeric($value))
 $value = "'" . mysql_real_escape_string($value) .
 "'";
 return $value;
}
```

# PHP - stripslashes



- string **stripslashes** ( string \$str )
- Ima string kao argument, već smo rekli da radi suprotno od addslashes()
- Vraća string vrednost, ali bez backslash-eva.  
Na primer:
  - \' postaje samo '
  - dupli backslash (\\) postaje jedan (\)

# PHP - is\_numeric



- **bool is\_numeric** ( mixed \$var )
- Argument \$var može biti različitog tipa!
- Vraća **TRUE** ako je argument *var* ili broj ili numerički string, **FALSE** u drugim slučajevima.

```
<?php
$tests = array("42", 1337, "1e4", "not numeric",
 Array(), 9.1);

foreach ($tests as $element) {
 if (is_numeric($element)) {
 echo "'{$element}' je broj!", PHP_EOL;
 } else {
 echo "'{$element}' nije broj!", PHP_EOL;
 }
}
?>
```



# PHP - mysql\_real\_escape\_string

- Izbacuje specifične znakove u stringu, kako bi se string koji vrati funkcija koristio u SQL naredbi.
- Ova funkcija mora uvek (uz nekoliko izuzetaka) da se koristi za pravljenje “bezbednih podataka” pre slanja upita MySQL bazi!

# login.php (6)



- Kada imamo korisničko ime i lozinku, možemo da izvršimo SQL upit:

```
$upit = "SELECT * FROM login WHERE kime =
$uname AND klozinka = $pword";
$rezultat = mysql_query($upit);
```

- Funkcija `mysql_query` izvršava upit na MySQL bazom. Šta vraća `mysql_query`?

# mysql\_query(\$upit)



- Vrednost u promenljivoj \$rezultat će biti ili true (ako postoji neki red u bazi koji zadovoljava upit) ili false (ako nijedan red nije vratio upit). U drugom slučaju, treba stavi poruku o grešci:

```
if ($rezultat) {

}
else {
 $errorMessage = "Nije uspesno
 logovanje";
}
```

# mysql\_num\_rows(\$ rezultat)

- Ako se SQL upit izvrši uspešno, baza može da vrati nekoliko redova.
- Tada koristimo `mysql_num_rows( )` funkciju
- Ako nema redova, znači da korisnik sa tim korisničkim imenom i tom lozinkom, ne postoji u bazi:

```
$num_rows = mysql_num_rows ($rezultat);
if ($num_rows > 0) {
 $errorMessage= "Ulogovani ste!";
}
else {
 $errorMessage= "Nije uspesno logovanje";
}
```

# Username = primarni ključ

- U primeru, ako je broj redova veći od 1, to znači da 2 korisnika imaju isto korisničko ime i lozinku. Ako imate sistem (bazu) gde svaki korisnik ima jedinstveno korisničko ime, onda mora da bude  $\$num\_rows = 1$ .



## *Kolačići i sesije*

# HTTP ne čuva stanje



- Veb pregledač i veb server međusobno komuniciraju pomoću HTTP protokola koji ne čuva stanje između dve razmene podataka.
- Svaki HTTP zahtev koji veb pregledač šalje veb serveru nezavisan je od svih drugih zahteva.
- Ovakva komunikacija veb pregledača i veb servera pogodna je za aplikacije koje korisnicima omogućavaju da pretražuju ili pregledaju grupe povezanih dokumenata koristeći hiperlinkove.

# Potreba za čuvanjem stanja

- U aplikacijama u kojima je potrebna složenija intervencija korisnika, mora se obezbediti čuvanje stanja pri prelasku sa jedne stranice aplikacije na drugu.
- *Primer:* Veb aplikacija prodavnice  
Korisnik dodaje stavke u korpu dok pretražuje ili pregleda neki katalog. Stanje korpe za kupovinu (stavke koje su sadržane u njoj) mora se negde čuvati. Kada korisnik zahteva stranicu za prikaz sadržaja korpe, mora se prikazati koje se stavke u njoj nalaze.



# Čuvanje stanja bez upotrebe sesije

- U svakom HTTP zahtevu, među podacima koji se šalju metodom GET ili POST, prosleđivati i podatke koji predstavljaju trenutno stanje aplikacije.
- Nepotrebno povećanje saobraćaja na veb-u.
- Ako se podaci koji opisuju stanje prenose HTTP GET metodom (kao deo URL-a), korisnik može ručno da izmeni vrednosti koje se šalju zahtevom, a često nastaju i dugačke i nepregledne URL adrese.

# Pravljenje aplikacija koje čuvaju stanje

- Stanje aplikacije (vrednosti pojedinih promenljivih) mora se negde skladištiti između dva HTTP zahteva.
- Promenljive koje opisuju stanje mogu se čuvati na dva mesta:
  1. u klijentskom veb pregledaču
  2. na veb serveru

# Pristup kolačićima u PHP skriptu

- Ako HTTP zahtev sadrži kolačiće, oni se smeštaju u superglobalnu promenljivu `$_COOKIE` (asocijativan niz, sa indeksom koji odgovara nazivu kolačića)
- Pristupanje kolačiću, čiji je naziv `moj_kolacic`, da bismo pročitali vrednost dobijamo pomoću `$_COOKIE["moj_kolacic"]`

# Funkcija setcookie (1)



- Funkcija setcookie() generiše ispravno polje zaglavlja i ugrađuje ga u HTTP odgovor
- Funkcija setcookie() se poziva sa 6 argumenata, iako je samo prvi (ime kolačića) obavezan

```
int setcookie(string name,
 [string value],
 [int expire],
 [string path],
 [string domain],
 [int secure])
```

# Funkcija setcookie (2)

- **name:** naziv kolačića; obavezan parametar
- **value:** vrednost kolačića koja će se čuvati na računaru klijenta; ne čuvati poverljive informacije; vrednost kolačića čiji je naziv `my_cookie` u okviru skripta se dohvata sa `$_COOKIE["my_cookie"]`
- **expire:** vreme u timestamp formatu kada kolačić prestaje da važi; najčešće korišćeni oblici **`time()+broj_sekundi`** ili **`mktime()`**

# Funkcija setcookie (3)



- **path**: putanja na serveru na kojoj će kolačić biti dostupan  
Primeri:
  - ' / ' - ceo domen
  - '/**www/admin/**' - u folderu www/admin i svim podfolderima,
  - **default path** - tekući direktorijum u kome se nalazi skript koji je kreirao kolačić

# Funkcija setcookie (4)



- **domain:** domen u kome je kolačić dostupan
- **Primeri:**
  - '' - (prazan string) domen kome pripada server na kome se nalazi skript koji kreira kolačić
  - **'www.example.com'** - u navedenom domenu,
  - **'.bg.ac.rs'** - u svim poddomenima domena Univerziteta u Beogradu bg.ac.rs,

# Funkcija setcookie (5)



- **secure :**

Označava da kolačić treba da bude poslat samo putem sigurne HTTPS konekcije.

Vrednosti TRUE ili 1 znače da se kolačić šalje samo kroz sigurne HTTPS konekcije.

Default vrednost je FALSE ili 0 što znači da se kolačić šalje po svim vezama (i sigurnim i nesigurnim).



# Funkcija setcookie - ograničenja

- Kao i ostali header-i, kolačići moraju biti poslani pre bilo koji naredbe izlaza u okviru skripta (ograničenje protokola).
- To znači da se poziv funkcije setcookie() prethodi bilo kom izlazu (npr. pozivu funkcije echo), uključujući i <html> i <head> tagove kao i bilo koje beline.

# Funkcija setcookie - povratne vred.

- Ako postoji naredba izlaza koja prethodi funkciji setcookie(), funkcija se neće uspešno završiti i vratiće FALSE.
- Ako se funkcije setcookie uspešno izvrši, vratiće vrednost TRUE, što ne znači da je korisnik prihvatio kolačić.

# Ograničenja za kolačiće (1)

- Kolačići se mogu koristiti u jednostavnim aplikacijama u kojima nije neophodno da se složeni podaci čuvaju između dva zahteva serveru.
- Broj i veličina kolačića su ograničeni: veb pregledač može da čuva samo poslednjih 20 kolačića koji su mu bili poslani iz određenog domena, a veličina svakog kolačića je ograničena na 4KB (ovo se vremenom menja i zavisi od verzije pregledača).

# Ograničenja za kolačiće (2)



- Pitanje privatnosti korisnika i zaštite aplikacije u kojima se koriste kolačići.
- Neki korisnici isključuju podršku za rad sa kolačićima.

# Upravljanje sesijama na vebu

- Podaci o tekućem stanju aplikacije čuvaju se na veb serveru tj. u srednjem sloju aplikacije
- Rešava se problem čuvanja promenljivih stanja koje zauzimaju više prostora i/ili većeg broja promenljivih stanja.
- Rešava se problem zaštite podataka sadržanih u promenljivama stanja od nenamernih ili namernih izmena koje bi korisnik mogao načiniti.

# Sesija



- Sesija (engl. *session*) je jedan od načina označavanja i upravljanja skupom podataka o stanju, pomoću sesijskih promenljivih datog korisnika.
- Kada korisnik pošalje HTTP zahtev, srednji sloj aplikacije mora da obradi tekući zahtev vodeći računa o kontekstu (stanju) sesije.

# Početak sesije



- Kada korisnik započne sesiju, klijentskom pregledaču šalje se identifikator sesije, najčešće u obliku kolačića, čija se vrednost ugrađuje u sve naredne zahteve koje veb pregledač upućuje serveru.
- Pomoću identifikatora sesije server identifikuje odgovarajuću sesiju pre nego što nastavi dalju obradu prispelog zahteva.

# Identifikator sesije



- Kod kolačića se lokalno (u klijentskom veb pregledaču) skladište vrednosti svih promenljivih koje su neophodne za održavanje stanja i one se ugrađuju u svaki HTTP zahtev.
- Ako se koriste sesije **web pregledač čuva i ugrađuje u svaki HTTP zahtev samo identifikator sesije, na osnovu koga se jednoznačno identifikuju sesijske promenljive njegove sesije.**



# Problem zamrznutih sesija (1)

- U srednjem sloju se čuvaju zasebni podaci za svaku sesiju.
- Pitanje: Koliko dugo?
- Kada se sve odvija kako bi trebalo korisnik se sam odjavljuje iz aplikacije (recimo klikom na dugme "Kraj rada"), a skript koji se onda pokreće završava sesiju.
- Međutim, u ovo se ne smete pouzdati. Korisnici se često ne odjavljuju iz aplikacije na adekvatan način.

# Problem zamrznutih sesija (2)

- Ako se korisnik ne odjavi adekvatno iz aplikacije, njegova sesija se neće završiti tj. njegove sesijske promenljive će se i dalje čuvati na serveru.
- Server nikada ne može da bude siguran da li se na drugoj strani veze još uvek nalazi korisnik (svaki HTTP zahtev je nezavisan od drugih zahteva).
- Zato server treba redovno da čisti stare, nezavršene (zamrznute) sesije u kojima se tokom određenog vremena nije ništa događalo.

# Problem zamrznutih sesija (3)

- Problemi koje izazivaju zamrznute sesije:
  - troše resurse servera
  - bezbedonosni rizik
- Dužina intervala čekanja, pre nego što se neka zamrznuta sesija očisti, nije univerzalni parametar i zavisi od potreba aplikacije.

# Odlike mehanizma za upravljanje sesijama na vebu - rezime

- Podaci koji čuvaju stanje aplikacije moraju se skladištiti na serveru.
- Svaki HTTP zahtev mora da sadrži identifikator koji serveru omogućava da obradi prispeli zahtev u kontekstu tekućeg stanja.
- Za sesije mora biti zadato vreme čekanja. U suprotnom, ako korisnik napusti veb lokaciju, nema drugog načina da se završi sesija.

# Upravljanje sesijama u PHP-u

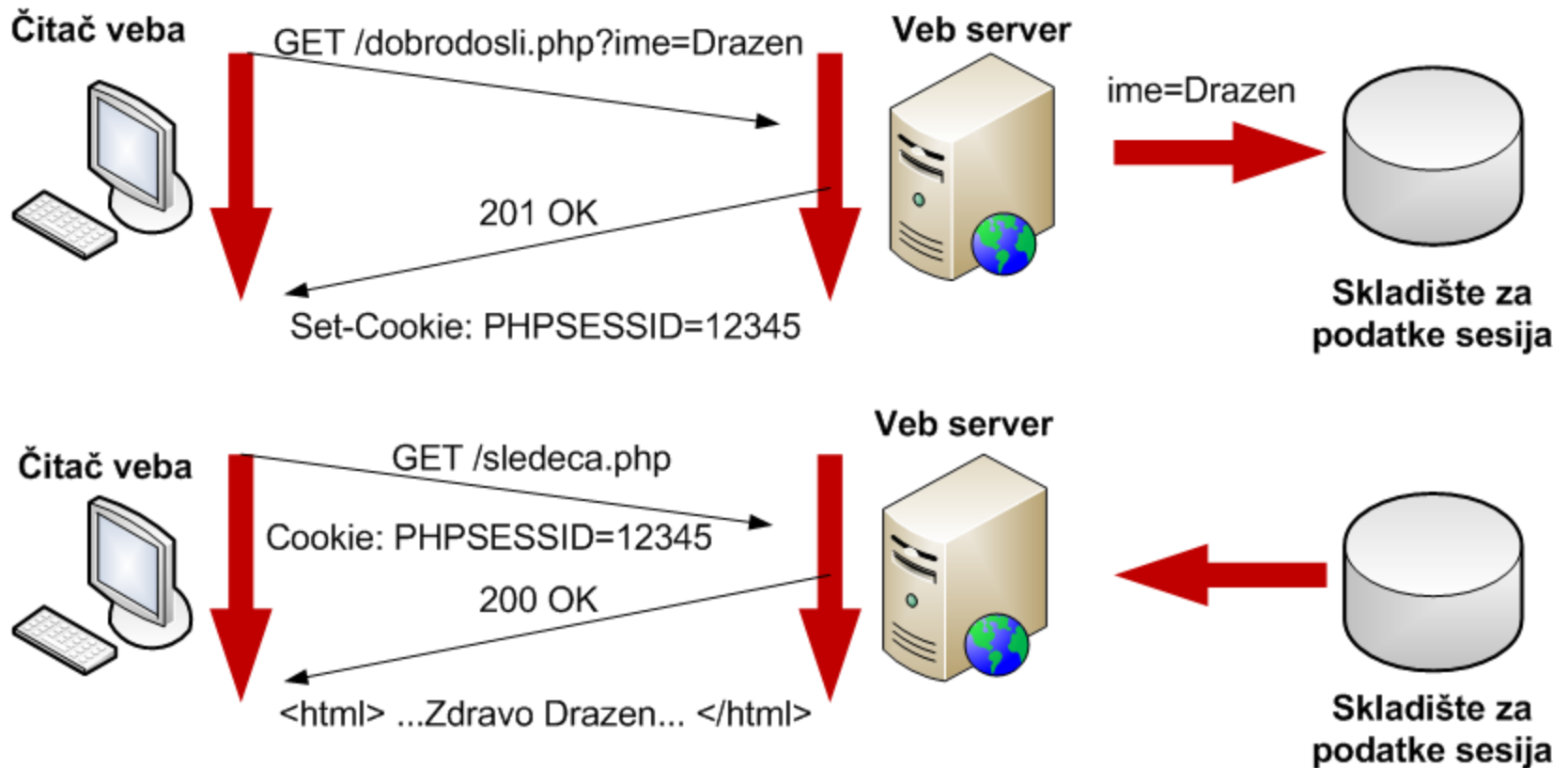
- PHP sadrži skup funkcija koje omogućavaju upravljanje sesijama, što olakšava razvoj aplikacija koje koriste sesije.
- Karakteristične situacije korišćenja sesija:
  - započinjanje nove sesije
  - korišćenje sesijskih promenljivih koje čuvaju stanje aplikacije
  - okončavanje sesije

# Mehanizam sesija (1)



- Kada korisnik prvi put pokrene aplikaciju koja upravlja stanjem, tako što zahteva veb stranicu koja započinje sesiju, PHP radi sledeće stvari:
  - generiše identifikator (ID) sesije i
  - formira datoteku za smeštaj vrednosti promenljivih koje se odnose na novu sesiju
- U odgovor, koji skript generiše, PHP ugrađuje kolačić koji sadrži ID sesije. Klijentski veb pregledač potom skladišti kolačić i umeće njegovu vrednost u sve naredne zahteve koje šalje serveru.

# Mehanizam sesije (2)



Razmena podataka između pregledača i servera, kada se prvi zahtev pošalje aplikaciji koja čuva stanje

# Gde čuvati sesijske promenljive?

- U standardnoj konfiguraciji, PHP upisuje vrednosti sesijskih promenljivih u datoteke na disku, što je pogodno za većinu aplikacija.
- Rešenje koje omogućava veći stepen prilagodljivosti, zasnovano je na upotrebu MySQL-ove baze podataka za čuvanje podataka o sesijama (samo za jako zahtevne aplikacije gde je broj i veličina sesijskih promenljivih izražena).



# Rad sa sesijama



- Započinjanje sesije
- Rad sa sesijskim promenljivama
- Okončavanje sesije

# Započinjanje sesije (1)



- Za započinjanje nove sesije ili učitavanje postojeće sesije koristi se funkcija `session_start()`:

```
bool session_start ()
```

- Ova funkcija kreira novu sesiju i time omogućava pristup superglobalnom nizu `$_SESSION` ili učitava postojeću sesiju koju identifikuje na osnovu ID sesije koji je prosleđen u HTTP zahtevu, putem GET, POST ili cookie.
- Ova funkcija uvek vraća vrednost TRUE.

# Započinjanje sesije (2)



- Ako se koriste sesije zasnovane na kolačićima (gde se identifikator sesije prenosi kao kolačić), funkcija `session_start()` mora biti pozvana pre nego što se bilo koji izlaz uputi veb pregledaču.

# Započinjanje sesije (3)



- Dva slučaja:
  1. u HTTP zahtevu nema identifikatora sesije
  
  2. u HTTP zahtevu ima identifikatora sesije
    1. Odgovarajuća datoteka sesije postoji
    2. Odgovarajuća datoteka sesije ne postoji

# HTTP zahtev nema identifikator sesije

- Generiše se identifikator nove sesije.
- U standardnoj konfiguraciji, u HTTP odgovor se automatski umeće zaglavlje Set-Cookie koji na klijentskom računaru formira kolačić čije je ime PHPSESSID, a vrednost ID sesije (grupa od 32 nasumično generisane heksadecimalne cifre)
- PHP formira datoteku sesije.  
U standardnoj konfiguraciji, datoteke sesija se smeštaju u direktorijum /tmp, a ime datoteke se sastoji od ID sesije i prefiksa sess\_

# HTTP zahtev ima identifikator sesije

- PHP će pokušati da nađe odgovarajuću datoteku sesije kojoj odgovara dati ID sesije.
- **Ako datoteka sesije postoji,** učitava se postojeća sesija, a promenljivama te sesije se može pristupiti sa:  
`$_SESSION["naziv_sesijske_promenljive"]`  
(mogući su i čitanje i upis)
- **Ako datoteka sesije ne postoji** (usled sakupljanja smeća tj. brisanja zamrznutih sesija), PHP formira praznu datoteku.

# Upotreba sesijskih promenljivih (1)

- Ispitivanje da li postoji određena sesijska promenljiva se vrši pomoću funkcije `isset()` na sledeći način:

```
isset($_SESSION["naziv_sesijske_prom"])
```

- Kreiranje nove sesijske promenljive ili dodela nove vrednosti sesijskoj promenljivoj se vrši pomoću naredbe:

```
$_SESSION["naziv_sesijske_prom"]=izraz
```

# Upotreba sesijskih promenljivih (2)

- Brisanje postojeće sesijske promenljive se vrši pomoću funkcije unset():

```
unset($_SESSION["naziv_ses_prom"])
```



# Okončavanje sesije (1)



- Sesija treba da se okonča kada se korisnik odjavi iz aplikacije (izborom opcije "Kraj rada").

Okončavanje sesije vrši se pozivom funkcije `session_destroy()`

# Okončavanje sesije (2)



```
bool session_destroy ()
```

- Funkcija `session_destroy()` uklanja datoteku sesije iz sistema, ali ne uklanja kolačić `$PHPSESSID`.
- Funkcija vraća vrednost `TRUE`, ako je sesija uspešno uklonjena, inače vraća vrednost `FALSE`.

# Upravljanje sesijama bez upotrebe kolačića (1)

- Jedna od izmena koja se može uneti u standardni PHP-ov način upravljanja sesijama jeste da se u zahtev koji se šalje metodom GET ili POST ugradi i vrednost ID sesije kao atribut da bi se izbegla potreba za formiranje kolačića.
- Razlog:  
korisnik može svoj veb pregledač da podesi tako da zabranjuje upotrebu kolačića.

# Upravljanje sesijama bez upotrebe kolačića (2)

- Posledice zabranjivanja upotrebe kolačića:
  - aplikacije koje koriste sesije, kod kojih se ID sesije prenosi putem kolačića, neće raditi
  - svaki put kada se zahteva veb stranica, u odgovarajućem folderu se kreira nova sesija
- Kako napisati aplikacije koje će korektno raditi i kada korisnici u svojim veb pregledačima zabranjuju upotrebu kolačića  
=> ID sesije se ne šalje u obliku kolačića, nego se prosleđuje unutar zahteva metodom GET ili POST!

# Upravljanje sesijama bez upotrebe kolačića (3)

- Funkcija `session_start()` može da koristi promenljivu `$PHPSESSID` bez obzira na to da li je njena vrednost prosleđena u zahtevu metodom GET ili POST.
- Uglavnom se koristi metoda GET.

# Isključivanje kolačića



- PHP-ov sistem za upravljanje sesijama se može podesiti tako da ne formira kolačić PHPSESSID.
- `session.use_cookies = 0` (default 1)  
(`php.ini`)

# Prosleđivanje ID sesije u obliku GET promenljive (1)

- Kada se pošalje prvi zahtev za izvršavanje skripta otvara se nova sesija i formira odgovarajuća datoteka čije bi ime bilo nešto poput:  
sess\_be2202...AE80
- Svi naredni zahtevi trebalo bi da sadrže atribut PHPSESSID i izgledali bi:  
http://localhost/page.php?  
PHPSESSID=be2202...AE80

# Prosleđivanje ID sesije u obliku GET promenljive (2)

- Kako navedeno postići u kodu?
- Skriptovi koji generišu hiperlinkove ka drugim veb stranicama koje koriste sesije moraju da ugrađuju u URL-ove GET atribut čiji je naziv PHPSESSID  
`<a href="abc.php?PHPSESSID=<?=session_id() ?>"`  
ili kraće  
`<a href="abc.php?<?=SID ?>"`
- PHP podešava konstantu SID u oblik pogodan za upotrebu u URL-u kao upitni deo, da bi se olakšalo pisanje URL-ova koji upućuju na skriptove u kojima se koriste sesije.



# Prosleđivanje ID sesije u obliku GET promenljive (3)

- Ako sesija nije kreirana pomoću funkcije `session_start()`, PHP podešava vrednost konstante SID na praznu znakovnu vrednost.
- Ako je sesija kreirana SID će biti oblika `PHPSESSID=be2202...AE80`

# Prosleđivanje ID sesije u obliku GET promenljive (4)

- Umesto da se piše kod koji ugrađuje ID sesije u URL, u PHP-u postoji opcija URL *rewrite*, koja automatski podešava referentne URL-ove tako da sadrže ID sesije kao GET atribut.
- Da bi se ova mogućnost aktivirala (php.ini)  
`session.use_trans_sid = 1 (default 0)`
- Pošto se uključi opcija URL *rewrite*, PHP analizira HTML kod koji skriptovi generišu i automatski dopunjava URL-ove u njemu tako da sadrže atribut PHPSESSID u upitnom delu.

# Prosleđivanje ID sesije u obliku GET promenljive (5)

- Mane opcije URL rewrite:
  - Potrebna je dodatna obrada zbog analiziranja svake generisane stranice
  - Bezbedonosni problemi  
(URL sa identifikatorom sesije se vidi u URL-u, može biti sačuvan u History ili Bookmarks na nekom public računaru,...) => LOŠE!!!

# Sakupljanje smeća (1)



- Treba praviti aplikacije koje korisniku pružaju mogućnost da sam zatvori sesiju, tako što će u skriptu pozvati funkciju `session_destroy()`
- Ne može se garantovati da će se korisnik uvek odjaviti iz aplikacije tako što će zahtevati odgovarajući skript
- U PHP-ov mehanizam upravljanja sesijama ugrađen je mehanizam za sakupljanje smeća, koji obezbeđuje da se datoteke neaktivnih (zamrznutih) sesija posle izvesnog vremena brišu

# Sakupljanje smeća (2)



- Neophodnost sakupljanja smeća:
  - sprečava se da se folder prepuni datotekama sesija, što slabi performanse sistema
  - smanjuje se rizik da neko ko nasumično pogađa ID sesija ukrade neku staru sesiju koja se više ne koristi
- Postoje dva parametra (definisana u datoteci php.ini) koja upravljaju sakupljanjem smeća:
  - `session.gc_maxlifetime` i
  - `session.gc_probability`

# Sakupljanje smeća (3)



- Tokom postupka sakupljanja smeća, ispituju se sve sesije i briše se svaka sesija kojoj niko nije pristupio tokom vremena određenog parametrom `gc_maxlifetime` (default vrednost 1440 sec)
- Parametar `gc_probability` određuje procenat verovatnoće sakupljanja smeća (100% - svaki put kada se pozove funkcija `session_start()`, 1% - sa verovatnoćom 0.01 svaki put kada se pozove funkcija `session_start()` )

# Sakupljanje smeća (4)



- Postupak sakupljanja smeća može poprilično da optereti server, naročito kod veb aplikacija sa velikim brojem korisnika (mora se ispitati datum poslednjeg ažuriranja svake sesije)
- Ako se parametar `gc_probability` podesi suviše visoko nepotrebno se opterećuje server, a ako se podesi suviše nisko javlja se problem zamrznutih sesija i problema koje one uzrokuju

# Sakupljanje smeća (5)



- Generalno, ne postoji pravilo za podešavanje parametara `gc_maxlifetime` i `gc_probability`
- Vrednosti ovih parametara trebalo bi da budu odabrane tako da obezbeđuju ravnotežu između potreba aplikacije i performansi sistema



# Podešavanje sistema za upravljanje sesijama (1)

- Podešavanje sistema za upravljanje sesijama vrši se postavljanjem vrednosti odgovarajućim parametrima koje PHP koristi za upravljanje sesijama u datoteci **php.ini**

# Podešavanje sistema za upravljanje sesijama (2)

- `session.auto_start`  
Određuje da li se sesija započinje automatski, na svakoj stranici. Default: 0 (isključeno)
- `session.cookie_domain`  
Ime domena koje se zadaje u sesijskom kolačiću. Default: ništa
- `session.cookie_lifetime`  
Životni vek kolačića na klijentskom računaru na kome se nalazi identifikator sesije. Default: 0 (dok se ne zatvori veb čitač)

# Podešavanje sistema za upravljanje sesijama (3)

- `session.cookie_path`  
Putanja koja se zadaje u kolačiću sesije.  
Default: /
- `session.name`  
Ime sesije koje se koristi i kao ime kolačića na klijentskom računaru.
- `session.use_cookies`  
Određuje da li sesije koriste kolačiće na klijentskoj strani (0-ne koriste, 1-koriste).  
Default: 1 (sesije koriste kolačiće)

# Podešavanje sistema za upravljanje sesijama (4)

- `session.save_handler`  
Definiše mesto gde se upisuju podaci sesije.  
To može biti i BP, ali u tom slučaju morate napisati vlastite funkcije.  
Default: files (podaci sesije se upisuju u fajlovima)
- `session.save_path`  
Putanja datoteke u koju se upisuju podaci sesije (ako je `session.save_handler=files`).  
Default: /tmp

# Primer sesije (1)



## Stranica page1.php:

```
<?php
 // pocetak sesije
 session_start();
 // postavljanje 3 vrednosti u sesiju
 $_SESSION['color']='red';
 $_SESSION['size']='small';
 $_SESSION['shape']='round';
 print "Uneti podaci u sesiju";
?>
```

# Gde staviti `session_start()` ?

- `session_start()` mora biti u zaglavlju i ne treba ništa da šaljete pregledaču pre toga.
- Najbolje bi bilo da sesiju otpočnete odmah posle `<?php` da ne bi nastali potencijalni problemi.

# Primer sesije (2)



Stranica page2.php:

```
<?php
 // pocetak sesije
 session_start();
 // prikazujemo sta je sacuvano u sesiji
 echo "Boja je ".$_SESSION['color'];
 echo "Velicina je ".$_SESSION['size'];
 echo "Oblik je ".$_SESSION['shape'];
?>
```

# Primer sesije (3)



Sve vrednosti sesije se čuvaju u superglobalnoj promenljivoj - nizu `$_SESSION`, kome smo ovde pristupili.

Drugi način da se ovo prikaže je:

```
<?php
 session_start();
 Print_r($_SESSION);
?>
```



# Funkcija Print\_r()



- Vraća elemente niza koji prosleđujemo kao argument funkcije: **Print\_r (\$your\_array)**

```
<?php
 $imena = array ('a' => 'Aleksandra', 'b' =>
 'Branislav', 'd' => array ('Drasko', 'Dusan'));
 print_r ($imena); ?>
```

**Rezultat je:**

```
Array
(
 [a] => Aleksandra
 [b] => Branislav
 [c] => Array
 (
 [0] => Drasko
 [1] => Dusan
)
)
```

# Primer sesije (4)



- U okviru sesije je moguće sačuvati i niz podataka:

```
<?php
 session_start();
 // pravimo niz
 $boje=array('crvena', 'zuta', 'plava');
 // dodavanje u sesiju
 $_SESSION['boja']=$boje;
 $_SESSION['size']='small';
 $_SESSION['shape']='round';
 print "Uneti podaci u sesiju";
?>
```

# Primer sesije (5)



- U okviru sesije je moguće sačuvati i niz podataka:

```
<?php
```

```
// otvaramo sesiju radi modifikovanja sesije
```

```
session_start();
```

```
// promena varijable u sesiji
```

```
$_SESSION['size']='large';
```

```
//uklanjanje varijable shape iz sesije
```

```
unset($_SESSION['shape']);
```

```
// uklanjanje svih varijabli iz sesije, ali ne
```

```
unistavamo sesiju
```

```
session_unset();
```

```
session_destroy(); // unistavanje sesije
```

```
?>
```



## *Rad sa fajlovima*

# Gde se mogu smeštati fajlovi na serveru?

- Fajlovi na serveru se mogu smeštati:
  - u **bazi podataka**, u kolonama tipa tinyblob (do ~65B), blob (do ~65KB), mediumblob (do ~16MB), longblob (do ~4GB)
  - u **nekom folderu na serveru** (obično u nekom podfolderu foldera u kome se nalazi aplikacija)

# Fajlovi koji podležu logičkoj strukturi baze podataka

- Fajlovi koji podležu logičkoj strukturi baze podataka su fajlovi koji su na neki način povezani sa podacima iz baze.
- Primer:  
u tabeli *Demonstratori* se čuvaju bitni podaci o studentima, a čuvaju se i fajlovi koji predstavljaju slike tih studenata. Svakom studentu odgovara jedna slika.
- Fajlovi koji podležu logičkoj strukturi BP se mogu smeštati u:
  - bazu podataka ili
  - u neki folder na serveru

# Smeštanje fajlova u bazu podataka

## ● Prednosti:

- Svi podaci se nalaze na jednom mestu (u bazi) i povezani su na standardan način
- Nema potrebe voditi računa o problemu poklapanja imena fajlova
- Jedinstveni back-up
- Atomska (nedeljivo) ažuriranje

## ● Nedostaci:

- Nepotrebno se opterećuje DBMS što usporava rad, posebno ako se radi o velikim fajlovima

# Smeštanje fajlova u folder



## ● Prednosti

- Ne opterećuje se DBMS što ubrzava rad

## ● Mane

- Prostorna razdeljenost logički povezanih podataka
- Treba nekako veštački obezbediti povezivanje podataka iz baze i odgovarajućih fajlova
- Treba rešiti potencijalni problem preklapanja imena fajlova
- Odvojeni backup baze i foldera sa fajlovima
- Ažuriranje nije atomsko (nedeljivo)



# Fajlovi koji ne podležu logičkoj strukturi baze podataka

- Fajlovi koji ne podležu logičkoj strukturi baze podataka tj. nisu ni na koji način povezani sa podacima u bazi, smeštaju se u folder na serveru.
- Fajlovi u veb aplikacijama koje ne koriste bazu podataka se smeštaju u folder na serveru.

# Slanje fajlova na server



- Da bi se u okviru HTML forme obezbedila mogućnost za upload fajlova na server, treba uraditi sledeće:
  - U okviru `<form>` taga postaviti sledeće vrednosti atributa:  
`enctype="multipart/form-data"` (tip sadržaja forme)  
`method="post"` (način slanja forme)
  - Unutar `form` taga definisati `input` element koji ispisuje klasičnu "Browse" strukturu  
`<input type="file" name="user_file">`

# Gde se čuva datoteka na serveru

- Kada korisnik pošalje sliku na server (klikom na dugme Submit u okviru forme), PHP automatski prebacuje datoteku na server i čuva je u privremenoj lokaciji na serveru.
- Ova lokacija se definiše parametrom `upload_file_dir` u datoteci `php.ini`.

# Superglobal `$_FILES` (#1)



- Podaci o upload-ovanom fajlu se nalaze superglobalnoj promenljivoj `$_FILES`, koja se automatski inicijalizuje po prijemu podataka putem HTTP post metode.
- `$_FILES` je po strukturi dvodimenzionalni asocijativan niz, gde je prva dimenzija naziv odgovarajućeg input elementa za upload fajla, a druga dimenzija predstavlja naziv određenog atributa fajla.

# Superglobal \$\_FILES (#2)



## ● Za slučaj

```
<input type="file" name="file1">
```

podaci će se nalaziti u `$_FILES["file1"]`

## ● Atributi fajla:

- `$_FILES["file1"]["tmp_name"]` - lokacija fajla na serveru
- `$_FILES["file1"]["name"]` - originalno ime fajla na računaru korisnika
- `$_FILES["file1"]["size"]` - veličina fajla u B
- `$_FILES["file1"]["type"]` - tip datoteke (ako postoji)
- `$_FILES["file1"]["error"]` - podaci o grešci

# Upload velikih fajlova (#1)



- Ako se na server prosleđuju fajlovi veći od nekoliko MB, neophodni su dodatni koraci:

- U HTML formu treba dodati skriveno polje koje će sadržati maksimalnu dozvoljenu veličinu datoteke (u B) koja se može poslati na server:

```
<INPUT TYPE="hidden"
 NAME="MAX_FILE_SIZE" VALUE="100000">
```

- Maksimalna količina memorije koja se može dodeliti PHP skriptu mora biti veća od maksimalne veličine datoteke (parametar `memory_limit` u `php.ini`).

# Upload velikih fajlova (#2)



- Podesiti maksimalnu veličinu datoteke koja se može prebaciti na server (parametar `upload_max_filesize` u `php.ini`)
- Podesiti maksimalnu veličinu POST zahteva tako da bude veća od maksimalne veličine fajla koji se upload-uje (parametar `post_max_size` u `php.ini`)
- Najduže vreme izvršavanja php skriptova treba podasiti treba podasiti na odgovarajuću vrednost koji omogućava da se postupak slanja datoteke obavi do kraja (parametar `max_execution_time` u `php.ini` - po defaultu 30 sec)

# Opšte napomene za smeštanje fajlova u BP

- Upload-ovanom fajlu se pristupa preko atributa "tmp\_name", koji sadrži lokaciju (putanju) fajla na serveru
- U kolonu tipa blob (i tipova izvedenih iz blob) upisuje se **binarni sadržaj** upload-ovanog fajla, a sam upis nije ni po čemu specifičan od upisa podataka u kolone drugih tipova
- Podaci iz kolone tipa blob se brišu kao i svi ostali podaci



# Redosled koraka pri upisu fajlova u BP

1. Proverava se da li je fajl poslat (upload) na server pomoću funkcije `is_uploaded_file()` ako jeste prelazi se na sledeći korak.
2. Proverava se da li je fajl dozvoljene veličine pomoću funkcije `filesize()` ako jeste prelazi se na sledeći korak.
3. Otvara se fajl pomoću funkcije `fopen()`.
4. Čita se binarni sadržaj poslatog fajla pomoću funkcije `fread()` i pamti u nekoj promenljivoj.
5. Binarni sadržaj poslatog fajla (sadržan u promenljivoj) se prečišćava pomoću funkcije `AddSlashes()`.
6. Upis podataka u bazu, pri čemu se u kolonu tipa blob (ili tipa izvedenog iz blob) upisuje prečišćeni binarni sadržaj poslatog fajla koji se čuva u promenljivoj.

# Prikaz slike smeštene u BP (#1)

- Prikaz slike smeštene u BP (u koloni tipa blob) se postiže tako što se okviru `<img>` taga atributu `src` dodeliti vrednost koja predstavlja putanju skripta koji čita i prikazuje sadržaj slike, uz dodatne parametre koji mu se prosleđuju:

```

```

# Prikaz slike smeštene u BP (#2)

- Skript `read_and_show_image.php` na osnovu prosleđenog parametra `id_item` (u opštem slučaju se prosleđuje više parametara) čita binarni sadržaj slike iz baze.
- Slika se prikazuje tako što se binarni sadržaj slike dobijen iz baze "štampa" pomoću naredbe `echo`:  
`echo $binary_content;`
- Neposredno pre štampanja slike naredbom `echo` veb pregledač se šalje zaglavlje koje definiše MIME tip fajla (resursa) koji se štampa pomoću funkcije `header` (type atribut u `$_FILES["file_name"]`)  
`header("Content-Type: image/jpeg");`

# Smeštanje fajlova u folder - opšte napomene

- Ideja: u bazi se ne čuva sama slika nego informacija pomoću koje je moguće locirati sliku na serveru (tj. odrediti putanju slike).
- Potrebno je obezbediti:
  - povezivanje podataka iz baze sa odgovarajućim fajlovima i
  - jedinstvenost imenovanja fajlova (kako ne bi došlo do prepisivanja ili do nemogućnosti upload-ovanja)
- Ovi problemi se mogu rešiti primenom određenih konvencija.

# Rešenje problema - konvencija

- U tabeli umesto kolone tipa blob staviti kolonu tipa varchar u kojoj se pamti originalno ime fajla.
- Svi fajlovi se čuvaju u jednom folderu db\_files.
- Za svaku tabelu u bazi koja "logički" sadrži fajlove kreira se poseban podfolder u folderu db\_files čiji je naziv jednak nazivu tabele.
- Za svaku kolonu u tabeli u kojoj se pamti originalno ime fajla kreira se podfolder u folderu koji odgovara toj tabeli čiji je naziv jednak nazivu kolone.
- Fajl se preimenuje, tako što mu se kao prefiks doda vrednost primarnog ključa iz njemu odgovarajućeg reda u tabeli.

# Redosled pri upisu podataka



1. Izvrši se upis podataka u bazu;  
ako tabela "logički" sadrži fajlove u odgovarajuće kolone se upisuju originalna imena tih fajlova.  
Ako je upis uspeo prelazi se na sledeći korak.
2. Smeštanje upload-ovanih fajlova u njima odgovarajuće foldere i izmena imena tih fajlova (u skladu sa uvedenom konvencijom).

# Redosled koraka pri smeštanju upload-ovanih fajlova u foldere

1. Provera da li je korisnik upload-ovao neki fajl, pomoću funkcije `is_uploaded_file()`
2. Provera da li je fajl dozvoljenog tipa i da li poseduje adekvatnu ekstenziju (opciono), ispitivanjem `type` i `name` atributa fajla
3. Provera da li je fajl dozvoljene veličine, ispitivanjem `size` atributa fajla ili korišćenjem `filesize()` funkcije
4. Provera da li odredišni direktorijum postoji (trebalo bi da uvek postoji), korišćenjem funkcija `file_exists()` i `is_dir()`
5. Provera da li već postoji fajl sa istim imenom u odredišnom folderu (ne bi trebalo nikad da postoji), pomoću funkcije `file_exists()`
6. Premestiti fajl u odredišni folder (u skladu sa konvencijom), korišćenjem funkcije `move_uploaded_file()`
7. Proveriti da li se premeštanje fajla izvršilo bez greške, ispitivanjem povratne vrednosti funkcije `move_uploaded_file()`

# Prikaz slike smeštene u folderu

- Prikaz slike smeštene u folderu se postiže tako što se okviru `<img>` taga atributu `source` dodeliti vrednost koja predstavlja relativnu (u odnosu na tekući skript) putanju slike

```

```



# Redosled pri brisanju podataka



1. Pročita se vrednost primarnog ključa i atributa koji predstavlja originalno ime fajla iz reda koji treba obrisati, a zatim se red obriše.
2. Obriše sa fajl iz odgovarajućeg foldera koji je odgovarao ovom redu. Folder se može jednoznačno odrediti ako se koristi navedena konvencija.

# Redosled koraka pri brisanju upload-ovanih fajlova iz foldera

1. Proverava se da li postoji fajl koji odgovara redu koji se briše u odgovarajućem folderu, korišćenjem funkcije `file_exists()`.  
Ako postoji prelazi se na sledeći korak, u suprotnom se ništa ne preduzima.
2. Briše se postojeći fajl koji odgovara obrisanom redu u tabeli, korišćenjem funkcije `unlink()`.
3. Proverava se da li je brisanje fajla uspešno izvršeno, ispitivanjem povratne vrednosti funkcije `unlink()`.

# Funkcije za rad sa direktorijumima (1)

## ● Čitanje sadržaja direktorijuma

- `opendir()`, `closedir()` i `readdir()`

- **Primer:**

```
$tekuci_dir = '/uploads/';
$dir = opendir($tekuci_dir);
echo "<p>Listing direktorijuma:</p>";
while ($fajl = readdir($dir)) {
 echo "$fajl";
}
echo "";
closedir($dir);
```

# Funkcije za rad sa direktorijumima (2)

- Učitvanje podataka o tekućem direktorijumu
  - `dirname ($putanja)`, `basename ($putanja)`
  - Ove funkcije daju deo putanje koji prikazuje sadržaj direktorijuma, odnosno ime datoteke.

# Funkcije za rad sa direktorijumima (3)

## ● Pravljenje i brisanje direktorijuma

- `mkdir()`, `rmdir()`
- Funkciji za kreiranje direktorijuma potrebno je da prosledite dva parametra: putanju ka direktorijumu, koja sadrži i ime novog direktorijuma i ovlašćenja za pristup tom direktorijumu, na primer:  
`mkdir("/tmp/testing", 0777);`
- Direktorijum koji nameravate da brišete, mora biti prazan.

# Rad sa sistemom datoteka (1)

## ● Dobijanje podataka o datoteci

- Funkcije `fileatime()` i `filemtime()` vraćaju vremensku oznaku kada je poslednji put pristupano datoteci, odnosno direktorijumu.
- Funkcije `fileowner()` i `filegroup()` vraćaju identif. vlasnika (`uid`), odnosno vlasničke grupe (`gid`).
- Funkcija `fileperms()` vraća ovlašćenja za rad sa datotekom.
- Funkcija `filetype()` vraća nekoliko podataka o tipu datoteke koja se ispituje (`fifo`, `char`, `dir`, `block`, `link`, `file` i `unknown`)
- Funkcija `filesize()` vraća veličinu datoteke u bajtovima.

# Rad sa sistemom datoteka (2)

- Dobijanje podataka o datoteci
  - Funkcije koje ispituju vrednost određenog atributa datoteke vraćaju `true` ili `false`:
    - `is_dir()`
    - `is_executable()`
    - `is_file()`
    - `is_link()`
    - `is_readable()`
    - `is_writable()`

# Rad sa sistemom datoteka (3)

- Pravljenje, brisanje i premeštanje datoteka
  - Pomoću funkcije `touch` možete napraviti novu datoteku ili izmeniti vreme kada je poslednji put menjan sadržaj postojeće datoteke.
  - Datoteke se brišu pomoću funkcije `unlink($ime_datoteke)`. Prototip funkcije:  
`unlink($ime_datoteke)`
  - Datoteke možete kopirati i premeštati pomoću funkcija `copy` i `rename`. Prototip tih funkcija izgleda ovako:  
`copy($izvorna_putanja, $ciljna_putanja)`  
`rename($staro_ime, $novo_ime)`
  - Funkcija `rename()`, osim preimenovanja, radi premeštanje datoteke s jednog mesta na drugo, jer PHP nema posebnu funkciju za premeštanje!





## *Upotreba mrežnih funkcija i protokola*

# Funkcija header()

- `void header($string)`
- Šalje sirovo HTTP zaglavlje.
- Funkcija mora biti pozvana pre nego što se pošalje bilo koji izlaz - HTML tagovi, blanko linije u fajlu ili iz PHP.  
To je vrlo česta greška kada se koriste `include/require` ili neke druge funkcije za pristup fajlu.
- Postoje 2 specijalna slučaja pozivanja ove funkcije.

# Pozivanje funkcije header()

- Prvi slučaj:

Parametar koji se prosleđuje funkciji počinje sa **HTTP/**, a nakon toga se piše HTTP statusni kod za slanje:

```
header("HTTP/1.0 404 Not Found");
```

ili kraće: 

```
header("Status: 404 Not Found");
```

- Drugi slučaj:

Kao parametar se navodi ključna reč **Location:**, a nakon toga lokacija na koju se prelazi (tačnije veb čitač redirektuje na drugu veb stranicu):

```
header("Location: welcome.php");
```

# Slanje i primanje poruka e-poštom

- Upotreba jednostavne funkcije `mail()`
- Šalje poruke preko protokola SMTP (*Simple Mail Transfer Protocol*)
- Protokol SMTP omogućava samo slanje poruka
- Protokoli IMAP (*Internet Message Access Protocol*) i POP (*Post Office Protocol*) omogućavaju čitanje poruka sa servera za e-poštu, ali ne i slanje poruka

# Funkcija mail()



Metod	Opis
To	Adresa e-pošte na koju se šalje poruka.
Subject	Naslov poruke.
Message	Tekst (sadržaj) poruke.
Additional Headers	Opciono: dodatna zaglavlja (From, Reply-To,...)
Additional Parameters	Opciono: bilo koji dodatni parametri koje želite da pošaljete mejl serveru

# Primer - Slanje mejla (1)



`//posalji_mejl.php`

```
<?php
 if (!array_key_exists('Submitted', $_POST)) {
?>
<form method="post" action="Mail.php">
 <input type="hidden" name="Submitted" value="true"/>
 Mail Server: <input type="text" name="Host" size="25"/>

 To: <input type="text" name="To" size="25"/>

 From: <input type="text" name="From" size="25"/>

 Subject: <input type="text" name="Subject" size="25"/>

 <textarea name="Message" cols="50" rows="10">
 </textarea>

 <input type="submit" value="Send Email"/>
</form>
```

# Primer - Slanje mejla (2)

`//posalji_mejl.php`

```
<?php
} else {
 ini_set('SMTP', $_POST['Host']);
 $To = $_POST['To'];
 $From = 'From: ' . $_POST['From'];
 $Subject = $_POST['Subject'];
 $Message = $_POST['Message'];

 if(mail($To, $Subject, $Message, $From)) {
 echo "Message Sent";
 } else {
 echo "Message Not Sent";
 }
}
?>
```

- Da bi ovaj primer radio morate imati podešen mejl server. Funkcija `ini_set()` postavlja adresu mejl servera.

# Ograničenja mail() funkcije

- Nema podrške za SMTP autentifikaciju.
- Teško se šalju poruke formatirane u HTML-u.
- Teško se prosleđuju prilozi u poruci (*attachments*).

Šta je rešenje?

Koristiti mnogobrojne besplatne klase koje služe za slanje mejlova!



# PHPMailer



- Vrlo dobra ekstenzija za slanje e-pošte je PHPMailer. Dostupan je besplatno:  
<http://phpmailer.sourceforge.net>
- U sledećem primeru je pokazano kako se radi sa PHPMailer klasom.

# PHPMailer metode



Metoda	Opis
<code>AddAddress ()</code>	Dodaje polje "to".
<code>AddAttachment ()</code>	Dodaje datoteku za slanje zadavanjem putanje u fajl sistemu.
<code>AddBCC ()</code>	Dodaje polje "bcc".
<code>AddCC ()</code>	Dodaje polje "cc".
<code>AddReplyTo ()</code>	Dodaje polje "Reply-to".
<code>IsHTML ()</code>	Postavlja tip poruke na HTML.
<code>IsSMTP ()</code>	Kaže svojoj klasi da šalje poruke preko SMTP.
<code>Send ()</code>	Funkcija koja kreira poruku, šalje je i vraća true vrednost, a u slučaju da ne pošalje poruku uspešno vraća false.

# PHPMailer atributi



Atribut	Opis
AltBody	Postavljanje teksta kao tela poruke.
Body	Postavljanje tela poruke, može biti tekst ili HTML.
ErrorInfo	Poslednje poruke o greškama.
From	Postavlja adresu e-pošte pošiljaoca.
FromName	Dodaje ime pošiljaoca u poruci koja se šalje.
Host	Postavlja SMTP host. Svi hostovi moraju biti razdvojeni sa ;
Password	Postavlja SMTP lozinku.
SMTPAuth	Postavlja SMTP autentifikaciju, koristeći username i password.
Subject	Dodaje naslov poruke.
Username	Postavlja SMTP korisničko ime.
WordWrap	Postavlja odsecanje teksta u telu poruke, nakon zadatog broja znakova.

# Primer - Korišćenje PHPMailer (1)

```
<?php // posalji_gmej1.php
 if (!array_key_exists('Submitted', $_POST)) { ?>
<form method="post" action="posalji_gmej1.php">
<input type="hidden" name="Submitted" value="true"/>
 Mail
 Server: <input type="text" name="Host" size="25"/>

 If authentication is required:

 Username: <input type="text" name="Username" size="25"/>

 Password: <input type="password" name="Password" size="10"/>
<hr/>
 To: <input type="text" name="To" size="25"/>

 From Email: <input type="text" name="From" size="25"/>

 From Name: <input type="text" name="FromName" size="25"/>

 Subject: <input type="text" name="Subject"
size="25"/>

 <textarea name="Message" cols="50" rows="10"></textarea>

 Using HTML: <input type="checkbox" name="HTML"/>
 <input type="submit" value="Send Email"/>
</form>
```

# Primer - Korišćenje PHPMailer (2)

```
<?php }
 else { require("class.phpmailer.php");
 $To = $_POST['To'];
 $From = $_POST['From'];
 $FromName = $_POST['FromName'];
 $Subject = $_POST['Subject'];
 $Message = $_POST['Message'];
 $Host = $_POST['Host'];
 if (array_key_exists('HTML', $_POST)) {
 $HTML = true;
 $Mail->Username=$_POST['Username']; //Gmail nalog
 $Mail->Password=$_POST['Password']; //Gmail sifra
 } else {
 $HTML = false;
 }
 $Mail = new PHPMailer();
 $Mail->IsSMTP(); // send via SMTP
 $Mail->Host = $Host; //SMTP server
```

# Primer - Korišćenje PHPMailer (3)

```
if (array_key_exists('Username', $_POST)) {
 $Mail->SMTPAuth=true; // dozvola SMTP autent.
} else { $Mail->SMTPAuth=false; }
$Mail->SMTPSecure = "ssl"; //postavljanje prefiksa serveru
$Mail->Host = "smtp.gmail.com"; // GMail SMTP
$Mail->Port = 465; //GMail SMTP port

$Mail->From = $From; //e-mail posaljioca
$Mail->FromName = $FromName; //ime posiljaoca
$Mail->AddAddress($To);
$Mail->AddReplyTo($From);
$Mail->WordWrap = 50;
$Mail->IsHTML($HTML);
$Mail->Subject = $Subject;
$Mail->Body = $Message;
```

# Primer - Korišćenje PHPMailer (4)

```
if ($Mail->Send())
{
 echo "Poruka poslata";
}
else
{
 echo "Poruka nije poslata
";
 echo "GRESKA: " . $Mail->ErrorInfo;
}
}
?>
```

# Upotreba FTP protokola



- FTP (*File Transfer Protocol*) je protokol za prenošenje datoteka između umreženih računara.
- PHP podržava i FTP veze, slično kao i HTML veze.
- FTP se koristi za izradu rezervnih kopija datoteka koje čine veb prezentaciju ili pri preslikavanju (engl. *mirroring*) tih datoteka na drugu veb lokaciju.





## *Rad sa datumom i vremenom*

# Funkcija `date()`

- Funkcija `date()` ima dva parametra:
  - vrednost znakovnog tipa, koja predstavlja zahtevani format rezultata
  - vrednost u formatu Unixove vremenske oznake
- Tipičan primer funkcije `date()`:  
`echo date( 'd.m.Y.' );`
- Rezultat će biti datum u obliku: 22.03.2012.
- Neki šabloni za formate navedeni su u sledećoj tabeli.

# Šabloni za date() funkciju (1)

Kod	Opis
a	Prikazuje <b>am</b> ili <b>pm</b> , u zavisnosti od doba dana
A	Prikazuje <b>AM</b> ili <b>PM</b> , u zavisnosti od doba dana
c	Datum u formatu ISO8601: <b>GGGG-MM-DD-TČČ:MM:SS+GMTtime</b> Primer: 2012-03-22-T22:45-21:45
d	Dan u mesecu datuma, kao dvocifren broj sa vodećom nulom. Opseg: <b>01-31</b>
D	Dan u nedelji kao skraćenica od 3 slova. Opseg: <b>Mon-Sun</b>
F	Puno englesko ime meseca. Opseg: <b>January-December</b>
g	Čas u 12-časovnom formatu, bez vodeće nule. Opseg: <b>1-12</b>
G	Čas u 24-časovnom formatu, bez vodeće nule. Opseg: <b>0-23</b>
h	Čas u 12-časovnom formatu, sa vodećom nulom. Opseg: <b>01-12</b>
H	Čas u 24-časovnom formatu, sa vodećom nulom. Opseg: <b>00-23</b>
i	Minuti u času sa vodećom nulom. Opseg: <b>00-59</b>
l	Režim računanja vremena. <b>1 za letnje računanje vremena, 0 za zimsko.</b>

# Šabloni za date() funkciju (2)

Kod	Opis
j	Dan u mesecu bez vodeće nule. Opseg: <b>1-31</b>
l	Puno (englesko) ime dana u nedelji. Opseg: <b>Monday-Sunday</b>
L	Podatak o tome da li je godina prestupna. Vraća vrednost <b>1 za datum u prestupnoj godini, a 0 ako datum nije u prestupnoj godini.</b>
m	Mesec u godini kao dvocifren broj sa vodećom nulom. Opseg: <b>01-12</b>
M	Mesec u godini kao skraćénica od 3 slova. Opseg: <b>Jan-Dec</b>
n	Mesec u godini kao broj bez vodeće nule. Opseg: <b>1-12</b>
O	Razlika u časovima između vremena u tekućoj zoni i GMT.
r	Datum i vreme u formatu RFC822. Primer: <b>Sun, 15 Apr 2012 11:55:30 +0100</b>
s	Sekunde u minutu sa vodećom nulom. Opseg: <b>00-59</b>
S	Sufiks (engleski) koji opisuje redni broj dana. Iz skupa: <b>st, nd, rd, th</b>
t	Ukupan broj dana u mesecu. Opseg: <b>28-31</b>
T	Vremenska zona servera u obliku skraćénice od 3 znaka. Primer: EST

# Šabloni za date() funkciju (3)

Kod	Opis
U	Ukupan broj sekundi koji je protekao od 1.januara 1970.godine do tekućeg trenutka. Ovo je poznat Unixov format vremenske oznake.
w	Redni broj dana u sedmici kao jednocifren broj. Opseg: <b>0 (nedelja) do 6 (subota)</b>
W	Redni broj sedmice u godini, prema standardu ISO-8601.
y	Godina kao dvocifren broj. Na primer: <b>12</b>
Y	Godina kao četvorocifren broj. Na primer: <b>2012</b>
z	Redni broj dana u godini kao broj. Opseg: <b>0-365</b>
Z	Pomak tekuće vremenske zone, u sekundama. Opseg: od -43200 do 43200

# Unix-ove vremenske oznake

- Vreme u formatu Unix-ove vremenske oznake predstavlja tzv. vremensku marku (engl. *timestamp*)
- Na većini sistema koji rade pod Unix-om tekući datum i vreme čuvaju se u obliku 32-celobrojne vrednosti koja predstavlja ukupan broj sekundi koje su protekle od ponoći 1.januara 1970.godine, po griničkom vremenu, što je poznato i kao *Unix-ova epoha*.
- Izgleda čudno... ali ovo je standard 😊

# Prednosti i nedostaci Unix formata

## ● Prednost:

- Čuvanje podataka o datumu i vremenu u sažetom obliku
- Na ovaj format ne utiče problem “milenijumske bube” (Y2K) koji ugrožava neke druge sažete ili skraćene formate za datum.

## ● Nedostaci:

- Pomoću 32-bitne vrednosti je ograničen opseg datuma: softver ne može da evidentira događaje pre 1902.god. ili posle 2038.god. (Windows ne podržava negativan opseg!)

- Funkcija `date()` i mnoge druge PHP-ove funkcije koriste ovaj standard, čak i kad se PHP koristi na serveru koji je pod Windows-om.

# Funkcija za konverziju u Unix format

- `int mktime ([int sati [, int minuti [, sekunde [, int mesec [, int dan [, int godina [, int letnje_r]]]]]])`
- Ova funkcija služi za pretvaranje podatka o datumu i vremenu u Unix-ovu vremensku oznaku.
- `letnje_r` je letnje računanje vremena i ima vrednost 1, ako se primenjuje, odnosno 0 ako se ne primenjuje, ili -1 ako ne znate.
- Zamerka:  
Redosled parametara je neintuitivan, ne dopušta izostavljanje parametara koji predstavlja vreme. Ako vreme nije bitno, možete zadati nule kao vrednosti parametara sati, minuti, sekunde.



# Funkcija `getdate()`



- Funkcija ima sledeći prototip:  
`array getdate([int vremenskaoznaka])`
- Funkcija razlaže opcionu vrednost vremenske oznake u niz, koja se sastoji od nekoliko komponentata, tj. podataka o datumu i vremenu.
- Ako funkciju pozovete bez datuma, dobićete vremensku oznaku tekućeg datuma i vremena.

# Komponente niza koji vraća getdate()

Ključ	Vrednost
seconds	Sekunde, numerička vrednost
minutes	Minuti, numerička vrednost
hours	Sati, numerička vrednost
mday	Dan u mesecu, numerička vrednost
wday	Redni broj dana u sedmici, numerička vrednost
mon	Mesec, numerička vrednost
year	Godina, numerička vrednost
yday	Redni broj dana u godini, numerička vrednost
weekday	Ime dana u sedmici, tekst
month	Puno ime meseca, tekst
0	Vremenska oznaka, numeričkog tipa

# Provera ispravnosti datuma

- Pomoću funkcije `checkdate()` možete utvrditi da li je datum ispravan (ovo je korisno kada korisnik unosi datum preko forme).
- Funkcija `checkdate()` ima sledeći prototip:  
`int checkdate (int mesec, int dan, int godina)`  
Primeri: `checkdate(4,15,2012) //true`  
`checkdate(4,31,2012) //false`
- Funkcija ispituje:
  - Da li je godina ispravna celobrojna vrednost u opsegu od 0 do 32767?
  - Da li je mesec celobrojna vrednost u opsegu od 1 do 12?
  - Da li zadati dan postoji u zadanom mesecu?

# Konverzija datuma između PHP i MySQL

- U MySQL datum i vreme u formatu ISO8601
- ISO8601 datum počinje godinom: GGGG-MM-DD
- Na primer:  
10.april 2012. zadaje se kao 2012-04-10  
ili kao 12-04-10
- Zbog toga je za razmenu datuma potrebna konverzija PHP ↔ MySQL
- Konverzija se vrši ili u PHP ili u MySQL

# DATE\_FORMAT() u MySQL-u

- Ako želite da konvertujete datume ili vreme u MySQL-u, koristite funkcije `DATE_FORMAT()` i `UNIX_TIMESTAMP()`.
- Funkcija `DATE_FORMAT()` radi na sličan način kao ekvivalentna PHP-ova funkcija, ali `date()` koristi drugačije šablone za formate.
- Evropski format: DD-MM-GGGG  
ISO (MySQL) format: GGGG-MM-DD
- Prototip funkcije:  

```
SELECT DATE_FORMAT(datumska_kolona,
'%d. %m. %Y') FROM tabela;
```

# Oznake u funkciji DATE\_FORMAT (1)

Kod	Opis
%M	Puno ime meseca
%W	Puno ime dana u nedelji
%D	Dan u mesecu, kao broj kome sledi tekstualni sufiks (Primer: 1st, 2nd, 3rd,...)
%Y	Godina kao četvorocifreni broj (Primer: 2012)
%y	Godina kao dvocifreni broj
%a	Skraćeno ime dana, do tri slova
%d	Dan u mesecu kao broj sa vodećom nulom
%e	Dan u mesecu kao broj bez vodeće nule
%m	Mesec kao broj sa vodećom nulom
%c	Mesec kao broj bez vodeće nule
%b	Skraćeno ime meseca sastavljeno od tri slova
%j	Redni broj dana u godini

# Oznake u funkciji DATE\_FORMAT (2)

Kod	Opis
%H	Čas u 24-časovnom formatu sa vodećom nulom
%k	Čas u 24-časovnom formatu bez vodeće nule
%h ili %l	Čas u 12-časovnom formatu sa vodećom nulom
%l	Čas u 12-časovnom formatu bez vodeće nule
%i	Minuti u času kao broj sa vodećom nulom
%r	Vreme u 12-časovnom formatu (Primer: hh:mm:ss [AM PM] )
%T	Vreme u 24-časovnom formatu (Primer: hh:mm:ss)
%S ili %s	Sekunde u minutu kao broj sa vodećom nulom
%P	Deo dana, AM ili PM
%w	Redni broj dana u nedelji kao broj u opsegu od 0 (nedelja) do 6 (subota)

# Funkcija UNIX\_TIMESTAMP()

- Funkcija `UNIX_TIMESTAMP()` radi na sličan način ali pretvara vrednosti iz kolone datumskog tipa u Unix-ovu vremensku oznaku.
- Prototip funkcije:  

```
SELECT UNIX_TIMESTAMP(datumska_kolona)
FROM tabela;
```
- Vraća datum u vidu vremenske marke, pa se u PHP-u može koristiti po želji.
- Proračuni i poređenja datuma su jednostavni kada su u Unix-ovom formatu!



# Primer - Broj godina



- Na osnovu unetog datuma rođenja u PHP kodu, izračunati starost osobe u godinama.

# Primer - Rešenje



```
<?php
 $dan=18;
 $mesec = 9;
 $godina = 1988;

 //formiramo datum rođenja u Unix formatu
 $rođendan = mktime(0,0,0,$mesec,$dan,$godina);

 //formiramo tekuci datum u Unix formatu
 $trenutno = time();

 $godine_unix = $trenutno - $rođendan; //razlika

 //konverzija iz sekundi u godinu
 $godine = floor($godine_unix / (365*24*60*60));
 echo "Broj godina je $godine";
?>
```



*Ostale korisne  
mogućnosti*

# Funkcija `eval()`



- Funkcija `eval()` obrađuje prosleđeni znakovni niz kao PHP kod. Primer:  

```
eval ("echo `Zdravo svete` ;");
```
- Rezultat će biti isti kao da smo napisali:  

```
echo `Zdravo svete` ;
```
- Zašto je korisna ova funkcija?
  - Ako u bazi čuvamo blokove koda koje ćemo potom učítavati i izvršavati pomoću funkcije `eval()`
  - Generisanje koda u petlji i izvršavanje pomoću funkcije `eval()`

# Funkcija `eval()` - Prednosti



- Funkcija `eval()` obično se koristi kao deo sistema šablona.
- Iz baze podataka može da se učita mešavina HTML koda, PHP koda i običnog teksta. Vaš sistem šablona može da formatira taj blok podataka i da pomoću funkcije `eval()` izvrši PHP kod iz njega.
- Funkcija `eval()` primenjuje se pri ažuriranju ili ispravljanju postojećeg koda.

# Komande die i exit

- Prekid izvršavanja skripta pomoću `die` i `exit`
- Komanda `exit` ne vraća ništa. Drugo ime za tu komandu je `die`. Ove komande mogu da se komanduju sa operatorom `OR`.

- Primer:

```
exit;
exit('Izvršavanje skripta je prekinuto');
```

- Česte greške nastaju kada želite da uspostavite vezu sa bazom ili izvršite neki upit nad bazom. Dobro je da tada korisnika obavestite o grešci:

```
mysql_query($upit) or die("Upit ne moze da se izvrši" . mysql_error());
```



## *Zaštita aplikacije*

# Problem



- Nikada se ne pouzdajte ni u šta nad čim nemate potpunu kontrolu.
- Podaci korisničkog unosa se ne smeju prihvatiti bez provere (tj. upisati u BP ili neki fajl na serveru ili proslediti dalje sistemu pomoću neke izvršne komande).
- Problemi:
  - Zaštita baze podataka
  - Zaštita od zlonamernih komandi
  - Zaštita od zlonamernih skriptova



# Zaštita baze podataka (#1)

- **P1:** Znakovi `"`, `'`, `\`, **NULL** mogu da izazovu probleme prilikom upisivanja u BP, zato što BP može da ih protumači kao upravljačke znakove.
- **R1:** Funkcija `addslashes()` pretvara sve ove znakove u izlazne sekvence dodajući ispred njih znak `\`.  
Funkcija `stripslashes()` vrši inverznu obradu tj. uklanja znakove `\` (vraća podatke u izvorni oblik).

# Zaštita baze podataka (#2)

- Napomena: `magic_quotes_gpc on` (php.ini)
  - PHP automatski poziva f-ju `addslashes()` za sve ulazne GET, POST i COOKIE promenljive.
  - PHP automatski poziva funkciju `stripslashes()` pre slanja podataka na izlaz.
  - Opcija je po defaultu postavljena na vrednost `on` u novijim verzijama PHP-a.
  - Ne može biti postavljena u runtime-u.
  - Pozivom funkcije `get_magic_quotes_gpc()` dobija se status promenljive `magic_quotes_gpc`.

# Zaštita baze podataka (#3)

- Napomena: `magic_quotes_runtime` on (php.ini)
  - PHP poziva funkciju `f-ju addslashes()` za sve podatke pročitane iz eksternih izvora (BP, tekst fajlovi, XML fajlovi).
  - Opcija je po defaultu postavljena na vrednost `off` u novijim verzijama PHP-a.
  - Može biti postavljena u runtime-u pozivom funkcije `set_magic_quotes_runtime()`.
  - Pozivom funkcije `get_magic_quotes_runtime()` dobija se status promenljive `magic_quotes_runtime`.

# Primer



```
$str1 = "Is your name O'reilly?";
$str2 = addslashes($str1);
echo($str2); // Outputs: Is your name O\'reilly?
$str3 = stripslashes($str2);
echo($str3); //Outputs: Is your name O'reilly?
```

# Zaštita od zlonamernih komandi

- **P2:** Ako se podaci korisničkog unosa prosleđuju funkcijama `system()` ili `exec()` koje izvršavaju na veb serveru komandu koja im se prosledi kao parametar, pomoću određenih metaznakova zlonamerni korisnici mogu da izvrše proizvoljne naredbe na sistemu.
- **R2:** Funkcija `escapeshellcmd()` pretvara sve ove znakove u izlazne sekvence dodajući ispred njih znak `\`.

# Zaštita od zlonamernih komandi

- **P3:** Korisnici mogu da ugrade štetne skriptove (HTML i PHP oznake) u parametre koje prosleđuju što može dovesti do grešaka u prikazu veb stranica koje šalju te podatke na izlaz.
- **R3:** F-ja `strip_tags()` uklanja sve HTML i PHP oznake iz znakovnih podataka. F-ja `htmlspecialchars()` pretvara znakove koji u HTML-u nešto znače u odgovarajuće HTML specijalne znake tj. entitete (& => &amp; , " => &quot; , ' => &#039; , < => &lt; , > => &gt; )

# Primer



```
$text = 'Hello world';

echo strip_tags($text);
// Output: Hello world

echo strip_tags($text, '');
// Output: Hello world
// => Hello world (u browser-u)

echo htmlspecialchars($text);
// Output: Hello world
// => Hello world (u browser-u)
```

# Uklanjanje krajnjih belina



- **P4:** Vrednost korisnički unetog parametra sadrži krajnje leve i krajnje desne beline.
- **R4:** Pomoću funkcije `trim()` uklanjaju se krajnje beline sa obe strane stringa.



# Ograničavanje dužine stringova

- **P5:** Dužina (broj karaktera) vrednosti nekog parametra korisničkog unosa ne zadovoljava ograničenja baze podataka.
- **R5:** Pomoću `substr()` ograničava se dužina korisničkog parametra odsecanjem karaktera sa desne strane ili se prekida izvršavanje i traži od korisnika ponovni unos podatka.

# Značaj provere podataka



- Ključno je obezbediti da svi podaci zadovoljavaju:
  - Sve zahteve korisnika i sistema
  - Ograničenja definisana u bazi podataka

# Mesta provere podataka



- Provera na klijentskoj strani - opciono  
Najčešće korišćenjem Java Script-a!
- Provere podataka u srednjem sloju  
(na serverskoj strani) - obavezno  
Korišćenjem PHP!
- Provera podataka u bazi (na serverskoj strani) -  
automatski DBMS vrši ovu proveru.

# Provera podataka na klijentskoj strani

- Obavlja se u klijenskom veb pregledaču najčešće pomoću JavaScript-a.
- Obično se vrši provera vrednosti elemenata forme.
- Nedostatak:  
Zavisí od korisnika i njegovog okruženja - korisnik može da zabrani upotrebu JavaScript-a, kao i da namerno ili nenamerno zaobiđe proveru ispravnosti podataka.

# Provera podataka u srednjem sloju

- Provera se vrši u skriptu srednjeg sloja (PHP skriptu) i predstavlja glavni način provere.
- Ovo se ne sme izostaviti ni u jednoj veb aplikaciji.

# Provera podataka u bazi



- Provera se vrši automatski pri ažuriranju baze, a proverava se zadovoljenost ograničenja definisanih u bazi.
- Nije poželjno osloniti se na implicitnu proveru ispravnosti podataka u bazi iz sledećih razloga:
  - Presretanje grešaka koje MySQL javlja uz pomoć PHP funkcija je komplikovano
  - Nepotrebno se opterećuje mreža i DBMS
  - Teško je korisniku podatke i poruke o grešci predstaviti u razumljivom obliku

# PHP f-je koje se koriste za proveru

- O nekim funkcijama je već bilo reči:

- `bool empty( mixed var )`

- `bool isset( mixed var, ..., mixed var)`

- `void var_dump ( mixed expression, ...,  
mixed expression )`

- Funkcije za rad sa regularnim izrazima

# Funkcija `empty()`



## **bool empty ( mixed var )**

- Utvrđuje da li se promenljiva može smatrati praznom
- Vraća TRUE ako se promenljiva može smatrati praznom, ili FALSE u suprotnom
- Promenljiva se smatra praznom ako nije definisana ili ima nultu vrednost. Sledeće vrednosti se smatraju praznim: "" (prazan string), **0** (0 kao integer), "**0**" (0 kao string), **NULL**, **FALSE**, **array()** (prazan niz), **var \$var;** (promenljiva tj. polje klase koja je definisana, ali joj nije dodeljena vrednost)



# Funkcija isset()



## **bool isset ( mixed var, ... ,mixed var)**

- Funkcija vraća TRUE ako promenljiva var postoji, ili FALSE u suprotnom. U varijanti više promenljivih (poziv funkcije sa više argumenata) vraća se TRUE samo ako sve promenljive postoje.
- Smatra se da promenljiva \$x ne postoji u sledeća tri slučaja:
  - Pre nego što joj je dodeljena neka vrednost
  - Ako joj je dodeljena vrednost null (\$x=null)
  - Nakon poziva funkcije unset (unset(\$x))

# Funkcija `var_dump()`



**`void var_dump ( mixed expression, ... ,  
mixed expression )`**

- Za izraze skalarnog tipa ispisuju se tip i vrednost izraza.
- Za nizove se ispisuje tip (array), broj članova niza, a zatim za svaki član niza indeks, tip i vrednost.
- Za objekte se ispisuje tip (Object), naziv klase, broj polja objekta/klase, a zatim za svako polje objekta naziv polja, tip i vrednost.
- Koristi se u procesu testiranja podataka prosleđenih skriptu, a ne za samu proveru.

# Funkcije za rad sa regularnim izrazima

- **Pronalaženje i izdvajanje vrednosti:**

```
mixed ereg[i] (string uzorak, string izvor
 [, array ®s])
```

- **Zamena delova izvorne vrednosti:**

```
string ereg[i]_replace (string pattern,
 string replacement, string string)
```

- **Razdvajanje izvorne vrednosti na elemente niza:**

```
array split[i] (string pattern, string string
 [, int limit])
```

# Funkcija `ereg()` (#1)



```
mixed ereg (string uzorak, string izvor
 [, array ®s])
```

- Proverava se da li se unutar stringa *izvor* nalazi deo koji odgovara regularnom izrazu *uzorak* (case sensitive).
- Ako se opcioni parametar *regs* ne koristi funkcija vraća vrednost 1 u slučaju da se unutar stringa *izvor* nalazi deo koji odgovara regularnom izrazu *uzorak*. U suprotnom vraća se false.

# Funkcija `ereg()` (#2)



- Parametar *regs* se koristi da se u njemu čuvaju delovi stringa *izvor* koji odgovaraju delovima regularnog izraza *uzorak* (koji se u tom slučaju moraju nalaziti unutar malih zagrada).
- `$regs[1]` će sadržati podstring parametra *izvor* koji odgovara delu regularnog izraza koji počinje od prve leve zagrade; `$regs[2]` podstringu koji počinje od druge leve zagrade, itd.  
`$regs[0]` će sadržati kopiju kompletnog stringa koji odgovara regularnom izrazu.

# Funkcija eregi()



```
mixed eregi (string uzorak, string izvor
 [, array ®s])
```

- Identična funkciji ereg() samo je case-insensitive.

# Regularni izrazi



- Pomoću regularnih izraza (regular expressions) može se utvrditi da li neki podaci tipa string zadovoljavaju određene složene šablone (datum, adresa e-pošte,...).
- Funkcije za rad sa regularnim izrazima omogućavaju čak i izdvajanje i zamenu složenijih podvrednosti unutar zadatog stringa.
- Iako funkcije za rad sa regularnim izrazima pružaju znatno više mogućnosti nego prethodno opisane funkcije, treba ih izbeći kad god je to moguće zato što troše dosta resursa (prostornih i vremenskih).

# POSIX-ova sintaksa za regularne izraze #1

- . - bilo koji znak (džokerski znak)
- \. - znak "." (tačka)
- \\ - znak "\" (obrnuta kosa crta)
- Da biste izbegli zabunu kada kao regularan izraz zadajete tačno određeni string bolje je da koristite jednostruke navodnike
- [ ] - bilo koji od znakova zadatih unutar liste znakova navedenih između [ i ]; unutar liste se navode pojedinačni znaci i/ili opsezi znakova



# POSIX-ova sintaksa za regularne izraze #2

- [ ^ ] - nijedan od znakova zadatih unutar liste znakova navedenih između [ ^ i ]; unutar liste se navode pojedinačni znaci i/ili opsezi znakova
- Ako želite da se znak ^ tumači kao običan znak postavite ga na bilo koje mesto osim na prvo unutar srednjih zagrada.
- Ako želite da se znak - (znak za opseg) tumači kao običan znak stavite ga na početak ili na kraj liste unutar srednjih zagrada.

# POSIX-ova sintaksa za regularne izraze #3

- Pomoću sidra u regularnom izrazu može se zadati da uzorak treba da se nalazi na početku ili na kraju vrednosti koja se ispisuje.
- Znak `^` sidri uzorak na početak, a znak `$` na kraj znakovne vrednosti.
- Obe vrste sidra se koriste u regularnom izrazu u kome se zahteva potpuno poklapanje.

# POSIX-ova sintaksa za regularne izraze #4

- Znak ? znači 0 ili 1 pojavljivanje
- Znak \* znači 0,1 ili više pojavljivanja
- Znak + znači 1 ili više pojavljivanja
- Fiksni broj pojavljivanja tražene podvrednosti može se zadati između vitičastih zagrada, u obliku {min\_broj}
- Pomoću sintakse sa vitičastim zgradama mogu se zadati minimalan i maksimalan broj pojavljivanja, u obliku {min\_broj,max\_broj}

# POSIX-ova sintaksa za regularne izraze #5

- Znaci `?`, `*`, `+`, `{ }` se mogu pojavljivati iza znaka, liste znakova ili podizraza regularnog izraza (razdvojenog zagradama) i u tom slučaju označava određeni broj ponavljanja znaka, liste znakova, podizraza regularnog izraza .

# POSIX-ova sintaksa za regularne izraze #6

- Podizrazi u regularnom izrazu se uokviruju malim zagradama.
- Mogućnost grupisanja izraza u podizraze omogućava definisanje složenih regularnih izraza i izdvajanje vrednosti koje odgovaraju podizrazima u regularnom izrazu u niz.
- Alternativni uzorci se razdvajaju pomoću operatora | koji ima najniži prioritet od svih operatora.

# POSIX-ova sintaksa za regularne izraze #7

- Za predstavljanje specijalnih znakova koristi se sekvenca obrnuta kosa crta - specijalni znak.
- Drugi način da se predstavi specijalan znak je da se on stavi u listu znakova ako tamo nema specijalno značenje.
- U regularnim izrazima mogu se upotrebljavati metaznaci (`\t` - tabulator, `\n` - novi red, `\d` - bilo koja cifra, `\s` - bilo koja belina).



## *Pitanja?*

Sve ispravke, predloge i komentare uputiti na mejl:  
[drazen.draskovic@etf.bg.ac.rs](mailto:drazen.draskovic@etf.bg.ac.rs)

Hvala!