**PHP i baze podataka**

**00:20:22**

Jedinica: 19 od 19

**Rezime**

* PHP obezbeđuje podršku za Open Database Connectivity (ODBC) i na taj način je moguće da PHP program pristupi bilo kojoj bazi podataka koja podržava ODBC;
* odbc\_connect(dsn/dsn-less connection string, username, password) – funkcija koja se koristi za uspostavljanje konekcije sa bazom;
* odbc\_exec(connection\_id, SQL query\_string) – ova funkcija se koristi za izvršavanje SQL naredbi;
* Ekstenzija mysql se od verzije PHP 5.5.0 smatra prevaziđenom i njena upotreba nije preporučljiva;
* Prikaz podataka iz baze se vrši uz pomoć SELECT upita.

U ovoj lekciji ćete imati prilike da naučite sa kojim sve bazama i na koji način možete raditi u PHP jeziku.

**PHP i ODBC**

PHP obezbeđuje podršku za *Open* *Database* *Connectivity* (ODBC). Na taj način je moguće da PHP program pristupi bilo kojoj bazi podataka koja podržava [ODBC](https://www.link-elearning.com/linkdl/opisPojma.php?id=139120), kao što su *Oracle*, *DB2*, *MS* *SQLServer* ili *MS* *Access*. PHP obezbeđuje metode koji mogu da pristupe bazi na osnovu njenog *DSN-a*, ali i da ostvare tzv. *DSN-less* konekcije. Ovaj drugi vid konekcija se koristi u slučajevima kada ne postoji [DSN](https://www.link-elearning.com/linkdl/opisPojma.php?id=139104) ili on nije poznat. Na primer, u slučaju *MS Access* baze.

Najčešće korišćene funkcije za ODBC konekcije su:

* + ***odbc\_connect(dsn/dsn-less connection string, username, password)*** – funkcija koja se koristi za uspostavljanje konekcije sa bazom. Ova funkcija ima tri argumenta, pri čemu prvi od njih predstavlja DSN baze ili DSN-less string, i ovaj argument je neophodan. Druga dva argumenta predstavljaju korisničko ime i šifru. Ukoliko nije potrebno njihovo prosleđivanje bazi, moguće je jednostavno proslediti prazan string. Nakon uspostavljanja konekcije, ova funkcija vraća identifikacioni broj konekcije koji koriste druge ODBC funkcije;
  + ***odbc\_exec(connection\_id, SQL query\_string)*** – ova funkcija se koristi za izvršavanje [SQL](https://www.link-elearning.com/linkdl/opisPojma.php?id=139125) naredbi. Ukoliko dođe do greške, funkcija vraća vrednost FALSE. Ukoliko se SQL naredba izvrši uspešno, funkcija vraća skup slogova koji zadovoljavaju upit;
  + ***odbc\_fetch\_array(recordset name****)* – korsti se za smeštanje skupa slogova u asocijativni niz;
  + ***odbc\_num\_rows(recordset name*)** – vraća broj slogova koji se nalaze u rezultujućem setu. Ukoliko dođe do greške, vraća vrednost -1. Za naredbe INSERT, UPDATE ili DELETE, ova funkcija vraća broj slogova izmenjenih ovim naredbama. Za naredbu SELECT ova funkcija vraća broj selektovanih slogova;
  + ***odbc\_close(connection\_id)*** – zatvara konekciju.

**PHP i MySQL**

Najčešće korišćena baza podataka na internetu je svakako MySQL. Ovaj format baze je zbog svoje brzine, velike pouzdanosti i jednostavnosti korišćenja za kratko vreme postao primaran kada su u pitanju internet aplikacije. PHP obezbeđuje podršku za MySQL baze pomoću niza klasa i funkcija koje mogu biti korišćene za manipulaciju podacima.

U PHP-u je moguće rukovati bazom podataka na više načina. Najrasprostranjeniji način je upotreba ekstenzija mysqli i PDO.

|  |
| --- |
| **Napomena** Ekstenzija mysql se od verzije PHP 5.5.0 smatra prevaziđenom i njena upotreba nije preporučljiva. Naslednik ove ekstenzije je unapređena mysqli ekstenzija čija je upotreba sasvim prihvatljiva i koja se može koristiti objektno i proceduralno. Pored ove, možete koristiti i PDO ekstenziju, koja će biti detaljnije obrađena u narednim kursevima i koja podržava samo objektni pristup. Sledećom ilustracijom bismo mogli prikazati odnos između mysqli i PDO ekstenzija:  https://www.link-elearning.com/linkdl/coursefiles/1046/COPHPP_19_01.jpg *19.1. MySQLi vs PDO*    Dakle, proceduralni pristup omogućava samo mysqli, ali zato može raditi isključivo sa MySQL-om. Sa druge strane, PDO može raditi sa većim brojem baza (12), ali zato ne nudi proceduralni pristup. Obe ekstenzije veoma dobro rade OOP i podržavaju MySQL. |

U biblioteci php mysqli kompletno rukovanje bazom može se obavljati proceduralno ili objektno.

Pregled nekih od najvažnijih fukcija mysqli ekstenzije:

* + *mysqli\_connect( MySQL server name, username, password, dbname )* – otvara konekciju ka MySQL serveru i selektuje bazu podataka,
  + *mysqli\_query( connection, sql query )* – šalje upit trenutno aktivnoj bazi,
  + *mysqli\_fetch\_array(recordset)* – vraća niz koji odgovara rezultujućem skupu slogova,
  + *mysql\_num\_rows( recordset id )* – utvrđuje broj redova sadržanih u rezultujućem skupu podataka dobijenog nakon izvršavanja SELECT naredbe,
  + *mysqli\_affected\_rows( connection )*– utvrđuje broj redova izmenjenih naredbama INSERT, DELETE ili UPDATE,
  + *mysqli\_close( connection )* – zatvara konekciju.

**Kreiranje podataka**

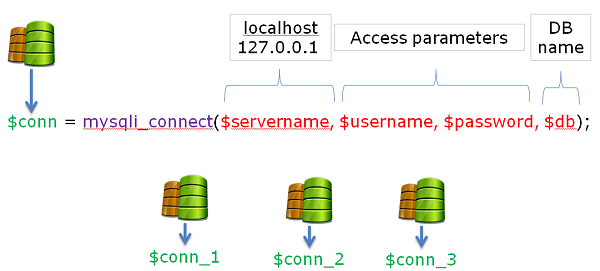
Sve izmene i pretraživanja podataka u bazi realizuju se korišćenjem odgovarajućih SQL naredbi, koje se prosleđuju bazi.

U nastavku ćemo MySQL serveru poslati dve naredbe. Jednu za kreiranje baze podataka, a drugu za kreiranje tabele. Rezultat će biti kreirana baza ili tabela sa podacima.

Prvi korak je, naravno, kreiranje konekcije. To ćemo uraditi funkcijom mysqli\_connect. Na primer:

|  |  |
| --- | --- |
| 1 | $conn = mysqli\_connect("localhost","root",""); |

Već ovde odmah treba napomenuti da je moguće istovremeno vršiti i održavati konekciju ka većem broju baza podataka. To znači da ne moraju svi podaci koji su potrebni za rad naše aplikacije biti smešteni na jednom serveru baze podataka, već bismo istovremeno mogli komunicirati sa više ovakvih servera. Grafički bi ova funkcija i povezivanje ka različitim bazama moglo biti prikazano na sledeći način:



*19.2. mysqli connect*

Drugi korak će biti izvršavanje upita. Upit treba da kreira bazu podataka, pa će cela linija glasiti:

|  |  |
| --- | --- |
| 1 | mysqli\_query($conn,"CREATE DATABASE first\_php\_test\_db"); |

Zatim ćemo pristupiti kreiranoj bazi i u njoj napraviti jednu tabelu uz pomoć SQL upita, na način veoma sličan onom iz prethodne linije:

|  |  |
| --- | --- |
| 1  2 | mysqli\_select\_db($conn,"first\_php\_test\_db");  mysqli\_query($conn,"CREATE TABLE mytable (id int primary key auto\_increment, username varchar(50))"); |

Nakon toga, takođe uz pomoć upita i funkcije mysqli\_query, ubacujemo i neke podatke u tabelu. To će biti tri imena: Peter, Sally i John.

|  |  |
| --- | --- |
| 1 | mysqli\_query($conn,"INSERT INTO mytable values(null,'Peter'),(null,'Sally'),(null,'John')"); |

Na kraju, prekinućemo konekciju sa bazom podataka funkcijom mysqli\_close:

|  |  |
| --- | --- |
| 1 | mysqli\_close($conn); |

Po izvršenju koda, na serveru će biti kreirana baza podataka. U nastavku sledi kompletan kod (ne zaboravite da nakon izvršenja komentarišete sve linije koje se tiču kreiranja baze podataka i tabela, kako se ne bi svaki put iznova izvršavale i pokušavale da kreiraju postojeće objekte u bazi):

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | $conn = mysqli\_connect("localhost","root","");  mysqli\_query($conn,"CREATE DATABASE first\_php\_test\_db");  mysqli\_select\_db($conn,"first\_php\_test\_db");  mysqli\_query($conn,"CREATE TABLE mytable (id int primary key auto\_increment, username varchar(50))");  mysqli\_query($conn,"INSERT INTO mytable values(null,'Peter'),(null,'Sally'),(null,'John')");  mysqli\_close($conn); |

**Selektovanje i izmena podataka**

Prikaz podataka iz baze se vrši uz pomoć SELECT upita. Ovim upitom navodimo koje podatke hoćemo da prikažemo i na koji način. Na primer, recimo da hoćemo da preuzmemo podatke koje smo malopre uneli. Poslaćemo MySQL serveru sledeći SQL upit:

|  |  |
| --- | --- |
| 1 | SELECT \* FROM mytable |

Ovaj upit će u tabelarnoj formi prikazati sadržaj tabele i mi ga možemo preuzeti uz pomoć php funkcija.  
  
Ovde treba naglasiti da zvezdica u upitu znači da će biti selektovane vrednosti svih kolona. Umesto zvezdice, ovde se mogu navoditi i nazivi kolona odvojeni zarezima. Na primer, da smo želeli da selektujemo samo vrednost kolone username napisali bismo:

|  |  |
| --- | --- |
| 1 | SELECT username FROM mytable |

Već znamo da je php funkcija za startovanje SQL upita mysqli\_query, pa bi linija koda mogla izgledati ovako:

|  |  |
| --- | --- |
| 1 | $result = mysqli\_query($conn,"SELECT \* FROM mytable"); |

Ova će linija vratiti rezultat upita i smestiti ga u promenljivu $result. Ova promenljiva će nakon toga sadržati resurs kroz koji možemo proći na različite načine. Jedan od najčešće korišćenih je upotreba fetch funkcija. Fetch funkcija vraća aktuelni red i pozicionira se na sledeći, pri čemu red može biti vraćen u obliku indeksiranog niza, asocijativnog niza ili objekta. Moguće je čak vratiti rezultat u obliku kombinovanog asocijativnog i indeksiranog niza. Ukoliko nema više redova, ova funkcija će vratiti null, što se može protumačiti i kao false, na osnovu čega bismo mogli da konstruišemo sledeći kod koji će prikazati sve korisnike iz tabele:

|  |  |
| --- | --- |
| 1  2 | while($rw=mysqli\_fetch\_row($result))      echo "ID: " . $rw[0] . " - Name: " . $rw[1] . "<br />"; |

Rezultat na izlazu treba da bude sledeći:

ID: 1 - Name: Peter  
ID: 2 - Name: Sally  
ID: 3 - Name: John

U praksi, upotreba baze podataka će biti znatno kompleksnija i uključivaće svakako i upotrebu podataka iz formi. Iako će u pozadini i dalje kompletan koncept počivati na gorepomenutim pravilima.

Primer koji sledi predstavlja stranu neke kompanije na kojoj se nalazi forma za pretraživanje podataka o zaposlenima:

Napomena: Primer sadrži više delova neobrađenih u gradivu i zahteva postojanje odgovarajuće baze podataka, pa ga nije potrebno raditi, već samo analizirati njegovu strukturu.

**index.php**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | <?php  if (isset($\_POST['searchName']) && !empty($\_POST['searchName'])) {    $string = $\_POST[ 'searchName' ];    $conn = mysqli\_connect('localhost', 'root', '');    $db = mysqli\_select\_db($conn, 'membership');    $sql = "SELECT \* FROM directory WHERE LName = '$string'";    $rs = mysqli\_query($conn, $sql);        if (mysqli\_num\_rows($rs) == 0){          echo "No records found!";      }else{          while($row = mysqli\_fetch\_array($rs)) {              echo "Ime: " .$row['FName'] . "  ";              echo  $row['LName'] . "<br/>";              echo "Tel: " . $row['Tel'] . "<br/>";              echo "E-mail: " . $row['email'] . "<br/>";          }      }      mysqli\_close($conn);  }  ?>    <!DOCTYPE html>  <head>  <title> A Web Page </title>  </head>  <body>  <form action = "index.php" method = "post">      <p>Enter the last name of the employee and click the "Search"</p>  <table>      <tr>          <td colspan = "2"> Company XYZ Directory </td>      </tr>          <td><input type = "text" size = "15" name = "searchName"/></td>          <td><input type = "submit" value = "Search"/>      </tr>  </table>  </form>  </body>  </html> |

Nakon unosa imena u input polje i klika na dugme „Search”, vrši se pretraga podataka iz baze. Ukoliko rezultat postoji, ispisuju se podaci o zaposlenom. Sa druge strane, ukoliko nema poklapanja korisničkog unosa sa podacima u bazi, ispisuje se poruka: „No records found!”.

Analilzirajući ovaj kod, uoćavamo da se on može podeliti na dva velika bloka: PHP blok i HTML blok.

Razmotrimo najpre HTML blok:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | <!DOCTYPE html>  <head>  <title> A Web Page </title>  </head>  <body>  <form action = "index.php" method = "POST">      <p>Enter the last name of the employee and click the "Search"</p>  <table>      <tr>          <td colspan = "2"> Company XYZ Directory </td>      </tr>          <td><input type = "text" size = "15" name = "searchName"/></td>          <td><input type = "submit" value = "Search"/>      </tr>  </table>  </form>  </body>  </html> |

Uočavamo da se radi o jednostavnom HTML kodu, te njegove delove ne treba posebno razmatrati, ali ćemo ipak prokomentarisati samu formu. Forma će se poslužiti POST metodom kako bi prosledila parametre ka serveru. Action atributom je definisan poziv index.php fajla (odnosno fajla koji se trenutno koristi). Input polje u koje je potrebno da korisnik izvrši unos ima vrednost atributa name: „searchName”, na osnovu kojeg ćemo ga kasnije identifikovati kroz PHP kod.

Sada možemo pristupiti analizi PHP dela koda:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | <?php  if (isset($\_POST['searchName']) && !empty($\_POST['searchName'])) {    $string = $\_POST[ 'searchName' ];    $conn = mysqli\_connect('localhost', 'root', '');    $db = mysqli\_select\_db($conn, 'membership');    $sql = "SELECT \* FROM directory WHERE LName = '$string'";    $rs = mysqli\_query($conn, $sql);        if (mysqli\_num\_rows($rs) == 0){          echo "No records found!";      }else{          while($row = mysqli\_fetch\_array($rs)) {              echo "Ime: " .$row['FName'] . "  ";              echo  $row['LName'] . "<br/>";              echo "Tel: " . $row['Tel'] . "<br/>";              echo "E-mail: " . $row['email'] . "<br/>";          }      }      mysqli\_close($conn);  }  ?> |

Prilikom prvog otvaranja strane index.php, izvršenje PHP koda nije potrebno, jer korisnik nije stigao da izvrši nikakav unos u input polje. Zato ovde treba postaviti uslovno izvršavanje i PHP kod dalje izvršavati samo ako je POST parametar „searchName” prosleđen i ako se ne radi o praznom parametru (što može biti posledica klika na dugme „Search” bez unosa u input polje). Ovakvu proveri vršimo pomoću sledeće linije koda:

if (isset($\_POST['searchName']) && !empty($\_POST['searchName'])) {

U promenljivu $string postavljamo vrednost prosleđenog parametra.

Sada možemo izvršiti konekciju ka bazi podataka na sledeći način:

$conn = mysqli\_connect('localhost', 'root', '');

Nakon konekcije vršimo selektovanje željene baze podataka:

$db = mysqli\_select\_db($conn, 'membership');

Kako je sada naša konekcija spremna, možemo pripremiti i SQL upit:

$sql = "SELECT \* FROM Directory WHERE LName = '$string'";

Za kreiranje upita smo koristili SQL sintaksu i pripremljenu promenljivu $string.

|  |
| --- |
| **Napomena**  Ovaj primer urađen je za potrebe vežbe. U produkciji bi ovakav upit predstavljao visok bezbednosni rizik i trebalo bi se poslužiti nekim bezbednosnim konceptima. |

Kako je sada sve spremno za upućivanje upita ka bazi, možemo to i uraditi na sledeći način:

$rs = mysqli\_query($conn, $sql);

Sada je potrebno proveriti da li ima rezultata pretrage, i u skladu sa tim usmeriti našu aplikaciju na sledeće korake izvršavanja:

if (mysqli\_num\_rows($rs) == 0){

Ukoliko nema rezultata pretrage, potrebno je korisnika obavestiti o tome:

echo "No records found!";

Sa druge strane, ukoliko rezultata pretrage ipak ima, potrebno je proći kroz svaki od njih i prikazati ih na strani:

while($row = mysqli\_fetch\_array($rs)) {  
                            echo "Ime: " .$row['FName'] . "  ";  
                            echo  $row['LName'] . "<br/>";  
                            echo "Tel: " . $row['Tel'] . "<br/>";  
                            echo "E-mail: " . $row['email'] . "<br/>";  
                     }

Na kraju čitavog procesa, možemo zatvoriti konekciju:

mysqli\_close($conn);

Na sličan način se vrši i brisanje, dodavanje ili izmena nekog reda, s tim što se u tom slučaju koriste odgovarajuće SQL naredbe: *DELETE*, *INSERT* i *UPDATE*.

**Vrste i nivoi ovlašćenja u radu sa MySQL-om**

Iako u ovom kursu razmatramo samo osnove upotrebe kombinacije PHP-a i MySQL-a, potrebno je znati šta smete da koristite i u kojoj meri da ne biste došli u poziciju da vas greška postavi na pogrešan put do rešenja. Preporučljivo je da znate ovlašćenja prilikom korišćenja baze podataka da biste znali kako da pravilno upotrebite funkcije. U MySQL-u postoje tri osnovne vrste korisnika:

* + Obični korisnici,
  + Administratori,
  + I nekoliko specijalnih dozvola.

Sada ćemo pogledati samo ovlašćenja koja su dodeljena običnim korisnicima:

|  |  |  |
| --- | --- | --- |
| ***Ovlašćenje*** | ***Dodeljuje se za*** | ***Opis*** |
| SELECT | Tabele, kolone | Dozvoljava korisnicima da čitaju redove u tabelama. |
| INSERT | Tabele, kolone | Dozvoljava korisnicima da unose nove redove. |
| UPDATE | Tabele, kolone | Dozvoljava korisnicima da menjaju vrednosti u postojećim redovima u tabeli. |
| DELETE | Tabele | Dozvoljava korisnicima da brišu postojeće redove. |
| TRUNCATE | Tabele | Dozvoljava korisnicima da obrišu sve redove. |
| INDEX | Tabele | Dozvoljava korisnicima da dodeljuju indekse određenim tabelama. |
| ALTER | Tabele | Dozvoljava korisnicima da menjaju strukturu postojećih tabela kao sto je, na primer, dodavanje kolona, menjanje imena kolona ili tabela ili tipove podataka u tabeli. |
| CREATE | Baze podataka, tabele | Dozvoljava korisnicima da prave nove baze podataka i tabele. Ako je u naredbi GRANT navedena određena baza podataka ili tabela, korisnici mogu da naredbom CREATE naprave samo tu bazu ili tabelu, što znači sledeće: ako postoji, moraju prvo da je izbrišu naredbom DROP. |
| DROP | Baze podataka, tabele | Dozvoljava korisnicima da brišu baze podataka i tabele. |

*Tabela 19.1*

**Vežba 1**

***Problem:***

Uz pomoć PHP MySQL-a potrebno je kreirati bazu podataka test\_db i u njoj tabelu users, koja sadrži tri polja: id, username i password.

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <?php  $conn = mysqli\_connect( "localhost", "root", "" );  mysqli\_query( $conn, "CREATE DATABASE test\_db" );  mysqli\_select\_db( $conn, "test\_db" );  mysqli\_query($conn, "CREATE TABLE users (userid int primary key auto\_increment, username varchar(256), password varchar(256))");  mysqli\_close( $conn );  ?> |

Cilj ovog primera je prezentovanje mogućnosti koje imaju mysqli funkcije, ali se u kombinaciji sa njima koristi SQL sintaksa o kojoj će više reči biti na kursu MySQL programiranje i administracija.  
  
Najpre kreiramo konekciju uz pomoć funkcije mysqli\_connect, kojoj prosleđujemo host, name i password. Zatim vršimo upit kojim kreiramo novu bazu podataka:

|  |  |
| --- | --- |
| 1 | mysqli\_query( $conn, "CREATE DATABASE test\_db" ); |

Odmah potom vršimo i selekciju te baze (ovo nismo mogli izvršiti prilikom konekcije, jer baza nije postojala u tom trenutku):

|  |  |
| --- | --- |
| 1 | mysqli\_select\_db( $conn, "test\_db" ); |

Kako bismo kreirali tabelu, u ovoj bazi vršimo sledeći upit:

|  |  |
| --- | --- |
| 1 | mysqli\_query($conn, "CREATE TABLE users (userid int primary key auto\_increment, username varchar(256), password varchar(256))"); |

Naredba CREATE TABLE služi za kreiranje nove tabele u bazi. Odmah nakon ove naredbe definišemo željeni naziv tabele. Tabela može imati više kolona. Definicija svake kolone se odvaja zarezima. Prva kolona ima naziv userid, tip podataka je int (integer) i ova kolona predstavlja primarni ključ (primary key). Takođe, s obzirom na to da želimo da vrednost ove kolone budu unikatni zapisi, podešavamo i automatsku inkrementaciju auto\_increment. Naredna kolona nosi naziv username, a tip podataka je varchar. Unutar zagrada definišemo broj koji označava maksimalan broj karaktera koji ovde može biti upisan. Istu stvar vršimo i za poslednju kolonu password.  
  
Nakon ovoga zatvaramo konekciju sa bazom.

**Vežba 2**

***Problem:***

Dat je sledeći niz:

$users = array(array("petar", "123"), array("jovan", "456"), array("petar", "789"));

Potrebno je korisnike iz niza smestiti u bazu test\_db, tabelu users, tako da svaki prvi član svakog podniza bude username polje, a svaki drugi član password.

***Rešenje 1:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?php  $conn = mysqli\_connect("localhost", "root", "");  mysqli\_select\_db($conn,"test\_db");  $users = array(array("Peter", "123"), array("John", "456"), array("Thomas", "789"));  foreach($users as $user){      mysqli\_query($conn,"INSERT INTO users (username, password) VALUES ('{$user[0]}', '{$user[1]}')");  }  mysqli\_close($conn);  ?> |

***Rešenje 2:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <?php  $conn = mysqli\_connect("localhost", "root", "");  mysqli\_select\_db($conn,"test\_db");  $users = array(array("Peter", "123"), array("John", "456"), array("Thomas", "789"));  $usersForQuery = "";  foreach($users as $user){     $usersForQuery.="('{$user[0]}', '{$user[1]}'),";  }  if(substr( $usersForQuery, strlen($usersForQuery) - 1) == ","){      $usersForQuery = substr($usersForQuery,0,strlen($usersForQuery)-1);  }  mysqli\_query($conn,"insert into users (username, password) values $usersForQuery");  mysqli\_close($conn);  ?> |

Ovaj zadatak se može uraditi na više načina. Kod prvog načina najpre vršimo konekciju sa bazom i pripremamo niz podataka koji će biti upisani u bazu:

|  |  |
| --- | --- |
| 1  2  3 | $conn = mysqli\_connect("localhost", "root", "");  mysqli\_select\_db($conn,"test\_db");  $users = array(array("Peter", "123"), array("John", "456"), array("Thomas", "789")); |

Da bismo prošli kroz svaki element niza, koristimo foreach petlju, gde u svakoj iteraciji vršimo po jedan upit ka bazi.

|  |  |
| --- | --- |
| 1 | mysqli\_query($conn,"INSERT INTO users (username, password) VALUES ('{$user[0]}', '{$user[1]}')"); |

Naredbom INSERT INTO vrši se upis podataka u bazu. Kolone u kojima će biti izvršen upis mogu biti definisane u običnim zagradama. Nakon ovog dela sledi naredba VALUES, posle koje se u zagradama definišu vrednosti koje će biti upisane u tabelu.  
  
Na kraju možemo zatvoriti konekciju sa bazom:

|  |  |
| --- | --- |
| 1 | mysqli\_close($conn); |

Drugi način podrazumeva isti pristup, s tim da se string koji predstavlja deo upita sa sadržajem promenljivih priprema ranije kroz upotrebu foreach petlje i kontrole toka, i smešta u promenljivu: $usersForQuery.

**Vežba 3**

***Problem:***

Potrebno je izvršiti validaciju korisnika iz baze, na osnovu korisničkog imena i šifre, koji se nalaze u promenljivama:

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <?php  $username = "John";  $password = "456";  $conn = mysqli\_connect("localhost", "root", "");  mysqli\_select\_db($conn,"test\_db");    $r = mysqli\_query($conn,"SELECT \* FROM users WHERE username = '$username' AND password = '$password'");  if(mysqli\_num\_rows($r) == 1)      echo "valid";  else      echo "invalid";  mysqli\_close($conn);  ?> |

|  |
| --- |
| **Napomena**  Ovo rešenje je u potpunosti nebezbedno, jer je na njemu vrlo lako izvršiti sql injection i dobiti validan rezultat u svakom trenutku. |

Zbog toga je obavezno izvršiti što bolju validaciju ulaza. U ovom slučaju, sledeće dve linije dovoljne su da obezbede sistem od pomenutog napada:

|  |  |
| --- | --- |
| 1  2 | $username = mysqli\_real\_escape\_string($conn, $username);  $password = mysqli\_real\_escape\_string($conn, $password); |

Nakon kreiranja promenljivih, konekcije i selektovanja baze podataka, vrši se sledeći upit:

|  |  |
| --- | --- |
| 1 | SELECT \* FROM users WHERE username = '$username' AND password = '$password' |

Rezultat ovog upita se smešta u promenljivu $r. Analizom upita uočavamo da nakon naredbe WHERE sledi uslovaljavanje selekcije, odnosno traži se da i (AND) polje username i polje password odgovaraju vrednostima koje se nalaze u pripremljenim promenljivama.  
  
Nakon upita se na poznat način vrši provera rezultata i ispisivanje poruke:

|  |  |
| --- | --- |
| 1  2  3  4 | if(mysqli\_num\_rows($r) == 1)      echo "valid";  else      echo "invalid"; |

Na samom kraju se konekcija zatvara:

|  |  |
| --- | --- |
| 1 | mysqli\_close($conn); |

**Vežba 4**

***Problem:***

Potrebno je svim poljima u bazi test\_db, tabela users, čija je šifra dužine tri karaktera, dodati ta tri ista karaktera na postojeću šifru (ako je šifra abc, nova šifra treba da bude abcabc i sl.)

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?php  $conn = mysqli\_connect("localhost", "root", "");  mysqli\_select\_db($conn,"test\_db");  $r = mysqli\_query($conn,"SELECT \* FROM users");  while($row = mysqli\_fetch\_array($r)){      mysqli\_query($conn,"UPDATE users SET password = concat(password,password) WHERE length (password) = 3");  }  mysqli\_close($conn);  ?> |

Ovaj primer poseduje sintaksu koja je već dobro poznata, ali i funkcije length i concat koje treba pojasniti. Funkcija length prebrojava karaktere koji se nalaze u polju na koje se odnosi. Funkcija concat vrši konkatenaciju zapisa iz polja. Kako je nama potrebna konkatenacija passworda sa samim sobom, tako kao parametre funkcije definišemo password dva puta, odvojeno zarezima.

**Vežba 5**

***Problem:***

Potrebno je napraviti funkciju koja na osnovu upita vraća dvodimenzionalni niz iz baze.

Potrebno je testirati napravljenu funkciju na tabeli users.

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?php  function arrayFromDB($query)  {      $conn = mysqli\_connect("localhost", "root", "");      mysqli\_select\_db($conn,"test\_db");      $r = mysqli\_query($conn,$query);      $resArray = array();      while($tmpArr = mysqli\_fetch\_row($r))          $resArray[] = $tmpArr;      return $resArray;  }  $users = arrayFromDB("SELECT \* FROM users");  foreach($users as $user)      echo "id: " . $user[0] . ", name: " . $user[1] . ", password: " . $user[2] . "<br>";  ?> |

Funkcija arrayFromDB će nam služiti za izvršavanje upita ka bazi na osnovu prosleđenog parametra. Funkcija na mesto poziva vraća dvodimenzionalni niz sa rezultatima. Unutr funkcije vršimo konekciju sa bazom i izvršavamo upit. Kreiramo niz $resArray, koji će sadržati rezultate kao podnizove. Prolaskom kroz while petlju preuzimamo red po red i smeštamo ga u prethodno pripremljen niz. Nakon što funkcija izvrši sve iteracije, ona vraća pripremljeni niz.

**Vežba 6**

Potrebno je napraviti funkciju koja na osnovu upita prosleđenog kao parametar vraća numeričku vrednost koja predstavlja broj redova nakon upita.

Funkciju treba testirati na tabeli users.

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <?php  function scalar($query)  {      $conn = mysqli\_connect("localhost", "root", "");      mysqli\_select\_db($conn,"test\_db");      $r = mysqli\_query($conn,$query);      if(mysqli\_num\_rows($r) == 1)          {                  $value = mysqli\_fetch\_row($r);                  return (int)$value[0];          }      else          return "";  }  echo scalar("select count(userid) from users");  ?> |

Funkcija scalar() će nam poslužiti za upućivanje upita ka bazi, nakon čega ćemo proveriti vraćenu brojčanu vrednost. Na poslednjoj liniji koda stoji poziv funkcije:

|  |  |
| --- | --- |
| 1 | echo scalar("select count(userid) from users"); |

Ovde koristimo funkciju count(), koja prebrojava redove u tabeli.  
  
Unutar funkcije uspostavljamo konekciju sa bazom i vršimo upit:

|  |  |
| --- | --- |
| 1  2  3 | $conn = mysqli\_connect("localhost", "root", "");  mysqli\_select\_db($conn,"test\_db");  $r = mysqli\_query($conn,$query); |

Zatim proveravamo vraćeni rezultat i konvertujemo podatak u integer, koji i vraćamo iz funkcije:

|  |  |
| --- | --- |
| 1  2 | $value = mysqli\_fetch\_row($r);  return (int)$value[0]; |

Ukoliko nema rezultata, vraćamo prazan string:

|  |  |
| --- | --- |
| 1 | return ""; |

**Koja mysqli funkcija prosleđuje upit MySQL serveru?**

 mysqli\_query

 mysqli\_connect

 mysqli\_fetch\_result

 mysqli\_close