

# Lekcija 06

## PHP I BAZE PODATAKA

*dr Miroslava Raspopović, Andrej Stanišev, Jovana Kovač*



# PHP I BAZE PODATAKA

Uvod

01

02

03

04

Uvod

PHP i baze podataka

PHP i SQL Injection

PHP šabloni

PHP aplikacije

☐ Sadržaj

- ODBC
- Kreiranje baze podataka
- Forma za unos podataka
- Funkcije za čitanje iz baze podataka

1. PDO
2. SQL Injection

1. Stanja
2. Master/Detail navigacija
3. Cookies
4. Sesije

☐ Primeri PHP aplikacija

# UVOD

*Cilj ovog predavanja je da objasni osnovne funkcionalnosti rada PHP jezika, u kombinaciji sa HTML-om i CSS-om koji su bili obrađivani u prethodnim lekcijama*

U okviru ove lekce biće reči o sledećim temama:

ØPHP i baze podataka

ØPHP i SQL Injection

ØPHP šabloni

ØPHP aplikacije

# Računarski sistem

- 
- *ODBC*
  - *Kreiranje baze podataka*
  - *Forma za unos podataka*
  - *Funkcije za čitanje iz baze podataka*

01

# KORIŠĆENJE BAZA PODATAKA IZ PHP JEZIKA

## *Integracija baza podataka i sajtova predstavlja veoma važan segment u razvoju Web interfejsa*

Razvoj WWW omogućio je pristup velikom broju informacija. Međutim, često se dešava da suština informacije ostane u senci grafičkog okruženja. U želji da privuku što veći broj posetilaca, programeri koristeći se snažnim alatima, razvijaju vizuelno atraktivne Web stranice često uskraćujući posetiocima važne informacije. Snaga WWW je upravo u mogućnosti prikazivanja sadržaja koji na taj način postaju dostupni velikom broju ljudi.

Integracija baza podataka i sajtova predstavlja veoma važan segment u razvoju Web interfejsa, koji korisnicima omogućava da na lak i efikasan način dođu do željenih informacija. Primeri korišćenja baza podataka na Web stranicama su brojni. Na primer, sajt sa velikim brojem informacija teško je ažurirati. Poseban problem predstavlja promena ukupnog dizajna sajta. Ukoliko bi se čitav sadržaj smestio u bazu, za promenu čitavog dizajna potrebno je promeniti nekoliko stranica.

U marketinškom pogledu korišćenje baza je značajno, jer omogućava da se različitim posetiocima prikažu različiti baneri. Otvaranjem strane, iz baze se bira baner koji se prikazuje. Moguće je jednostavno praćenje broja pojavljivanja nekog sadržaja i njegova jednostavna promena. Dobar primer korišćenja baza na WWW predstavljaju i forumi. Ovaj pristup mnogo je efikasniji i pruža veliki broj mogućnosti.

Veliki značaj za korišćenje baza imaju skript jezici poput PHP, Perl, Python i sl.

Koristeći ove jezike u kombinaciji sa nekim od sistema za upravljanje bazama kakav je npr. MySQL, moguće je na jednostavan način integrisati bazu podataka sa Web sajtom.

# ODBC

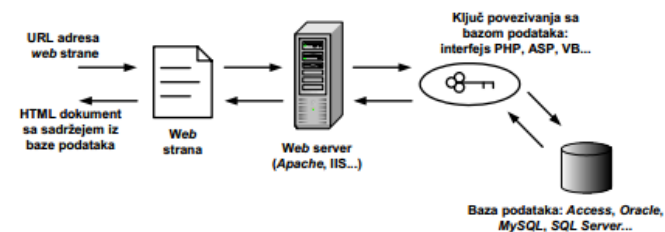
## *ODBC je široko rasprostranjen API interfejs za pristup bazama podataka*

ODBC (Open Database Connectivity) je standard koji je razvio Microsoft. Svaki komercijalni sistem za upravljanje bazama podataka i većina sistema koji nisu komercijalni imaju ODBC drajvere, koji omogućavaju pristup tim sistemima. To znači da se ne moraju koristiti drajveri tih sistema. ODBC je važan jer omogućava pristup bazi podataka koja se nalazi u Windows okruženju sa UNIX ili Linux sistema.

ODBC je široko rasprostranjen API interfejs (Application Programming Interface) za pristup bazama podataka. Ovaj standard omogućio je korišćenje u distribuiranim okruženju sistema kao što su MS Access, Oracle, SQL Server itd. ODBC omogućava apstraktni sloj koji krije specifičnosti pristupa određenoj bazi podataka i dozvoljava programerima razvoj bez potrebe da ulaze u detalje funkcionisanja baze. Na slici 1 prikazan je redosled pristupa bazi podataka korišćenjem web strana.

Ukratko, proces se odvija na sledeći način:

1. u adresnu liniju čitača unosi se URL ili se klikne na odgovarajuću hipervezu,
2. serveru se šalje zahtev za otvaranje odgovarajućeg dokumenta,
3. server sve datoteke sa ekstenzijom .php šalje posebnom interfejsu na obradu,
4. PHP čita stranu i obrađuje sve naredbe, pa i one koje se odnose na pristup ili promenu podataka u bazi, unoseći rezultate rada u HTML dokument, i
5. PHP pretvara sve naredbe i rezultate u HTML kod i vraća zahtev serveru, koji zatim šalje željenu stranu.

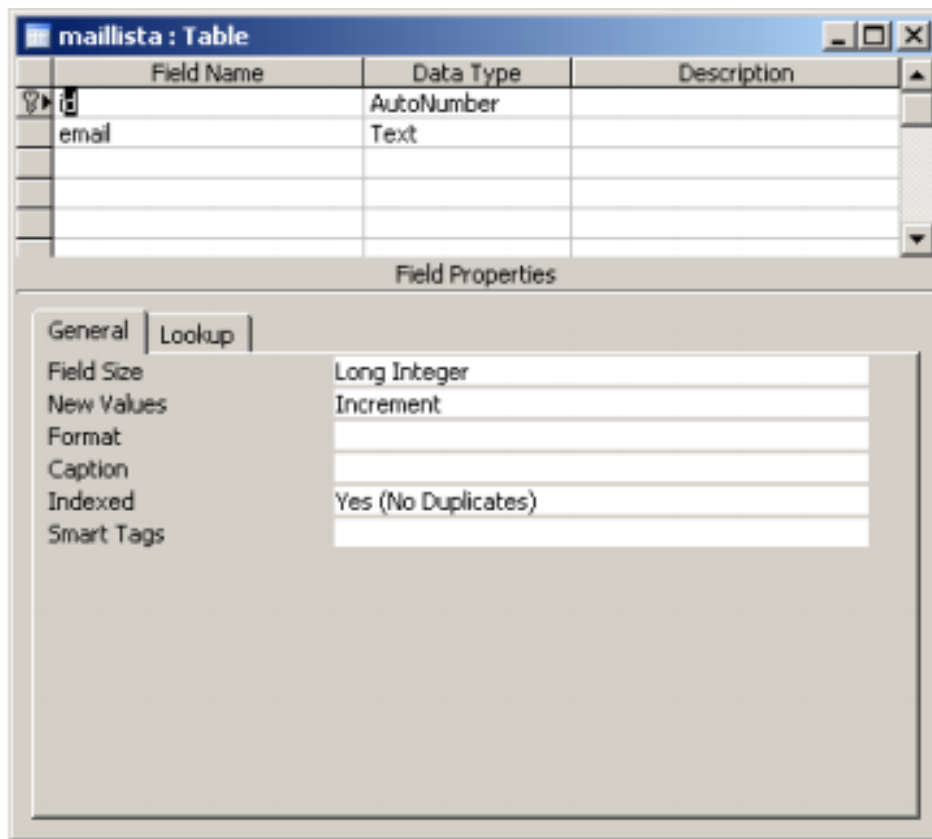


Slika-1 Proces pristupa bazi podataka

# KREIRANJE TABELE

*Potrebno je kreirati bazu podataka i tabelu*

Na početku ovog primera potrebno je, koristeći MS Access, kreirati bazu lista sa traženim podacima



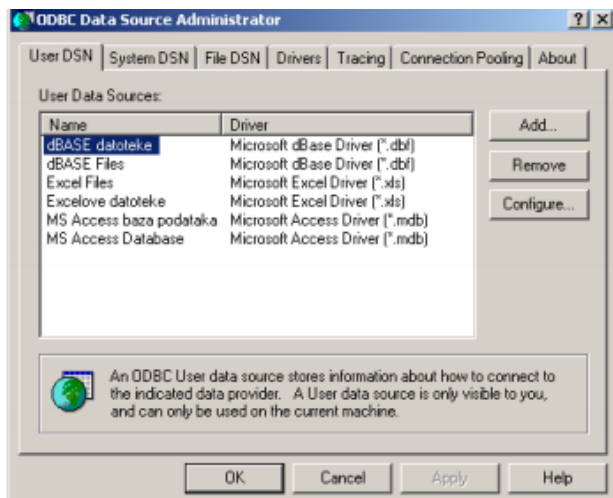
Slika-2 Kreiranje tabele

# KREIRANJE IZVORA PODATAKA

*Da bi se baza bila dostupna za aplikaciju, potrebno je kreirati ODBC datasource*

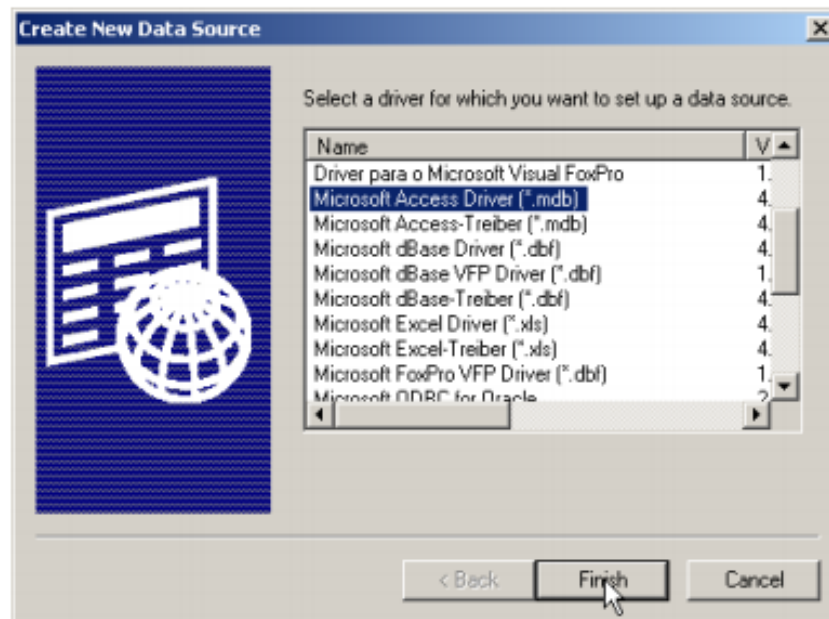
Baza se snima u direktorijum C:\phpdev\private. Sledeći korak predstavlja kreiranje ODBC izvora podataka. U Windows okruženju do aplikacije koja omogućava dodavanje, uklanjanje i konfiguraciju ODBC izvora podataka i drajvera dolazi se na sledeći način:

Start→Control Panel→Administrative Tools→Data Sources



Slika-3 Podešavanje ODBC izvora podataka

Na vrhu prozora klikne se na System DSN, a potom na opciju Add. U prozoru koji će se pojaviti, bira se odgovarajući drajver. U ovom slučaju to je Microsoft Access Driver (\*.mdb). Nakon izbora klikne se na Finish



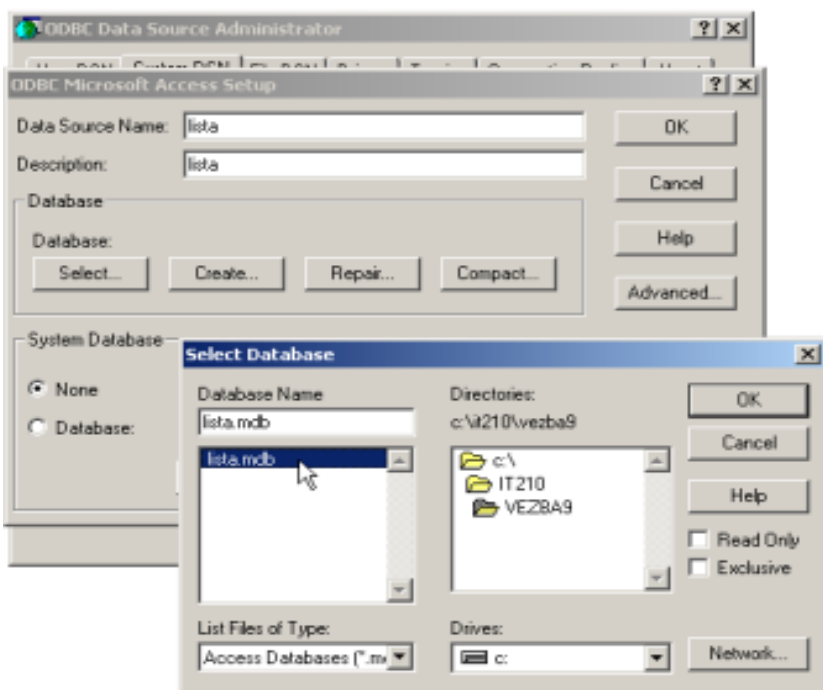
Slika-4 Odabir drivera



# IZBOR BAZE PODATAKA

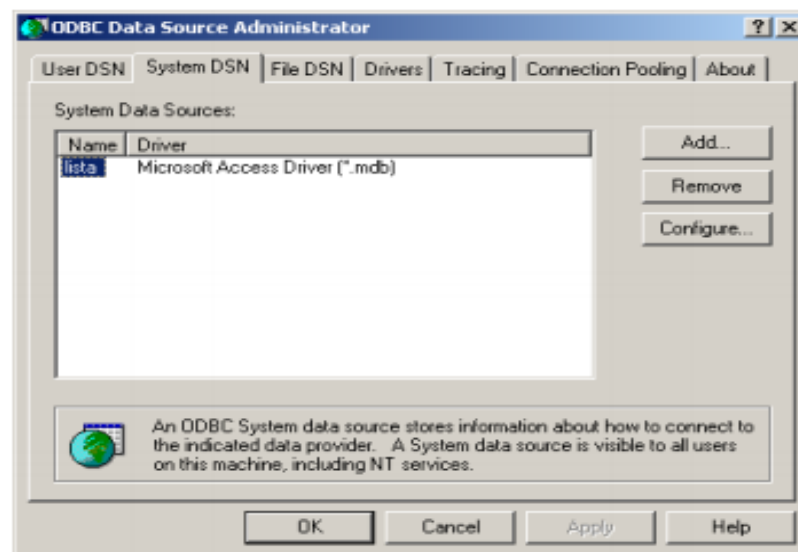
## *Potrebno je odabrati fajl baze podataka*

U sledećem prozoru u polja Data Source Name: i Description: može se uneti naziv baze – lista (slika 5).



Slika-5 Odabir fajla

Zatim se u delu prozora obeleženim sa Database klikne na opciju Select... U prozoru koji će se pojaviti, sa desne strane bira se direktorijum, a sa leve baza. Na listi System Data Sources: pojavice se i naziv lista (slika 6). Na kraju se „klikne“ na OK. U delu Advanced... moguće je izvršiti napredna podešavanja šifara, dozvola i sl. Za ovaj primer to neće biti neophodno.



Slika-6 Lista izvora podataka

# FORMA ZA UNOS PODATAKA

*Podaci uneti u tekstualno polje, nakon klika na dugme Unesi, šalju se PHP skriptu koji podatke dalje prosleđuje u bazu*

Tabele kreirane baze su prazne. Sa druge strane, HTML je statičke prirode i ne može da utiče na podatke u bazi. To je razlog korišćenja PHP skripta. U ovom slučaju HTML se koristi za kreiranje formulara kojim se podaci prosleđuju dalje. Kod HTML dokumenta može da izgleda ovako:

```
<html>
<head>
</head>
<body>
<form action="primer1.php" method="post">
Unesite e-mail: <input type="text"
name="email"><br><br>
<input type="Submit" value="Unos">
</form>
</body>
</html>
```

U ovom slučaju koristi se jedno polje za unos e-mail adrese i dugme kojim se podaci iz polja šalju. Tekstualno polje ima atribut name koji omogućava PHP skriptu da koristi vrednost koja se unese u polje. Dokument se snima na poznati način npr. kao primer1.html.

Podaci uneti u tekstualno polje, nakon klika na dugme Unesi, šalju se PHP skriptu koji podatke dalje prosleđuje u bazu. Za unos podataka u datu bazu može se koristiti sledeći PHP skript:

```
<?php
$email=$_POST['email'];
$con=odbc_connect('lista','');
$sql="INSERT INTO maillista (email) VALUES ('$email')";
$exc=odbc_exec($con,$sql);
if($exc) {print "Informacija je uspešno uneta u bazu.";}
else {print "Greska prilikom unosa.";}
odbc_close($con);
echo "<p><a href=primer1.html> Povratak </a>";
?>
```

# PRIKAZIVANJE PODATAKA IZ BAZE

*U ovom primeru se kreira HTML dokument i PHP skript koji će omogućiti prikazivanje svih adresa koje se nalaze u bazi lista iz prethodnog primera*

U ovom primeru se kreira HTML dokument i PHP skript koji će omogućiti prikazivanje svih adresa koje se nalaze u bazi lista iz prethodnog primera. Omogućiti da se klikom na prikazane adrese automatski aktivira program za slanje pošte na izabranu adresu.

U ovom primeru za aktiviranje skripta može se koristiti samo jedno dugme:

```
<html>
<head>
</head>
<body>
<form action="primer2.php" method="post">
<input type="Submit" value="Prikaži sve">
</form>
</body>
</html>
```

```
<?php
$con=odbc_connect('lista','');
$sql="SELECT email FROM maillista";
$exc=odbc_exec($con,$sql);

echo "<table border=1><tr><th>e-mail</th></tr>\n";
$nbrow=0;

while(odbc_fetch_row($exc))
{
$nbrow++;
$mail = odbc_result($exc,1);
echo "<tr><td><a href=mailto:$mail>$mail</a></td></tr>\n";
}

echo "<tr><td colspan=2>ukupno: $nbrow adresa
</td></tr></table>";

odbc_close($con);
echo "<p><a href=primer1.html> Unos nove adrese </a>";
?>
```

# FUKNCIJE ZA ČITANJE PODATAKA IZ BAZE

## *Uz pomoć funkcije `odbc_fetch_row` proverava se da li postoje vrednosti u posmatranoj vrsti tabele*

Značenje promenljivih `$con`, `$sql` i `$exc` detaljnije je objašnjeno u okviru 1. primera.

U nastavku koda se crta zaglavlje tabele u kojoj će biti prikazane adrese iz baze, a zatim definiše `$nbrow` promenljiva koja predstavlja broj vrsta tabele. Na početku je ova vrednost nula.

Uz pomoć funkcije `odbc_fetch_row` proverava se da li postoje vrednosti u posmatranoj vrsti tabele. Ukoliko postoji, tada je rezultat ove funkcije `TRUE`, u suprotnom je `FALSE`. Kao argument pojavljuje se rezultat funkcije `odbc_exec`.

Ukoliko vrsta nije prazna, vrednostima u njoj može se pristupati uz pomoć funkcije `odbc_result`, gde se kao drugi argument pojavljuje redni broj kolone

Tabela `maillista` ima dve kolone `id` i `email`, obeležene sa 0 i 1

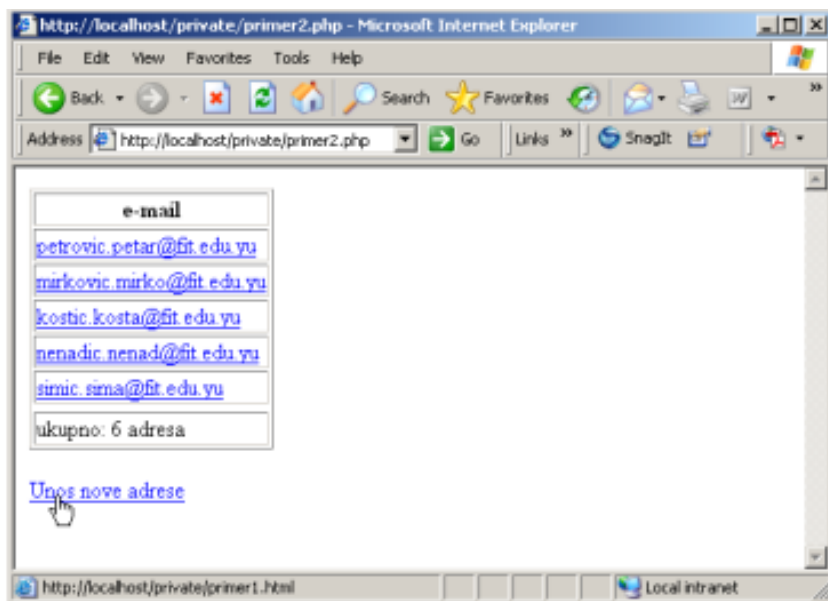
U ovom slučaju, vrednost iz druge kolone (obeležene sa 1) dodeljena je promenljivoj `$mail`. U okviru `while` petlje, `$nbrow` brojač uvećava se za jedan i prikazuje podatak iz druge kolone posmatrane vrste tabele. Nakon provere date vrste funkcija `odbc_fetch_row` prelazi na narednu vrstu. Petlja će se izvršavati sve dok funkcija `odbc_fetch_row` ne dođe do kraja tabele.

Kako je uslov zadatka da prikazane adrese predstavljaju ujedno i hiperveze koje omogućavaju slanje poruka na tu adresu, korišćeni su početni i završni `<a>` tag sa atributom `href`, o čemu je više reči bilo ranije. Na kraju se prikazuje ukupan broj vrsta, odnosno adresa, i kreira hiperveza ka HTML dokumentu za unos.

# REZULTAT

## *Skript generiše HTML stranu sa linkovima iz baze*

Dati skript snima se kao primer2.php. Za realizaciju je potrebno u čitaču otvoriti prvo primer2.html. Klikom na dugme Prikaži sve, aktivira se skript koji prikazuje sve adrese iz baze, uz mogućnost slanja poruke na adresu na koju se klikne (slika 7 i 8)



Slika-7 Rezultat izvršenja skripte

# PHP i SQL Injection

---

1. *PDO*

2. *SQL Injection*

02

# PDO

*Preferirani način za pristup bazi u modernom PHP programiranju je PDO.*

Opisani način za pristup bazi podataka je prihvatljiv u slučaju manjih aplikacija, ali ima više nedostataka. Npr. ako kompletan API za pristup bazi je ODBC specifičan. Ako bismo hteli da koristimo bazu podataka koja nema odgovarajući ODBC driver, morali bismo da koristimo drugi API.

Preferirani način za pristup bazi u modernom PHP programiranju je PDO. PDO je biblioteka koja apstrakuje pristup različitim relacionim bazama pod jedinstvenim interfejsom.

Potrebno je naglasiti da PDO ne apstrakuje drivere za bazu, niti prevodi SQL za različite vendore. Za propisno funkcionisanje PDO biblioteke, potrebno je da odgovarajući driver bude instaliran na sistemu.



Slika-1 Arhitektura PDO biblioteke

# PDO PRIMER

## *Da bi se konektovali na bazu kreiramo novi PDO objekat*

Sledi primer korišćenja PDO za pristup mysql bazi:

```
try {
    $conn = new
PDO('mysql:host=localhost;dbname=myDatabase',
$username, $password);

    $conn ->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $data = $conn->query('SELECT * FROM Student
WHERE ime = "Milan");

    foreach($data as $row) {
        echo $row['ime'];
        echo $row['prezime'];
    }
} catch(PDOException $e) {
    echo 'ERROR: ' . $e->getMessage();
}
```

Da bi se konektovali na bazu kreiramo novi PDO objekat, kao u prvoj liniji. Prvi parametar konstruktora PDO klase je connection string, koji zavisi od vrste baze podataka, i može se lako naći na internetu. Druga dva parametra su username i password za bazu.

Metoda query kreiranog objekta se koristi za izvršavanje datog SQL koda. Potom se vrši iteracija kroz niz redova. Svaki red je asocijativni niz napunjen podacima iz baze.

U slučaju greške, koristi se standardna try/catch konstrukcija za obradu izuzetka.



# SQL INJECTION

## *SQL injection je napad u kome maliciozni korisnik izvršava SQL kod*

U aplikacijama sa bazom podataka, često je potrebno konstruisati upit sa koristeći podatke koje je korisnik zadao kao ulaz. Npr. aplikacija će često imati polje za pretragu u koje korisnik može da upiše svoj upit.

Uzmimo za primer aplikaciju koja dozvoljava korisniku da pretražuje bazu studenata po imenu. Upit koji će se izvršiti u bazi će biti sličan sledećem, gde će se znak ? zameniti tekstem iz ulaza:

```
SELECT * FROM Student WHERE Student.ime = ?
```

Naivni način da se u PHP jeziku konstruiše ovakav upit bi bio sledeći:

```
$ime=$_POST['ime'];  
$con=odbc_connect('lista','');  
$sql="SELECT * FROM Student WHERE Student.ime=  
'$ime'";
```

Na izgled, ovaj kod radi bez problema, ali je izuzetno ranjiv na napade. Maliciozni korisnik bi mogao da u polje za pretragu upiše sledeći kod:

```
'a ; DROP DATABASE Studenti;
```

Rezultujući SQL upit bi bio:

```
SELECT * FROM Studenti WHERE Student.ime='a';DROP  
DATABASE Studenti;
```

Šta se tačno ovde desilo? Napadač je koristio single quote znak da zatvori navodnike koji označavaju string u where delu upita. To znači da se ostatak teksta izvršava kao SQL kod. Dobijamo sledeće upite:

```
SELECT * FROM Studenti WHERE Student.ime='a';  
DROP DATABASE Studenti;
```

# PREVENCIJA SQL INJECTION NAPADA

*Svi podaci koji dolaze od krajnjeg korisnika su potencijalno maliciozni.*

Prvi upit verovatno neće selektovati ništa, dok će drugi upit izbrisati kompletnu bazu studenata. Ovakav propust u aplikaciji bi imao katastrofalne posledice.

SQL Injection napadi dolaze u mnogo oblika, i obično nisu očigledni kao u ovom primeru. Parametri za upit mogu doći iz forme, URL parametara kroz cookie, itd. Svi podaci koji dolaze od krajnjeg korisnika su potencijalno maliciozni.

Rešenje je da se nedozvoljeni karakteri filtriraju iz ulaza. Postoje funkcije koje rade upravo ovo kao npr. [mysql\\_real\\_escape\\_string\(\)](#), ali ovo ipak stavlja preveliko opterećenje na programera.

Najbolje rešenje je korišćenje tzv. pripremljenih izraza (Prepared Statement).

# PREPARED STATEMENT

*Prepared Statement je dio SQL koda koji je prethodno pripremljen i parametrizovan.*

Prepared Statement je dio SQL koda koji je prethodno pripremljen i parametrizovan.

```
try {
    $conn = new
PDO('mysql:host=localhost;dbname=myDatabase',
$username, $password);

    $dbh->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    $stmt = $conn->prepare('SELECT * FROM Student
WHERE ime = :name');

    $stmt->bindParam(":name", "Milan");
    $data = $stmt->execute();

    foreach($data as $row) {
        echo $row['ime'];
        echo $row['prezime'];
    }
} catch(PDOException $e) {
    echo 'ERROR: ' . $e->getMessage();
}
```

Koristi se metoda PDO klase `prepare` da se kreira novi SQL izraz i dodeli varijabli `stmt`. U SQL stringu, vrednost parametra je označena placeholderom `:name`. Ovo znači da je ovaj dio upita promenljiv.

Metoda objekta `statement` `bindParam` postavlja konkretnu vrednost parametra. Prvi argument metode je ime parametra, a drugi vrednost.

Moguće je koristiti i anonimne parametre, kod kojih je placeholder znak `?`, ali je onda potrebno povezivati parametre po rednom broju. Prvi način je znatno čitljiviji.

# TRANSAKCIJE

*Glavna odlika transakcija je da će se transakcija izvršiti samo ako su se uspešno izvršile sve naredbe u transakciji*

Transakcije su način da se više upita izvrši kao grupa. Glavna odlika transakcija je da će se transakcija izvršiti samo ako su se uspešno izvršile sve naredbe u transakciji. Ovo je izuzetno korisno.

Uzmimo primer transfera novca sa računa u banci. U tom slučaju imamo dva SQL upita: prvi će da umanji vrednost na prvom računu, a drugi će da poveća vrenost na nekom drugom računu.

U slučaju da ovu operaciju izvršavamo bez transakcija, u slučaju bilo kakve greške, može se desiti da novac jednostavno nestane.

```
try {
    $dbh->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    $dbh->beginTransaction();
    $dbh->exec("insert into staff (id, first, last) values (23, 'Joe',
'Bloggs')");
    $dbh->exec("insert into salarychange (id, amount, changedate)
    values (23, 50000, NOW())");
    $dbh->commit();
} catch (Exception $e) {
    $dbh->rollBack();
    echo "Failed: " . $e->getMessage();
}
```

Transakcije počinju metodom `beginTransaction()`, a završavaju se metodom `commit()`. Sve SQL naredbe između poziva ove dve metode su dio jedne transakcije.

Metoda `rollback` će da otkáže kompletnu transakciju.

# PDO DODATNA PODEŠAVANJA

*SetAttribute metoda sa parametrom ATTR\_ERRMODE podešava ponašanje biblioteke prilikom grešaka*

```
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

SetAttribute metoda sa parametrom ATTR\_ERRMODE podešava ponašanje biblioteke prilikom grešaka. Generalno se preporučuje mod sa izuzecima.

# PHP šabloni

- 
1. *Stanja*
  2. *Master/Detail navigacija*
  3. *Cookies*
  4. *Sesije*

03

# STANJA

## Osnovni način čuvanja stanja je kroz GET query parametre

Jedna od osnovnih karakteristika HTTP protokola je to da je protokol „stateless“. Ovo znači da protokol ne čuva stanje između poruka, tj. svaka poruka se tretira kao posebna.

Ova karakteristika ima prednosti i mana. Osnovna prednost je da se stateless arhitektura može lako skalirati na više servera, a mana je dodatna kompleksnost u programiranju.

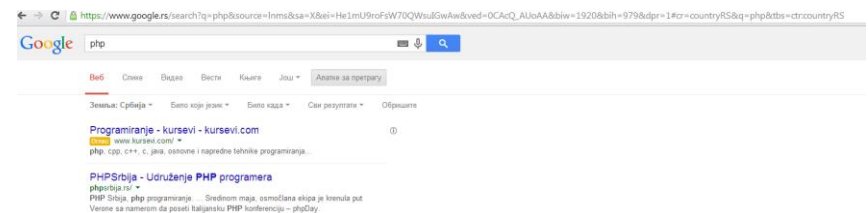
Generalno, životni ciklus PHP skripte se završi u deliću sekunde. Posle izvršenja skripte gube se vrednosti svih varijabli.

Medjutim, često je potrebno da se pamti stanje u interakcijama sa korisnikom. Nekada je se ovo stanje čuva tokom kompletne interakcije, a nekada je samo čuva preko nekoliko stranica (**conversational state**), npr. u wizardima.

Osnovni način čuvanja stanja je kroz GET query parametre. Svaka dodatna opcija koju je korisnik izabrao se dodaje kao poseban query parametar. Ovo se često koristi kada je potrebno implementirati pretragu sa više parametara.

Prednost ove metode je što generiše URL kakav je moguće kopirati ili bookmarkovati.

Dobar primer ove metode je Google pretraga. U URL baru na slici 1 vidi se veliki broj parametara. Kada je npr. izabrana zemlja, dodan je parametar country=RS.



Slika-1 Google pretraga

# MASTER/DETAIL NAVIGACIJA

*Najlakši način da se postigne ovaj stil navigacije je da svaka stavka iz liste ima jedinstven ID u bazi*

Čest slučaj u web programiranju je zahtev da se korisniku prikaže lista sa više elemenata, a da se klikom na neki element liste, na drugoj stranici prikažu detalji tog elementa

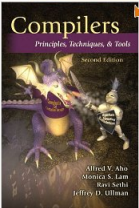
More Items to Consider

You viewed	Customers who viewed this also viewed		
 <p><b>Compilers: Principles, Techniques, and Tools (2nd Edition)</b> Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman Hardcover ★★★★☆ (83) \$179.99 \$137.75</p>	 <p><b>Compilers: Principles, Techniques, and Tools (2nd Edition)</b> Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman Hardcover ★★★★☆ (83)</p>	 <p><b>Engineering a Compiler, Second Edition</b> Keith Cooper, Linda Torczon Hardcover ★★★★☆ (9) \$89.95 \$78.22</p>	 <p><b>Programming Language Pragmatics</b> Michael Lee Scott Paperback ★★★★☆ (18) \$77.95 \$63.98</p>

Slika-2 Master strana

Browse Programming Books C | Java | PHP | Python

click to LOOK INSIDE!



**Compilers: Principles, Techniques, and Tools (2nd Edition) [Hardcover]**  
by Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman  
★★★★☆ (83 customer reviews)

**Buy New**  
\$137.75 & FREE Shipping. [Details](#)

**Rent**  
\$43.49 & FREE Shipping. [Details](#)

**Only 20 left in stock (more on the way).**  
Ships from and sold by Amazon.com. Gift-wrap available.

**In Stock.**  
Rented by RentU and Fulfilled by Amazon.

Want it Tuesday, May 6? Order within 20 hrs 42 mins and choose One-Day Shipping at checkout. [Details](#)

28 new from \$113.95 48 used from \$88.95

**FREE TWO-DAY SHIPPING FOR COLLEGE STUDENTS**  
[Learn more](#)

Slika-3 Detail strana

Na slikama 2 i 3 prikazana je Master Detail navigacija sa sajta amazon.com.

Najlakši način da se postigne ovaj stil navigacije je da svaka stavka iz liste ima jedinstven ID u bazi. Taj ID se onda postavlja kao query parametar u svim elementima liste. Kada korisnik klikne na jedan element iz liste, preusmerava se na drugu stranu, koja čita ID iz query parametra, nalazi red sa tim identifikatorom u bazi i prikazuje ga.



# PRIMER MASTER/DETAIL NAVIGACIJE

## *Prikaz koda za jednostavnu Master/Detail navigaciju*

Sledeći delovi koda prikazuju jednostavnu Master/Detail navigaciju.

```
$data = $conn->query('SELECT * FROM Student');

echo "<ul>";

foreach($data as $row) {
    echo "<li><a href=detail.php?id=".$row['id'] .
">$row['ime']</a></li>";
}

echo "</ul>";
```

Generiše se lista sa linkovima. Svaki link ima query parametar id postavljen na vrednost identifikatora datog reda.

```
stmt = $conn->prepare('SELECT * FROM Student WHERE id =
:id');

$stmt->bindParam(":id", $_GET['id']);

$data = $stmt->execute();
```

Na Detail strani uzima se ID iz parametra, i pomoću njega izvršava upit.

# COOKIES

*Cookie se može koristiti da čuva podatke koji treba da budu perzistenti izmedju zahteva*

HTTP Cookie je mali skup podataka koji HTTP server šalje klijentu i klijent čuva. Prilikom svakog zahteva, klijent cookie šalje serveru kao dio zahteva. Cookie se može koristiti da čuva podatke koji treba da budu perzistenti izmedju zahteva.

Za postavljanje vrednosti koristi se funkcija `set_cookie`

```
setcookie("user", "Alex", time()+3600);
```

Vrednost se čita iz posebnog niza `$_COOKIE`

```
if (isset($_COOKIE["user"]))  
echo "Welcome " . $_COOKIE["user"] . "!\n";  
else
```

# SESIJE

*Kod sesije cookie sadrži samo automatski generisan identifikator sesije, dok se ostatak podataka vezanih za identifikator čuva na serveru, u memoriji, ili u nekoj brzjoj bazi podataka*

Glavni nedostatak prethodnog pristupa je što se cookie šalje prilikom svakog zahteva, što je pogodno za samo malu količinu podataka. Takođe, cookie se nalazi na klijentskoj strani, što znači da ga maliciozni korisnik može lako promeniti.

Rešenje za ove probleme su sesije. Kod sesije cookie sadrži samo automatski generisan identifikator sesije, dok se ostatak podataka vezanih za identifikator čuva na serveru, u memoriji, ili u nekoj brzjoj bazi podataka.

Sesije su glavni mehanizam za čuvanje podataka trenutno ulogovanog korisnika, pošto će svaki korisnik imati svoju posebnu sesiju.

Za početak rada sa sesijom koristi se funkcija `session_start()`. Posle poziva funkcije može se pristupati članovima asocijativnog niza `$_SESSION`, koji se automatski popunjava.

```
<?php
session_start();
if (!isset($_SESSION['count'])) {
    $_SESSION['count'] = 0;
} else {
    $_SESSION['count']++;
}
?>
```

# REDIRECT AFTER POST

*Pošto POST zahtev u većini slučajeva ima posledice, pojavljuje se problem u slučaju da korisnik slučajno pošalje jedan zahtev više puta*

Po HTTP standardu, POST i PUT zahtevi nisu idempotentan, što u osnovni znači da zahtevi ovog tipa mogu prouzrokovati „nuspojave“ (side effects). GET zahtev je idempotentan.

Šta će server uraditi kao odgovor na bilo koji od ovih zahteva zavisi od programera, ali važno je poštovati pravilo da GET zahtev nikada ne bi trebalo da pravi bilo kakve izmene na serveru, tj. u bazi podataka.

Pošto POST zahtev u većini slučajeva ima posledice, pojavljuje se problem u slučaju da korisnik slučajno pošalje jedan zahtev više puta, npr. korišćenjem refresh dugmeta.

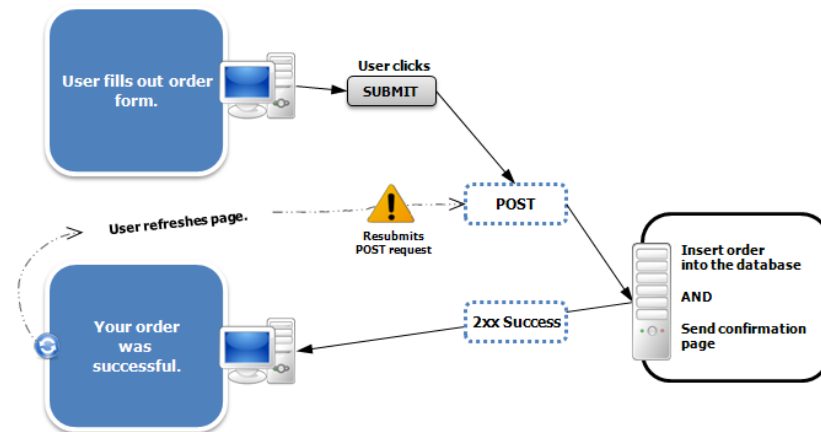
U tom slučaju, browser će prikazati poruku upozorenja.

Rešenje se posle POST zaheva vrati redirect, koji će naterati browser da pošalje GET zahtev za sledeću stranu.

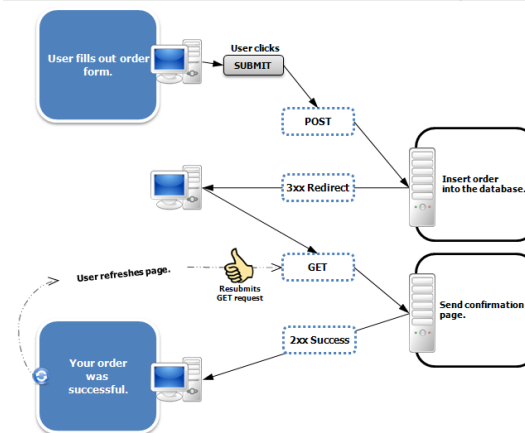
Redirect se može poslati na sledeći način:

```
header( 'Location: http://www.yoursite.com/new_page.html' )
```

```
;
```



Slika-4 POST bez preusmerenja



Slika-5 POST sa preusmerenjem

# ORGANIZACIJA KODA

*Najjednostavniji način je da se kod logike aplikacije, i kod za pristup bazi odvoji u posebne fajlove, sa funkcijama i klasama koje mogu da se koriste na više mesta*

Kako codebase aplikacije raste, tako postaje jasno da je potrebno organizovati kod aplikacije na neki način. Veliki fajlovi koji u sebi sadrže PHP, SQL i HTML nisu čitljivi.

Postoji mnogo PHP biblioteka (framework) za organizaciju koda, ali za početak je dovoljno izdvojiti što više HTML koda od logike aplikacije, i pristupa bazi.

Naime, potrebno je voditi računa da kod HTML stranica sadrži samo onoliko PHP koda koliko je potrebno da se dinamički renderuje sadržaj strane, ali ne i logiku aplikacije.

Najjednostavniji način je da se kod logike aplikacije, i kod za pristup bazi odvoji u posebne fajlove, sa funkcijama i klasama koje mogu da se koriste na više mesta.

Takvi fajlovi se mogu uključiti u trenutni fajl sledećim funkcijama: [include](#), [require](#) i [require\\_once](#).

# PHP aplikacije

---

➤ *Primeri PHP aplikacija*

04

# PRIMER APLIKACIJE : IGRA

*Kreirati PHP aplikaciju koja sa korisnikom igra igru makaze, papir , kamen.*

## Zadatak

Kreirati PHP aplikaciju koja sa korisnikom igra igru makaze, papir , kamen. Potrebno je kreirati formu u kojoj će korisnik izabrati jednu od opcija, dok server za svoj izbor koristi random. Kreirati stranu na kojoj će se ispisati rezultat poteza, i trenutni skor. Za čuvanje skora koristiti sesije.

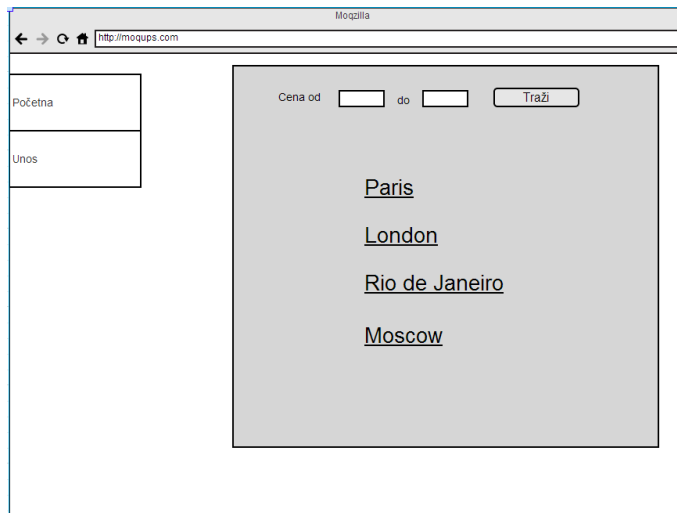
# PRIMER APLIKACIJE : TURISTIČKE PONUDE

## *Potrebno je kreirati sajt za praćenje ponuda turističkih agencija*

Potrebno je kreirati sajt za praćenje ponuda turističkih agencija. Na slikama 1, 2 i 3 dat je prototip izgleda sajta koji je potrebno kreirati upotrebom HTML-a, CSS-a, PHP-a i baze podataka.

U MySQL ili MS Access kreirati bazu podataka pod nazivom "ISPIT"

U okviru baze ISPIT kreirati tabelu "PUTOVANJE" koja sadrži kolone: ID, odrediste, opis, cena i pansion. ID mora predstavljati ključ tabele čija se vrednost automatski generiše



Slika-1 Početna strana



# STRANA SA FORMOM

*Kreirati stranu "Form" koja sadrži formu za nove ponude putovanja.*

Kreirati stranu "Form" koja sadrži formu za nove ponude putovanja.

The image shows a web browser window with the title 'Mozilla' and the address bar containing 'http://moqups.com'. On the left side, there is a vertical navigation menu with two items: 'Početna' and 'Unos'. The main content area is a grey box titled 'Nova ponuda' containing a form with the following fields:

- 'Odredište' with a text input field.
- 'Cena' with a text input field.
- 'Pun pansion' with a checked checkbox.
- 'Opis' with a large text area.
- A 'Dodaj' button at the bottom.

Slika-2 Strana sa formom za unos

# STRANA SA DETALJIMA

*Kreirati stranu "Detail" na kojoj će se prikazati svi detalji odabranog putovanja.*

Kreirati stranu "Detail" na kojoj će se prikazati svi detalji odabranog putovanja.



Slika-3 Strana sa detaljima

# PRIMER APLIKACIJE: ŠIFROVANJE

*Kreirati aplikaciju koja će omogućiti korisniku da šifruje tekst pomoću Cezarove šifre, ili neke njene varijacije.*

Kreirati aplikaciju koja će omogućiti korisniku da šifruje tekst pomoću Cezarove šifre, ili neke njene varijacije.

Cezarova šifra je tip šifre zamjene (supstitucije), u kome se svako slovo otvorenog teksta zamjenjuje odgovarajućim slovom abecede, pomaknutim za određeni broj mesta. Na primjer, s pomakom 3, A se zamijenjuje slovom D, B slovom E itd.

# PRIMER APLIKACIJE: AUKCIJA

## *Kreirati PHP aplikaciju za online aukcije*

Kreirati PHP aplikaciju za online aukcije. Korisnici se mogu registrovati i kreirati aukcije, a drugi korisnici mogu da postavljaju ponude.

Korisnici mogu postaviti više ponuda. Po isteku vremena predviđenog za aukciju, korisnik sa najvećom ponudom pobeđuje. Obavestiti sve korisnike koji su postavili ponude o rezultatu aukcije.