

Web programiranje

Vežbe 9 - Rad sa bazom podataka SQLite

Od verzije 5.0, PHP ne podržava MySQL bazu odmah po instalaciji podrazumevanog paketa, već je potrebno da korisnik (ili administrator) napravi izbor da li će koristiti klasičan **ext/mysql** ili **ext/mysqli** modul (ili oba), što neiskusnim korisnicima ponekad može biti dodatni balast.

Nasuprot tome, od verzije PHP 5.0, u jezgro PHP-a dodat je modul za podršku **SQLite** bazi podataka koji funkcioniše na doslovno svakoj PHP instalaciji. **SQLite** je, za razliku od **MySQL**-a, baza podataka bazirana na fajlovima, a ne na klijent-server paradigmi. Namenjena je jednostavnijim zadacima i manjem obimu baze, ali poseduje skoro sve klasične SQL komande i pored toga “teži” manje od 500KB. SQLite je projekat otvorenog koda i dosta se koristi i u raznim muzičkim plejerima, mobilnim telefonima... Baza u SQLite može imati veličinu do čak 2TB i u nekim situacijama je čak brža od MySQL-a.

Kreiranje baze i tabele u bazi

Evo jednostavnog načina da se kreira baza u fajlu i tabela u toj bazi:

```
C:\SQLite>sqlite3 preduzece.db
SQLite version 2.18.17
sqlite> create table radnik(
...> sradnik integer primary key,
...> ime varchar(255),
...> adresa varchar(255),
...> plata float
);
sqlite> insert into radnik(ime,adresa,plata) values('Arnold Svarceneger', 'JNA 433',
3567.78);
sqlite> insert into radnik(ime,adresa,plata) values('Kimi Raikonen', 'Djure Jaksica 89',
2564.11);
sqlite> insert into radnik(ime,adresa,plata) values('Muhamed Ali', 'Sumarice 11',
1564.22);
sqlite> select * from radnik;
1|Arnold Svarceneger|JNA 433|3567.78
2|Kimi Raikonen|Djure Jaksica 89|2564.11
3|Muhamed Ali|Sumarice 11|1564.22
```

Primećuje se da kolone koje su označene kao INTEGER PRIMARY KEY, SQLite tretira i kao AUTO_INCREMENT, tj. ukoliko nisu navedeni, ključevi se automatski dodeljuju. Iz PHP-a postoje čak 3 načina da se pristupi SQLite bazi, i to:

- interfejsom objektnog tipa
- pomoću PDO drajvera

<http://www.sqlite.org/different.html>

<http://www.chipkin.com/articles/sqlite-vs-mysql-short-summary>

Objektni pristup SQLite bazi

```
<html>
<head></head>
<body>
<?php
// setuj lokaciju fajla s bazom
$baza = "D:/SVN/preduzece.db";

// otvori fajl sa bazom
$konekcija = new SQLite3($baza) or die("Ne mogu da otvorim bazu");

// generisi string za upit
$upit = "SELECT * FROM radnik";

// izvrši upit, vraća se
$rezultat = $konekcija->query($upit) or die("Greska u upitu! ");

// ako ima rezultata
if ($rezultat->numColumns()) {
    // uzmi svaki red kao niz
    // stampaj vrednosti
    echo "<table cellpadding=10 border=1>";
    while($red = $rezultat->fetchArray()) {
        echo "<tr>";
        echo "<td>".$red[0]."</td>";
        echo "<td>".$red[1]."</td>";
        echo "<td>".$red[2]."</td>";
        echo "<td>".$red[3]."</td>";
        echo "</tr>";
    }
    echo "</table>";
}

// sve je završeno, zatvori konekciju
unset ($konekcija);
?>
</body>
</html>
```

PDO pristup SQLite bazi

I ovde PDO pokazuje sve svoje prednosti. Pored višeg nivoa apstrakcije i mogućnosti da se isti kod koristi na različitim bazama podataka, PDO dodaje i mogućnost korišćenja **SQLite verzije 3**, što kao mogućnost ne postoji ukoliko se koristi klasičan **SQLite** interfejs. PHP zajednica definitivno ide ka tome da se ubuduće skoro sav posao pristupa bazama podataka, osim nekih specifičnih slučajeva, odvija preko PDO interfejsa.

```
<html>
<head>
<title>PDO citanje iz SQLite tabele radnik</title>
</head>

<body>
<h1>Radnici</h1>

<?php
    // try-catch blok u kome se otvara konekcija na bazu i vrsi upit
    try {
        $konekcioni_string = "sqlite:c:/SQLite/preduzece.db";
        // $dbh objekat pamti konekciju prema bazi
        $dbh = new PDO($konekcioni_string);

        // konstrukcija SQL izraza
        $sql = "SELECT * FROM radnik";

        // $dbh->query() vraca PDOStatement objekat
        $pdo_izraz = $dbh->query($sql);
        // Svi podaci se vraćaju u obliku niza asocijativnih nizova
        $niz = $pdo_izraz->fetchAll(PDO::FETCH_ASSOC);

        // Iterisanje po vracenim rezultatima
        echo "<table cellpadding='5' border='1'>";
        echo "<tr><th>Ime</th><th>Adresa</th><th>Plata</th></tr>";
        foreach($niz as $radnik) {
            echo "<tr><td><b>". $radnik['ime']. "</b></td>";
            echo "<td>". $radnik['adresa']. "</td>";
            echo "<td>". $radnik['plata']. "</td></tr>";
        }
        echo "</table>";

    }
    catch(PDOException $e) {
        echo "GRESKA: ";
        echo $e->getMessage();
    }

    // Zatvaranje konekcije vrsi se jednostavnim dodeljivanjem null
    $dbh = null;
    ?>

</body>
</html>
```

Transakcije pomoću PDO interfejsa

PDO takođe podržava i vrlo koristan mehanizam transakcija, ali samo na onim RDBMS-ovima koji ga podržavaju. Na primer, SQLite sa bilo kojim i MySQL sa standardnim **MyISAM** tipom tabele (engine) **NE PODRŽAVAJU** transakcije! Ukoliko postoji zahtev za implementaciju transakcija, potrebno je izabrati *engine* **InnoDB** prilikom kreiranja MySQL tabele. Primer:

1) ALTER TABLE `knjige` ENGINE=InnoDB;

2) <?php

```
$ime_hosta = "localhost";
$korisnik = "branko";
$sifra = "*****";
$ime_baze = "branko";

// try-catch blok u kome se otvara konekcija na bazu i vrsi upit
try {
    $konekcioni_string = "mysql:host=$ime_hosta;dbname=$ime_baze";
    // $dbh objekat pamti konekciju prema bazi
    $dbh = new PDO($konekcioni_string, $korisnik, $sifra);
}
catch(PDOException $e) {
    echo "GRESKA: ";
    echo $e->getMessage();
}

try {
    /* Pocni transakciju, ovim se iskljucuje autocommit */
    $dbh->beginTransaction();

    /* Imeni podatke u bazi */
    $sth = $dbh->exec("DELETE FROM knjige WHERE naslov LIKE '%godina%'");
    $sth = $dbh->exec("UPDATE knjige SET naslov = 'Prokleta avlija'
        WHERE naslov LIKE '%godina%'");

n.    //$dbh->rollBack();
    $dbh->commit();
    /* Konekcija se vraca u autocommit mod */
}
catch(PDOException $e) {
    /* Ukoliko se prepozna greska, vrsi se povratak */
    $dbh->rollBack();
    echo "GRESKA: ";
    echo $e->getMessage();
}

// Zatvaranje konekcije vrsi se jednostavnim dodeljivanjem null
$dbh = null;
?>
```

U slučaju da linija **n** nije pod komentarima, sa tabelom se ne bi desilo ništa, jer je izvršen **rollBack()**. Tek kada se pozove **commit()**, cela transakcija biva izvršena.

Zadatak:

- Napisati PHP funkciju **prebaci_cenu(\$id1, \$id2, \$iznos)** koja knjizi sa šifrom **\$id2** prebacuje iznos **\$iznos** od knjige sa šifrom **\$id1**. Obavezno je korišćenje transakcija.
- Napraviti odgovarajuću web formu za izvođenje gorepomenute transakcije koja korisniku omogućava da izborom dveknjige (dva *combo-box*-a) i upisom iznosa koji treba prebaciti obavi prenos. Potrebno je i obraditi greške, npr. kada je uneti iznos za prenos veći od cene koju izabrana knjiga ima, kada se unese nešto što nije broj u polje za količinu novca ili kada se izaberu iste knjige.

Knjiga_db.php

```
<?php

/**
 * Klasa cije metode obavljaju sav pristup tabeli knjiga iz baze
 */
class Knjiga_db {
// Konstante
    const ime_hosta = 'localhost';
    const korisnik = 'branko';
    const sifra = '*****';
    const ime_baze = "branko";

    // Atributi
    private $dbh;

// konekcija prema bazi
    // Metode
    // Zadatak konstruktora je otvaranje prema bazi
    function __construct() {
        try {
            $konekcioni_string="mysql:host=".self::ime_hosta.";dbname=".self::ime_baze;
            $this->dbh = new PDO($konekcioni_string, self::korisnik, self::sifra);
        }
        catch(PDOException $e) {
            echo "GRESKA: ";
            echo $e->getMessage();
        }
    }

    // Zadatak destruktora je zatvaranje konekcije prema bazi
    function __destruct() {
        $this->dbh = null;
    }

    * Vraca niz asocijativnih nizova svih podataka o knjigama
    * @return array
    */
    public function sve_knjige() {
        try {
            $sql = "SELECT id, naslov, autor, godina,cena FROM knjiga";

            $pdo_izraz = $this->dbh->query($sql);
            $niz = $pdo_izraz->fetchAll(PDO::FETCH_ASSOC);

            return $niz;
        }
    }
}
```

```

        catch(PDOException $e) {
            echo "GRESKA: ";
            echo $e->getMessage();
        }
    }









    /**
     * Vraca asoc. niz podataka o knjizi ciji se ID salje kao argument
     * @param int $id
     * @return asoc. array
     */
    public function podaci_o_knjizi($id) {
        try {
            $sql = "SELECT id, naslov, autor, godina, cena FROM knjiga WHERE id=$id";
            $pdo_izraz = $this->dbh->query($sql);
            $obj = $pdo_izraz->fetch(PDO::FETCH_ASSOC);
            return $obj;
        }
        catch(PDOException $e) {
            echo "GRESKA: ";
            echo $e->getMessage();
        }
    }

    /**
     * Izvršava transakciju prenosa iznosa $iznos sa knjige $id1 na knjigu $id2
     * @param int $id1
     * @param int $id2
     * @param float $iznos
     */
    public function izvrši_transakciju($id1, $id2, $iznos) {
        try {
            /* Pocni transakciju, ovim se isključuje autocommit */
            $this->dbh->beginTransaction();
            /* Izmeni podatke u bazi */
            $sth = $this->dbh->exec("UPDATE knjiga SET cena=cena-$iznos WHERE id=$id1");
            $sth = $this->dbh->exec("UPDATE knjiga SET cena=cena+$iznos WHERE id=$id2");
            $this->dbh->commit();
            /* Konekcija se vraca u autocommit mod */
        }
        catch(PDOException $e) {
            $this->dbh->rollBack();
            echo "GRESKA: ";
            echo $e->getMessage();
        }
    }
}

} // Kraj klase Knjiga_db

?>

```

			id	naslov	autor	godina	cena
<input type="checkbox"/>			1	Sto godina samoce	Gabrijel Markes	1957	140
<input type="checkbox"/>			2	Na drini cuprija	Ivo Andric	1985	233
<input type="checkbox"/>			3	Prokleta avlija	Ivo Andric	1967	186
<input type="checkbox"/>			4	Deris i smrt	Mesa Selimovic	1978	200

transakcija.php

```

<html>
<head>
<title>Transakcija</title>
</head>

<body>
<h1>Transakcija za cene knjiga</h1>

<?php
    require_once 'Knjige_db.php';
    $knjiga_db = new Knjiga_db();

    // Ukoliko je forma poslata POST metodom, vrsi se obrada transakcije
    if (isset($_POST['izvrsi'])) {
        $id1 = $_POST['id1'];
        $id2 = $_POST['id2'];
        $iznos = $_POST['iznos'];

        // Uzimaju se podaci o knjizi da bi se videla njena cena
        $knjiga1 = $knjiga_db->podaci_o_knjizi($id1);

        // Nije dozvoljena transakcija izmedju istih knjiga
        if ($id1==$id2) {
            echo "<label style='color:red'>GRESKA: Ne moze se izvrstiti prenos istoj
                knjizi!</label>";
        }
        // Nije dozvoljen prenos ako cena manja od transfera
        elseif ($knjiga1['cena']<$iznos) {
            echo "<label style='color:red'>GRESKA: Knjiga nema dovoljno sredstava za
                prenos!</label>";
        }
        else {
            $knjiga_db->izvrsi_transakciju($id1, $id2, $iznos);
        }
    }

    // Postavi formu za transakciju
    $knjige = $knjiga_db->sve_knjige();

    // Opcije za combo-box se pamte u stringu da bi se pozivale za oba box-a
    $opcije_html = '';
    foreach ($knjige as $knjiga) {
        $id = $knjiga['id'];
        $naslov = $knjiga['naslov'];
        $autor = $knjiga['autor'];
        $opcije_html .= "<option value='$id'>$naslov - $autor</option>";
    }

    echo "<form action='".$_SERVER['PHP_SELF']."' method='POST'>";
    echo "Od knjige <select name='id1'>$opcije_html</select>";
    echo "knjizi <select name='id2'>$opcije_html</select>";
    echo "iznos: <input type='text' name='iznos' size='10' />";
    echo "<input type='submit' name='izvrsi' value='izvrsi' />";
    echo "</form>";

    ?>

</body>
</html>

```

transakcija.php -- validacija pomoću javascript-a (rešenje Marko Knežević)

U rešavanju zadatka je korišćen HTML5: http://www.w3schools.com/tags/att_global_data.asp

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Promena cene</title>
    <script>
      function setPrice(element, text)
      {
        elem = document.getElementById(text);
        elem.value = element.options[element.selectedIndex].getAttribute("data-
        cena");
      }

      function frm_submit()
      {
        elem = document.getElementsByName("knjiga1")[0];
        val1 = elem.options[elem.selectedIndex].value;
        elem = document.getElementsByName("knjiga2")[0];
        val2 = elem.options[elem.selectedIndex].value;
        if ((val1 === 'a') || (val2 === 'a') || (val1 === val2)){
          alert('Knjige moraju biti izabrane i razlicite!');
          return false;
        }
        val2 = document.getElementById("novaCena").value;
        if ((val2 == "") || (isNaN(val1))){
          alert('Cena mora biti broj');
          return false;
        }

        cena1 = document.getElementById("cena1").value;
        if (cena1 < val2){
          alert('Promena cene mora biti manja od cene prve knjige');
          return false;
        }
        return true;
      }
    </script>
  </head>
  <body>
    <?php
      require_once 'Knjiga_db.php';
      $knjige = new Knjiga_db();
      if (isset ($_POST['knjiga1']))
        $knjige->izvrsi_transakciju ($_POST['knjiga1'], $_POST['knjiga2'],
        $_POST['promena']);
    ?>
  </body>
</html>
```



```

<form action="<?php echo $_SERVER['PHP_SELF'];>" method="post" onsubmit="return
frm_submit();">
    <select name="knjiga1" onchange="setPrice(this, 'cena1');">
        <option value="a">Izaberi knjigu...</option>

        <?php
            $niz = $knjige->sve_knjige();
            foreach ($niz as $knjiga) {
                echo "<option value='". $knjiga['id']."' data-cena=
                    \"\". $knjiga['cena']. \"\">\". $knjiga['naslov']. \"</option>";
            }
        ?>
    </select>
    <input type="text" id="cena1" readonly /><br/>
    <select name="knjiga2" onchange="setPrice(this, 'cena2');">
        <option value="a">Izaberi knjigu...</option>
        <?php
            foreach ($niz as $knjiga) {
                echo "<option value='". $knjiga['id']."' data-cena=
                    \"\". $knjiga['cena']. \"\">\". $knjiga['naslov']. \"</option>";
            }
        ?>
    </select>
    <input type="text" id="cena2" readonly /><br/>
    Promena cene: <input type="text" name="promena" id="novaCena"/><br/>
    <input type="submit" value="Izmeni"/>
</form>
</body>
</html>

```