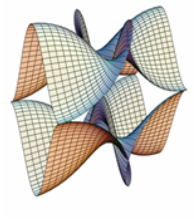


Prirodoslovno-matematički fakultet
Matematički odsjek
Sveučilište u Zagrebu

RAČUNARSKI PRAKTIKUM II

Predavanje 06 - PHP - Datoteke i baze podataka

9. travnja 2018.

Sastavio: Zvonimir Bujanović

Rad s datotekama

- PHP sadrži velik broj funkcija za rad s datotekama kao u C-u: `fopen`, `fclose`, `fscanf`, `fprintf`, `feof`, `fread`, `fwrite` ...

- Na primjer,

```php
if( ($f = fopen( '/tmp/data.txt' , 'w' )) === false )
    exit( "Ne mogu otvoriti file: $php_errormsg" );

if( fwrite( $f, $_COOKIE['boja'] ) === false )
    exit( "Ne mogu pisati u file: $php_errormsg" );

if( fclose( $f ) === false )
    exit( "Ne mogu zatvoriti file: $php_errormsg" );
```

Rad s datotekama

- User i/ili grupa kojem pripada PHP (na Linuxu obično `www-data`) mora imati pravo čitanja/pisanja/pristupanja direktoriju i datoteci s kojom radimo.

- Najjednostavnije: "ostalima" damo pravo `r` ili `w` ili `x`.

- Ako želimo čitati iz datoteke, onda:
  - Folder u kojem se datoteka nalazi mora imati pravo `x`.
  - Sama datoteka mora imati pravo `r`.

- Ako želimo pisati u već postojeću datoteku, onda:
  - Folder u kojem se datoteka nalazi mora imati pravo `x`.
  - Sama datoteka mora imati pravo `w`.

- Ako želimo stvoriti novu datoteku, onda:
  - Folder u kojem će se datoteka nalaziti mora imati pravo `w`.

Rad s datotekama

- `fprintf` - kao u C-u. Vraća broj ispisanih znakova.

```
1 $f = fopen( 'proba.txt', 'w' );
2 $dbl = 3.14; $str = 'Pero';
3 fprintf( $f, 'Broj: %5.2f, string: %s\n',
4               $dbl, $str );
```

- `fscanf` - svaki poziv pročita cijelu jednu liniju datoteke!

```
1 $f = fopen( 'users.txt', 'r' );
2 while( $userinfo = fscanf( $f, "%s\t%s\t%s\n" ) )
3 {
4     list( $name, $profession, $countrycode ) = $userinfo;
5     ...
6 }
```

Rad s datotekama

- PHP ima niz funkcija koje pojednostavljuju uobičajene zadaće.
- Učitavanje cijele datoteke u string sa `file_get_contents`:

```
1 $users = file_get_contents( 'users.txt' );
2 if( preg_match( '/username:(Ana|Pero)/' , $users ) )
3     echo 'Postoji user Ana ili Pero.';
```

Uoči: string u kojeg je učitan file zauzima memoriju dok se PHP skripta izvršava!

- Slično, postoji `file_put_contents`.
- Sa `readfile($filename)` se ispisuje sadržaj cijele datoteke. Ovo je efikasnije nego učitati u string pa ispisati sa `echo`.

Zadatak 1

Napišite PHP skriptu **`zadatak1.php`** koja:

- Pruža mogućnost korisniku da upiše svoje ime u formu.
- U datoteci **`users.txt`** drži popis od najviše 5 zadnjih imena korisnika koji su posjetili stranicu (starija imena se "zaboravljaju", tj. brišu). Imena su odvojena zarezima i sva se nalaze u jedinom retku datoteke.
- Iza forme za upis imena, ispisuje taj popis.

Uputa:

- Koristite `file_exists` da provjerite da li datoteka postoji.
- Koristite `file_get_contents` i `file_put_contents` za čitanje i pisanje u file.
- Koristite `explode` i `implode` za brzu konverziju pročitanog stringa u polje i obratno.

Rad s datotekama

- Obično izbjegavamo rad s datotekama dostupnim kroz web. Dakle, trebalo bi ih smjestiti izvan `public_html`!

- Na primjer:
  - Skripta je smještena u `/student1/pero/public_html/rp2/skripta.php`.
  - Datoteka je smještena u `/student1/pero/aux/users.txt`.
  - U skripti do datoteke dolazimo sa:

```php
// __DIR__ = direktorij u kojem se nalazi PHP skripta.
// Ovdje: __DIR__ = '/student1/pero/public_html/rp2'
$fileName = __DIR__ . '/../../aux/users.txt';
```

Rad s datotekama - Lokoti

- Što ako više korisnika istovremeno treba čitati/pisati u istu datoteku?

- Rješenje su tzv. lokoti i poziv funkcije `flock`.
  - `LOCK_EX` – ekskluzivni lokot. Ograničava pristup datoteci na samo jedan proces. Koristi se prilikom pisanja u datoteku.
  - `LOCK_SH` – dijeljeni lokot. Više procesa ima pravo pristupa datoteci. Koristi se prilikom čitanja iz datoteke.
  - `LOCK_UN` – otpuštanje lokota.
- Po defaultu, pokušaj dobivanja lokota sa `flock()` je blokirajući, tj. skripta stoji na toj naredbi dok ne dobije pristup datoteci.

- Tipično, umjesto datoteka ćemo koristiti baze podataka koje nemaju ovakve probleme.

Rad s datotekama - Lokoti

```php
$f = fopen( 'guestbook.txt', 'a' );

// Pokušaj dobiti ekskluzivni lokot.
if( flock( $f, LOCK_EX ) )
{
    // Dobili smo lokot. Sada smijemo zapisati nešto u datoteku.
    fwrite( $f, $_POST['poruka'] );

    // Prije otključavanja, treba napraviti fflush.
    fflush( $f );

    // Otključamo lokot.
    flock( $f, LOCK_UN );

    // Zatvorimo datoteku.
    fclose( $f );
}
else
    echo "Ne mogu dobiti lokot..." ;
```

Rad s datotekama

Još neke funkcije:

- `basename` – dohvati samo ime datoteke iz cijele putanje.
- `dirname` – dohvati samo direktorij iz cijele putanje.
- `file_exists` – vraća true/false ovisno o tome postoji li datoteka.
- `is_readable`, `is_writable`
- `filesize`
- `filemtime` – vrijeme zadnje izmjene datoteke.
- `unlink` – brisanje datoteke.
- `chmod`, `chown`, `chgrp` – promjena prava pristupa i vlasništva na datotekom.
- `copy`, `rename`, `mkdir`, `rmdir`

Rad s datotekama - Sigurnosni aspekti

- Što ako je `username='../etc'` i `filename='passwd'`?
- Što ako se PHP izvršava sa `root` ovlastima?

```php
$username = $_POST['username'];
$userfile = $_POST['filename'];
$homedir  = "/home/$username";
$filepath = "$homedir/$userfile";

unlink( $filepath );
```

Rad s datotekama - Sigurnosni aspekti

- Što ako je `username='../etc'` i `filename='passwd'`?
- Što ako se PHP izvršava sa `root` ovlastima?

```php
$username = $_POST['username']; // sanit+valid+session...
$userfile = basename($_POST['filename']);

if( !preg_match('/^[a-z0-9_-]+$/', $username )  ||
    !preg_match('/^[a-z0-9_-]+(\.[a-z0-9_-]+)?$/', $userfile) )
{
    exit( 'Bad username/filename' );
}
else {
    $homedir  = "/home/$username";
    $filepath = "$homedir/$userfile";
    unlink( $filepath );
}
```

Baze podataka

- PHP podržava bilo koju ODBC (Open Database Connectivity) bazu.
- Specijalno dobra podrška za SQL baze podataka.
- Mi ćemo se prvenstveno orijentirati na MySQL (tj. MariaDB).
- PHP nudi 3 sučelja za pristup MySQL bazama:
  - `mysql` – zastarjelo, ne koristiti.
  - `mysqli` – za pristup isključivo MySQL bazama. Nudi i proceduralno i objektno-orijentirano sučelje.
  - `PDO` (PHP Data Objects) – konzistentno i fleksibilno sučelje za pristup raznim tipovima baza podataka, objektno orijentirano.
- Koristit ćemo PDO zbog fleksibilnosti, jer podržava i SQLite:
  - Baza podataka koja ne zahtijeva dedicirani server, nego koristi običnu datoteku.
  - `sudo apt-get install php5-sqlite`
  - Instalirana na rp2-serveru (ali ne i na studentu).

Zadatak 2

- Ulogirajte se u phpmyadmin na rp2-serveru.
- Pronađite bazu čije ime je jednako vašem prezimenu.
- U toj bazi stvorite tablicu `Studenti`:
  - Tablica neka ima stupce JMBAG, Ime, Prezime, Ocjena.
- Unesite nekoliko redaka u tu tablicu.

Baze podataka - Spajanje iz PHP-a

- Spajanje na bazu podataka = konstrukcija objekta tipa `PDO`:

```php
$user = 'pero'; $pass = 'perinasifra';

try {
  $db = new PDO(
        'mysql:host=db.com;dbname=imeBaze;charset=utf8',
       $user, $pass );
} catch( PDOException $e ) {
  echo "Greška: " . $e->getMessage(); exit();
}
```

- *Collation* u bazi podesiti na `utf8_unicode_ci` (ako već nije).
- Podatke za spajanje treba držati nedostupnim kroz web (konfiguracijska datoteka/.htaccess/izvan www-root).
- Za spajanje na SQLite, samo treba poslati drugi string konstruktoru; nema username/password:

```php
$db = new PDO( 'sqlite:/tmp/db.sqlite' );
```

Baze podataka - Upiti

- Upiti se kreiraju pomoću članske funkcije `query` kojoj proslijedimo SQL naredbu.

```
1 $st = $db->query( 'SELECT JMBAG,Ime,Prezime FROM Studenti' );
```

- `query` vraća objekt tipa `PDOStatement`. Kroz retke rezultata možemo iterirati sa `fetch` ili `fetchAll`.

```
1 // ili: while( $row = $st->fetch() )
2 foreach( $st->fetchAll() as $row )
3     echo " JMBAG = " . $row[ 'JMBAG' ] .
4          " Ime = " . $row[ 'Ime' ] .
5          " Prezime = " . $row[ "Prezime" ] . "<br />\n";
```

Zadatak 3

- Spojite se na svoju bazu na rp2-serveru.
- Dohvatite sve podatke o studentima iz tablice `Studenti`.
- Prikažite te podatke u HTML tablici.

Baze podataka - Izmjena podataka

- Dodavanje, brisanje ili izmjena podataka se može raditi pomoću funkcije exec.

```
1 $db->exec("INSERT INTO Studenti (JMBAG,Ime)" .
2                     " VALUES ('8293746591','Ana')" );
3 $db->exec("DELETE FROM Studenti WHERE Ime LIKE 'Ana'" );
4 $db->exec("UPDATE Studenti SET Prezime='Anić'" .
5                           " WHERE JMBAG='8293746591'" );
```

- `exec` vraća broj izmijenjenih/dodanih redaka.

- Oprez: podaci koje je unio korisnik neće biti sanitizirani ako koristimo `exec` i `query`! Zato je bolje koristiti tzv. *prepared statements*, iako zahtjevaju dva koraka: prepare + execute.

SQL Injection

- Napadač kreira ili mijenja postojeće SQL naredbe da bi otkrio skrivene podatke ili izvršio opasne naredbe na serveru gdje se nalazi baza.
- Ovo postiže tako da ubacuje neočekivane parametre u SQL upite.

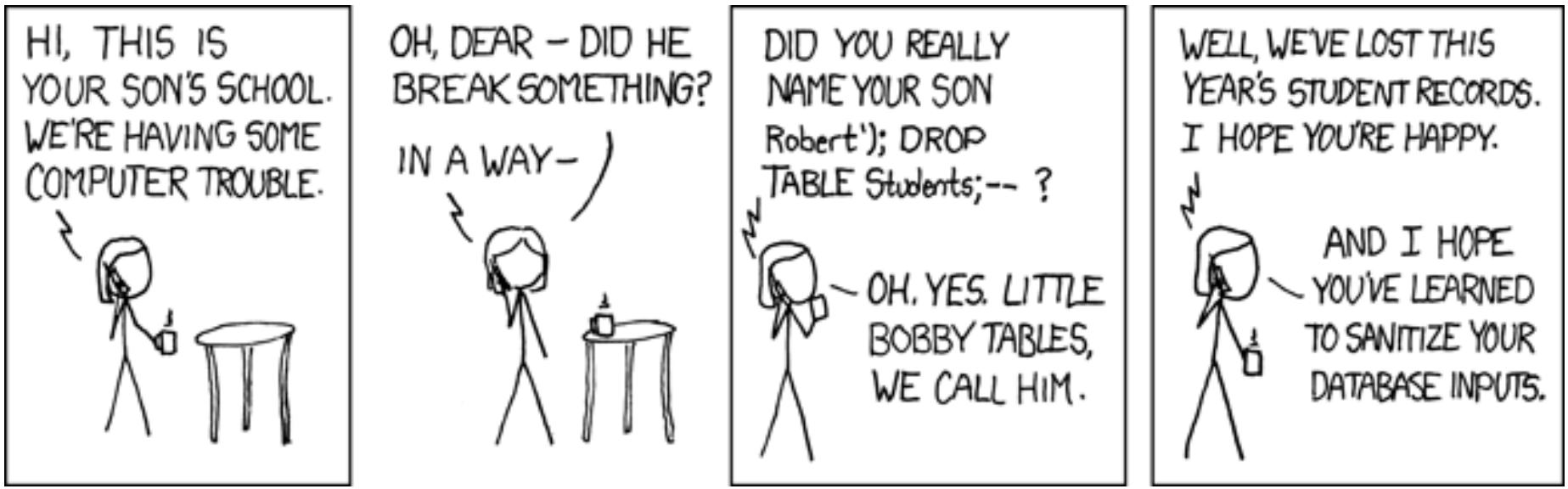
```
1 $st = $db->query( "SELECT * FROM users WHERE name ='" .
2                   $userName . "'" );
```

- Ako je `$username` jednak `' OR '1'='1`, onda:
  `SELECT * FROM users WHERE name ='' OR '1'='1'`
  ⇝ dohvaćena je cijela tablica sa svim korisnicima!

- Ako je `$username` jednak `a';DROP TABLE users; SELECT * FROM userinfo WHERE 't' = 't`, onda:
  ⇝ obrisana tablica `users`, dohvaćeni svi podaci iz druge tablice!

- Rješenje: sanitizacija + validacija, *prepared statements*.

Baze podataka - Sigurnosni aspekti



https://xkcd.com/327/

Baze podataka - Ponavljanje upita

- Ako se isti upit ponavlja više puta, ali sa različitim parametrima, bolje je koristiti *prepared statements*.
- Prednosti:
  - Upiti su efikasniji jer se parsiraju samo jednom.
  - Upiti su sigurniji jer se parametri automatski sanitiziraju.

```
1 $st = $db->prepare( "SELECT JMBAG FROM Studenti " .
2                     "WHERE Ime LIKE :ime " .
3                     "OR Prezime LIKE :prezime " );
4
5 $st->execute( array( 'ime' => 'Mirko',
6                      'prezime' => 'Anić' ) );
7
8 while( $row = $st->fetch() )
9     echo "JMBAG = " . $row[ 'JMBAG' ] . "<br />\n";
```

- Pripremiti i izvršiti se mogu i INSERT/DELETE/UPDATE. Nakon izvršavanja sa `$st->rowCount()` možemo doznati broj izmijenjenih redaka.

Baze podataka - Detekcija grešaka u upitima

- Ako želimo detektirati greške prilikom spajanja, te greške prilikom pripremanja i izvršavanja upita, potrebno je konfigurirati konekciju na bazu ovako:

```php
$db = new PDO( $db_base, $db_user, $db_pass );
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$db->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
```

- Nakon toga, možemo hvatati iznimke koje bacaju PDO naredbe:

```php
try {
    $st = $db->prepare( 'SELECT JMBAG FROM Studenti' );
    $st->execute();
}
catch(PDOException $e) {
    echo 'Greška: ' . $e->getMessage();
}
```

Zadatak 4

Modificirajte rješenje Zadatka 3 tako da:

- Dodate formu u kojoj pitate korisnika za ocjenu studenta.
- Nakon slanja forme, prikazuju se samo studenti koji imaju tu ocjenu.
- Koristite *prepared statement*.

Singleton pattern

- Problem: puno pomoćnih funkcija treba dohvaćati podatke iz baze podataka
  ⇝ ili bi svaka funkcija trebala ponovno stvarati PDO objekt,
  ⇝ ili bismo svakoj trebali prosljeđivati PDO objekt.

- Rješenje (jedno moguće): tzv. singleton pattern za klasu `DB`:
  - osigurat će da postoji samo jedan PDO objekt u cijelom programu;
  - osigurat će da je objekt dostupan svima.

  ⇝ privatni prazni konstruktor, zabrana kopiranja;
  ⇝ statička funkcija `getConnection()` za pristup PDO objektu koji će biti stvoren samo jednom.

```
1 // Pristup iz svake funkcije će izgledati ovako:
2 $db = DB::getConnection();
```

Singleton pattern

```php
class DB {
    // Interna statička varijabla koja čuva konekciju na bazu
    private static $db = null;

    // Zabranimo new DB() i kloniranje;
    final private function __construct() { }
    final private function __clone() { }

    // Statička funkcija za pristup bazi.
    public static function getConnection() {
        // Spoji se samo ako već nisi nekad ranije.
        if( DB::$db === null ) {
            // U glob. varijablama su parametri za spajanje
            global $db_base, $db_user, $db_pass;
            DB::$db = new PDO($db_base, $db_user, $db_pass);
        }

        return DB::$db;
    }
}
```

Baze podataka - Sigurnosni aspekti

- Pretpostavimo da imamo tablicu u kojoj čuvamo korisničke podatke.
- Kako čuvati lozinke?
  - Nikad ih ne čuvati doslovno onako kako ih je unio korisnik!
  - Lozinke je potrebno hashirati i spremiti samo hash u tablicu.
  - Ako netko i dobije neovlašteni pristup bazi, neće vidjeti lozinku nego samo hash.
  - U donjem primjeru, u bazi za hash treba polje od 255 znakova.

```
1 // U bazu spremamo $hash_password, a ne $_POST["password"]!
2 $hash = password_hash( $_POST["password"],
3                        PASSWORD_DEFAULT );
```

```
1 // Kasnije, da provjerimo je li uneseni $_POST["password"] OK,
2 // dohvatimo $hash iz baze i onda:
3 if( password_verify( $_POST["password"], $hash ) )
4 { ... }
```

Zadatak 5

Napišite PHP skriptu **`zadatak5.php`** koja prikazuje formu za unos korisničkog imena i lozinka.

- Forma ima 2 gumba: "Ulogiraj se", "Stvori novog korisnika".
- Klikom na prvi gumb, provjerava se u bazi postoji li taj korisnik i je li to njegova lozinka. Ako da, ispisuje se "Dobro došli, uspješno ste se ulogirali". Ako ne, ponovno se prikazuje forma.
- Klikom na drugi gumb, provjerava se u bazi postoji li taj korisnik, i ako ne postoji, dodaje se u bazu zajedno s pripadnom lozinkom. Treba ispisati odgovarajuću poruku ("Dodani ste u bazu" ili "Taj korisnik već postoji").

Koristite **`DB::getConnection()`**, te hashiranje passworda prije spremanja u bazu.

```
1 // Bit će $_POST["gumb"]==="login" ili $_POST["gumb"]==="novi"
2 // ovisno o tome na koji je gumb kliknuto:
3 <button type="submit" name="gumb" value="login">Logiraj!</button>
4 <button type="submit" name="gumb" value="novi">Novi!</button>
```

Primjer 1 - SQLite

- SQLite je baza podataka koja je jednostavno spremljena u jednoj datoteci.
- Zbog toga ju je vrlo lako prenijeti s jednog servera na drugi.
- Apache/PHP trebaju podržavati taj tip baze.
- Spajanje na bazu:

```
1 $db = new PDO( 'sqlite:/tmp/studenti.sqlite' );
```

  Ovdje je baza spremljena u datoteku `/tmp/studenti.sqlite`
- Sve ostale funkcije iz PDO sada rade kao i sa MySQL!
- Postoji više programa za rad sa SQLite bazama, na primjer (besplatni i open-source) DB Browser for SQLite.