Web dizajn i programiranje

Prof. dr.sc. Dragutin Kermek
Sveučilište u Zagrebu
Fakultet organizacije i informatike
Pavlinska 2, Varaždin 42000
dkermek@foi.hr

16. dio

Programiranje na strani poslužitelja

Uvod u programiranje na strani poslužitelja/CGI.
Programski jezik PHP.
Izvršavanje obrade (pretraživanje, zapis podataka i sl.) na strani poslužitelja.
Ponovno korištenje koda.
Rad s datumom i vremenom. Objektna orijentacija.
Slanje e-mail poruke. Generiranje HTML stranica.
Upravljenje pogreškama.
Korištenje i administriranje baze podataka (MySQL).
Realizacija autentikacije i autorizacije korisnika.

Rad s datotekama

Aplikacijska logika najčešće traži da WWW poslužitelj pohranjuje i čita podatke iz datoteke (razne postavke npr. vezanu uz pristup do baze podataka, aktivnost pojedinih modula i sl.). Danas se sve više koristi zapis u XML formatu.

Postoje glavne funkcije:

- `fopen` - otvaranje
- `fclose` - zatvaranje
- `fread` - čitanje određenog broja byte-a
- `file` - učitavanje cijele datoteke u niz (svaki red jedan element)
- `fwrite` - upisivanje
- `flock` - zaključavanje
- `fseek` - pozicioniranje

Otvaranje datoteke

Datoteka može biti otvorena za jednu ili više vrsta operacija:

```
$fp = fopen ("podaci.txt", "r");
```
- čitanje

```
$fp = fopen ("podaci.txt", "w+");
```
- upisivanje tj. prepisivanje

```
$fp = fopen ("podaci.txt", "a+");
```
-dodavanje na kraj

Otvaranje datoteke / 1.

```
<form id="form1" method="post" name="form1" action="primjer31.php">
  <p><label for="ime">Ime: </label>
  <input name="ime" size="20" /><br />
  <label for="prezime">Prezime: </label>
  <input name="prezime" size="20" /><br />
  <input id="submit1" name="submit1" type="submit" value="Šalji" />
  <input id="reset1" name="reset1" type="reset" value="Obriši" /> </p>
</form>
```


Otvaranje datoteke / 2.

```
<?php
$ime = $_POST["ime"];
$prezime = $_POST["prezime"];
$fp = fopen ("podaci/ADRESAR.TXT", "a+");
fwrite ($fp, $ime);
fwrite ($fp, " | ");
fwrite ($fp, $prezime);
fwrite ($fp, "\n");
fclose ($fp);
?>
```


Upisivanje u datoteku

```
<?php
$ime = $_POST["ime"];
$prezime = $_POST["prezime"];
$fp = fopen ("podaci/ADRESAR.TXT", "a+");
fwrite ($fp, $ime);
fwrite ($fp, " | ");
fwrite ($fp, $prezime);
fwrite ($fp, "\n");
fclose ($fp);
?>
```

Primjer31 – Upisivanje u datoteku

Zatvaranje datoteke

```
<?php
$ime = $_POST["ime"];
$prezime = $_POST["prezime"];
$fp = fopen ("podaci/ADRESAR.TXT", "a+");
fwrite ($fp, $ime);
fwrite ($fp, " | ");
fwrite ($fp, $prezime);
fwrite ($fp, "\n");
fclose ($fp);
?>
```


Čitanje datoteke - određeni broj znakova

```
<?php
$fn = "podaci/ADRESAR.TXT";
$fp = fopen ($fn, "r");
$contents = fread ($fp, filesize ($fn));
fclose ($fp);
echo $contents;
?>
```

čitanje n znakova u string $contents jer se uzela veličina datoteke putem funkcije filesize()

Primjer32 – Čitanje datoteke

Čitanje datoteke - cijela u niz

```
<?php
$fn = "podaci/ADRESAR.TXT";
$fcontents = file ($fn);
while (list ($line_num, $line) = each ($fcontents)) {
    echo "<b>Line $line_num:</b> " . htmlspecialchars ($line) .
"<br>\n";
}
?>
```

Čitanje svih zapisa (redaka) u niz $fcontents

Primjer33 – Čitanje datoteke

Zaključavanje datoteke

Simultano korištenje datototeke od strane 2 ili više skripata koji može dovesti do nekonzistentnosti podataka datoteke ukoliko se ne ograniči pristup samo jednoj skripti u datom vremenu. To znači da ostale skripte trebaju čekati dok prva ne završi svoj rad s datotekom. Zaključavanje datoteke biti će pouzdano rješenje samo u slučaju kada sve skripte traže zaključavanje datoteke. Ukoliko jedna skripta ne koristi taj princip nego direktno pristupa podacima, cijeli koncept gubi smisao jer u slučaju paralelnog rada neće postojati mehanizam međusobnog isključivanja. Ovisi o operacijskom sustavu!

Funkcija zaključavanje ima sintaksu:

```
flock(broj_veze_datoteke, operacija);
```

Uglavnom se koriste dvije operacije zaključavanja:

- **ekskluzivno zaključavanje (LOCK_EX)**
- **otključavanje ranijeg zaključavanja (LOCK_UN)**

Pozicioniranje u datoteci

Kod otvaranja datoteke važeće pozicija datoteke je njen početak. Moguće je realizirati datoteku sa zapisima fiksne duljine kod koje je moguće ažuriranje podataka. Za to je potrebno postaviti važeću poziciju u datoteci na početak zapisa koji se ažurira i nakon toga provesti zapis podataka.
Drugi slučaj kod kojeg je potrebno primjeniti pozicioniranje u datoteci odnosi se na zaključavanje datoteke. Kod poziva zaključavanja datoteke može se čekati dok druga skripta ne završi svoj posao, a ona može promijeniti neke podatke u datoteci kao npr. dodati novi zapis(e). Ako se dodaje na kraj datoteke potrebno je postaviti važeću poziciju u datoteci na njen novi kraj. Ovisi o operacijskom sustavu!
Funkcija pozicioniranja u datoteci ima sintaksu:

```
fseek (broj_veze_datoteke, pomak, [, odakle]);
```

Pomak je broj bajtova, a odakle može biti:
SEEK_SET – početak datoteke
SEEK_CUR - važeća pozicija u datoteci
SEEK_END - kraj datoteke.

Obrada pogreške u radu s datotekama

Rad s datotekama temelji se na određenim pretpostavkama koje mogu biti ispunjenje, ali i ne. Različiti utjecaji uvjetuju da pretpostavke mogu biti pogrešne zbog čega bi izvršavanje skripte krenulo u neželjenom smjeru.
Kao primjer može poslužiti otvaranje datoteke za čitanje koja ne postoji ili nisu ispravno podešena prava za tu operaciju, otvaranje datoteke za upisivanje kada nije dozvoljeno upisivanje, izvršavanje operacije koja nije u skladu s modom otvaranja datoteke i sl.
PHP u slučaju pojavljivanja pogreške u nekoj liniji nastavlja sa sljedećom kao da nije došlo do pogreške.
Potrebno je uključiti obradu pogrešaka kod većine U/I operacija s datotekama.
Za obradu pogreške U/I funkcija povezuje se operatorom `or` s funkcijom za obradu pogreške.
Kada U/I funkcija završi s kodom neuspješnosti tada se aktivira funkcija za obradu pogreške.

```
U/I_funkcija(...) or funkcija_za_obradu_pogreske(...);
```


Obrada pogreške / 1.

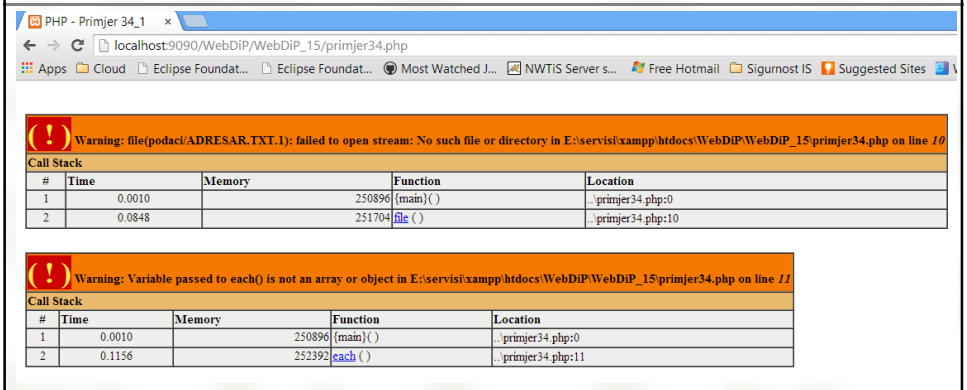
```
<?php
$fn = "podaci/ADRESAR.TXT.1";
$fcontents = file ($fn);
while (list ($line_num, $line) = each ($fcontents)) {
    echo "<b>Line $line_num:</b> " . htmlspecialchars ($line) .
"<br>\n";
}

function pogreska($poruka)
{
    echo "$poruka \n";
    exit;
}
?>
```

Nema obrade pogreške.

Primjer34 – Nema obrada pogreške

Obrada pogreške / 1.



U testnoj varijanti postavki javljaju se upozorenja.

Obrada pogreške / 1.

```
<?php
$fn = "podaci/ADRESAR.TXT.1";
$fcontents = file ($fn) or pogreska("Problem kod otvaranja: $fn");
while (list ($line_num, $line) = each ($fcontents)) {
    echo "<b>Line $line_num:</b> " . htmlspecialchars ($line) .
"<br>\n";
}

function pogreska($poruka)
{
    echo "$poruka \n";
    exit;
}
?>
```

Primjer34_1 – Obrada pogreške

Slanje e-mail poruke

Vrlo često se koristi mogućnost prikupljanja mišljenja putem formulara pri čemu se podaci ne upisuju u datoteke ili baze podataka nego se šalju kao e-mail poruke određenoj osobi-grupi.
S druge strane, moderni oblici poslovanja kao što su elektronička kupovina (amazon.com), elektronička nadmetanja-dražbe (ebay.com) i drugi slični oblici poslovanja primjenjuju informiranje poslovnih partnera putem elektroničke pošte. OPASNOSTI od generiranja neželjene pošte tj. spam-a.

Predložak za funkciju mail:

```
bool mail (string to, string subject, string message
       [, string additional_headers
       [, string additional_parameters]])
```

Da bi se mogla slati email poruka putem funkcije mail(...) potrebno je podesiti u konfiguracijskoj datoteci (php.ini) postavku za SMTP poslužitelja i PORT.
To radi administrator sustava! Kod XAMPP-a postoji Mercury email poslužitelj pa se nakon kreiranja korisnika može osnovnom konfiguracijom PHP-a slati email poruka na Marcury.

Slanje e-mail poruke / 1.

```
<HTML><HEAD>
</HEAD>
<BODY>

<FORM id=form1 method=post name="form1" action="primjer35.php">
<P>
<LABEL for="email">E-mail adresa: </LABEL>
    <INPUT name="email"></BR>
<LABEL for="subjekt">Subjekt: </LABEL>
    <INPUT name="subjekt"></BR>
<LABEL for="tekst">Tekst: </LABEL>
    <TEXTAREA rows="4" cols="50" name="tekst"></TEXTAREA></BR>
<INPUT id=submit1 name=submit1 type=submit value=Šalji>
<INPUT id=reset1 name=reset1 type=reset value=Obriši>
</FORM>

</BODY>
</HTML>
```


Slanje e-mail poruke / 2.

```
<?php
$mail_to = $_POST["email"];
$mail_from = "From: WebDiP@foi.hr";
$mail_subject = $_POST["subjekt"];
$mail_body = $_POST["tekst"];

if(mail($mail_to, $mail_subject, $mail_body, $mail_from))
{
        echo("Poslana poruka za: '$mail_to'!");
} else {
        echo("Problem kod poruke za: '$mail_to'!");
}
?>
```

Primjer35 – Slanje e-mail poruke

Generiranje HTML stranica

Statičke HTML stranice danas ne zadovoljavaju potrebe modernog WWW poslovanja. Sve više postoji ideja za povezivanjem WWW stranica i podataka koji su unešeni preko HTML stranica i pohranjeni u spremnike (datoteke, baze podataka) ili koji već postoje iz drugih aplikacija.
Kao primjer može poslužiti adresar poslovnih partnera. Treba li pružiti mogućnost izbora poslovnog partnera tada više ne dolazi u obzir statička HTML stranica koja bi sadržavala izbornik sa njihovim popisom jer adresar se neprestano dopunjuje. Zbog toga jedino preostaje da se podaci generiraju kod svakog zahtjeva.

Kada se podaci nalaze u datoteci poželjno je obaviti "čišćenje" od nepotrebnih znakova (" ", ", ', <, >,...).
To rade funkcije

```
$varijabla = htmlspecialchars(trim($varijabla));
```


Datoteka za rad

```
podaci/ADRESAR_EMAIL.TXT
pero@localhost
dkermek@foi.hr
pero@foi.hr
grgur@vip.hr
dr_ivo@vlada.hr
suzana@foi.hr
vseks@foi.hr
predsjednik@hr
spavkovi@foi.hr
mato@foi.hr
gradonacelnik@varazdin.hr
```


Generiranje HTML stranica / 1.

```
<table border="1">
<tr><td>Adresa</td><td>Šalji poruku</td></tr>
<?php
$fn = "podaci/ADRESAR.TXT";
$fcontents = file ($fn);
while (list ($line_num, $line) = each ($fcontents)) {
        $value = htmlspecialchars(trim($line));
        print "<tr><td>$value</td><td><a href=\"mailto:$value\">Šalji poruku za:
$value</a></td></tr>\n";
}
?>
</table><br>
```

Primjer36 – Generiranje HTML stranica

Generiranje HTML stranica / 2.

```
<form id=form1 method=post name="form1" action="primjer35.php">
          <label for="email">E-mail adresa: </label>
          <select name="email">
              <?php
              $fn = "podaci/ADRESAR_EMAIL.TXT";
              $fcontents = file($fn);
              while (list ($line_num, $line) = each($fcontents)) {
                  $value = htmlspecialchars(trim($line));
                  print "<option value=\"$value\">$value</option>\n";
              }
              ?>
          </select><br>
          <label for="subjekt">Subjekt: </label>
          <input name="subjekt"><br>
          <label for="tekst">Tekst: </label>
          <textarea rows="4" cols="50" name="tekst"></textarea><><br>
          <input id=submit1 name=submit1 type=submit value=Šalji>
          <input id=reset1 name=reset1 type=reset value=Obriši>
      </form>
```

Primjer36_1 – Generiranje HTML stranica

Datoteka za rad

```
podaci/ADRESAR-VELIKI.TXT
pero@localhost #Pero Kos
dkermek@foi.hr #Dragutin Kermek
grgur@vip.hr #Grgur Ninski
dr_ivo@vlada.hr #Ivo Sanader
dr_ivo@@predsjednik.hr #Ivo Josipović
```

Datoteka je strukturirana tako da svaki zapis sadrži dva podatka (email adresa, prezima i ime) odvojena znakom #.
Kod pripreme podataka za ispis potrebno je odvojiti dijelove zapisa u zasebne podatke.

Generiranje HTML stranica / 3.

```
<form id=form1 method=post name="form1" action="primjer35.php">
                <label for="email">E-mail adresa: </label>
                <select name="email">
                    <?php
                    $fn = "podaci/ADRESAR_VELIKI.TXT";
                    $fcontents = file($fn);
                    while (list ($line_num, $line) = each($fcontents)) {
                        list($adresa, $pime) = explode("#", $line);
                        $adresa = htmlspecialchars(trim($adresa));
                        $pime = htmlspecialchars(trim($pime));
                        print "<option value=\"$adresa\">$pime</option>\n";
                    }
                    ?>
                </select><br>
                <label for="subjekt">Subjekt: </label>
                <input name="subjekt"><br>
                <label for="tekst">Tekst: </label>
                <textarea rows="4" cols="50" name="tekst"></textarea><br>
                <input id=submit1 name=submit1 type=submit value=Šalji>
                <input id=reset1 name=reset1 type=reset value=Obriši>
        </form>
```

Primjer36_2 – Generiranje HTML stranica

Razina izvještaja o pogreškama / 1.

```
int error_reporting ( [int level] )
```

Funkcijom `error_reporting` može se postaviti ili utvrditi važeća razina izvještavanja o pogreškama u skriptama.
Pretpostavljena razina izvještavanja postavljena je varijablom `error_reporting` u konfiguracijskoj datoteci PHP-a – `php.ini`

Razina izvještaja o pogreškama / 2.

```
<?php
if($_SERVER["REQUEST_METHOD"] != "GET") {
  echo "Samo za GET metodu!";
  exit;
}
if(! isset($_GET["vrsta"])) {
  $vrsta = 99;
} else {
  $vrsta = $_GET["vrsta"];
}

echo "Postojeća razina: " . error_reporting() . "<br>";

switch ($vrsta) {
        case 0: $razina = E_ALL;
         break;
        case 1: $razina = E_ALL ^ E_NOTICE;
         break;
        case 2: $razina = E_ERROR | E_WARNING | E_PARSE | E_NOTICE;
         break;
```


Razina izvještaja o pogreškama / 3.

```
        case 3: $razina = E_ERROR | E_WARNING | E_PARSE;
         break;
        case 4: $razina = E_ERROR | E_WARNING;
         break;
        case 5: $razina = E_ERROR;
         break;
        default: $razina = 0;
         break;
}

error_reporting($razina);

echo "Postavljena razina: " . error_reporting() . "<br>";

echo $varijabla;
echo 3/0;
?>
```

Primjer01 – Razina izvještaja o pogreškama

Razina izvještaja o pogreškama / 4.

```
<html>
<head>
<title>PHP - Primjer 1_1</title>
</head>
<body>
<h1>Odabir razine izvještavanja o pogreškama</h1>
<form method="POST" action="primjer01_2.php">
Odaberi elemente:<br>
<input type="checkbox" name="E_ALL" value="1">E_ALL<br>
<input type="checkbox" name="E_ERROR" value="1">E_ERROR<br>
<input type="checkbox" name="E_WARNING" value="1">E_WARNING<br>
<input type="checkbox" name="E_PARSE" value="1">E_PARSE<br>
<input type="checkbox" name="E_NOTICE" value="1">E_NOTICE<br>
<input type="submit" value=" Pošalji ">
</form>
</body>
</html>
```

Primjer01_1 – Postavljenje razine izvještaja o pogreškama

Razina izvještaja o pogreškama / 5.

```
<?php
if($_SERVER["REQUEST_METHOD"] != "POST") {
  echo "Samo za POST metodu!";
  exit;
}

echo "Postojeca razina: " . error_reporting() . "<br>";

$vrste = array("E_ALL", "E_ERROR", "E_WARNING", "E_PARSE", "E_NOTICE");
$razina = 0;
```


Razina izvještaja o pogreškama / 6.

```
foreach($vrste as $i => $v) {
        if(isset($_POST[$v])) {
                echo "i: " . $i . " v: " . $v . "<br>";
                switch ($i) {
                        case 0: $razina |= E_ALL;
                                        break;
                        case 1: $razina |= E_ERROR;
                                        break;
                        case 2: $razina |= E_WARNING;
                                        break;
                        case 3: $razina |= E_PARSE;
                                        break;
                        case 4: $razina |= E_NOTICE;
                                       break;
                }
        }
}
```


Razina izvještaja o pogreškama / 7.

```
error_reporting($razina);

echo "Postavljena razina: " . error_reporting() . "<br>";

echo $varijabla;
echo 3/0;
?>
```

Primjer01_2 – Razina izvještaja o pogreškama

Razina izvještaja o pogreškama / 8.

Možemo željenu razinu izvještavanja o pogreškama spremiti u datoteku kao postavku i kasnije učitavati u radu. Može biti sastavni dio vlastitog programskog okvira (framework-a). O tome kasnije.

```
<html><head>
<title>PHP - Primjer 1_3</title>
</head>
<body>
<h1>Odabir razine izvještavanja o pogreškama</h1>
<form method="POST" action="primjer01_4.php">
Odaberi elemente:<br>
<input type="checkbox" name="E_ALL" value="1">E_ALL<br>
<input type="checkbox" name="E_ERROR" value="1">E_ERROR<br>
<input type="checkbox" name="E_WARNING" value="1">E_WARNING<br>
<input type="checkbox" name="E_PARSE" value="1">E_PARSE<br>
<input type="checkbox" name="E_NOTICE" value="1">E_NOTICE<br>
<input type="submit" value=" Pošalji ">
</form></body></html>
```

Primjer01_3 – Postavljenje razine izvještaja o pogreškama

Razina izvještaja o pogreškama / 9.

```
<?php
if($_SERVER["REQUEST_METHOD"] != "POST") {
  echo "Samo za POST metodu!";
  exit;
}
$vrste = array("E_ALL", "E_ERROR", "E_WARNING", "E_PARSE", "E_NOTICE");
$razina = 0;
foreach($vrste as $i => $v) {
       if(isset($_POST[$v])) {
              echo "i: " . $i . " v: " . $v . "<br>";
              switch ($i) {
                     case 0: $razina |= E_ALL;     break;
                     case 1: $razina |= E_ERROR;   break;
                     case 2: $razina |= E_WARNING; break;
                     case 3: $razina |= E_PARSE;   break;
                     case 4: $razina |= E_NOTICE;  break;
              }
       }
}
```


Razina izvještaja o pogreškama / 10.

```
$zapis = '<?php $razina = ' . $razina . ';' . "\n";
$zapis .= 'error_reporting($razina);' . "\n";
$zapis .= '?>' . "\n";

$fp = fopen("postavke.php", "w");
fwrite($fp, $zapis);
fclose($fp);
?>

</body>
</html>
```

generiramo php programski kod

Primjer01_4 – Spremanje postavke za razinu izvještaja o pogreškama

Razina izvještaja o pogreškama / 11.

```
<html>
<head>
<title>PHP - Primjer 1_5</title>
</head>
<body>
<h1>Korištenje postavljene razine izvještavanja o pogreškama</h1>
<?php
include_once 'postavke.php';

echo "Postavljena razina: " . error_reporting() . "<br>";

echo $varijabla;
echo 3/0;
?>

</body>
</html>
```

Primjer01_5 – Korištenje postavljene razine izvještaja o pogreškama

Rukovanje s pogreškama / 1.

```
mixed set_error_handler ( callback error_handler [, int
error_types] )
```

Funkcijom `set_error_handler` može se preusmjeriti izvržavanje skripte na definiranu funkciju za rukovanje s pogreškama u slučaju određene razine pogreške u skriptama.
Pretpostavljena razina izvještavanja postavljena je varijablom error_reporting u konfiguracijskoj datoteci PHP-a – php.ini
Može biti sastavni dio vlastitog programskog okvira (framework-a).

Rukovanje s pogreškama / 2.

Funkcija za rukovanje s pogreškama ima sljedeći format:

```
handler ( int errno, string errstr [, string errfile [,
int errline [, array errcontext]]] )
```

`errno` – razina pogreška koja se desila
`errstr` – poruka pogreške
`errfile` – naziv datoteke u kojoj se desila pogreška
`errline` – broj linije u kojoj se desila pogreška
`errcontext` – vektor varijabli koje postoje u pogledu dešavanja pogreške.
NE smiju se mijenjati!

Rukovanje s pogreškama / 3.

```
<?php
include_once 'postavke.php';
function obradaPogresaka($errno, $errstr, $errfile, $errline,
$errcontext) {
  echo "Desila se pogreška kod izvršavanja!<br>";
  echo "Datoteka: $errfile<br>";
  echo "Linija: $errline<br>";
  echo "Opis: $errstr<br>";
  echo "Kod: $errno<br>";
  exit;
}
if($_SERVER["REQUEST_METHOD"] != "GET") {
  echo "Samo za GET metodu!";
  exit;
}
echo "Postavljena razina: " . error_reporting() . "<br>";
set_error_handler('obradaPogresaka');
echo $varijabla;
echo 3/0;
```

Primjer02_1 – Rukovanje s pogreškama

Ignoriranje pogrešaka / 1.

Operatorom @ može se ignorirati pogreška u izrazu koji slijedi operator

```
<?php

@$varijabla = 3/0;

?>
```

Primjer03 – Ignoriranje pogrešaka

Generiranje pogrešaka / 1.

```
bool trigger_error( string error_msg [, int error_type] )
```

Funkcijom `trigger_error` može se generirati pogreška željene poruke i razine te preusmjeriti izvržavanje skripte na postavljenog rukovatelja pogrešaka.
Može biti sastavni dio vlastitog programskog okvira (framework-a).

Generiranje pogrešaka / 2.

```
if($_SERVER["REQUEST_METHOD"] != "GET") {
  echo "Samo za GET metodu!";
  exit;
}
if(! isset($_GET["a"])) {
  $a = 5;
} else {
  $a = $_GET["a"];
}
if(! isset($_GET["b"])) {
  $b = 1;
} else {
  $b = $_GET["b"];
}
if($b == 0) {
  trigger_error("Djeljenje s nulom!", E_USER_ERROR);
}
echo "Rezultat: " . $a / $b;
```

E_USER_ERROR
E_USER_WARNING
E_USER_NOTICE

Primjer04 – Generiranje pogrešaka

Obrada pogrešaka – cjeloviti pristup

```php
<?php
include_once 'postavke.php';

set_error_handler('obradaPogresaka');

function obradaPogresaka($errno, $errstr, $errfile, $errline,
$errcontext) {
  echo "Desila se pogreška kod izvršavanja!<br>";
  echo "Datoteka: $errfile<br>";
  echo "Linija: $errline<br>";
  echo "Opis: $errstr<br>";
  echo "Kod: $errno<br>";
  exit;
}
?>
```

obradaPogresaka – Obrada pogrešaka - cjelovit pristup

Obrada pogrešaka – cjeloviti pristup / 2.

```php
include_once 'obradaPogresaka.php';
if($_SERVER["REQUEST_METHOD"] != "GET") {
  echo "Samo za GET metodu!";
  exit;
}
if(! isset($_GET["a"])) {
  $a = 5;
} else {
  $a = $_GET["a"];
}
if(! isset($_GET["b"])) {
  $b = 1;
} else {
  $b = $_GET["b"];
}
if($b == 0) {
  trigger_error("Djeljenje s nulom!", E_USER_ERROR);
}
echo "Rezultat: " . $a / $b;
```

Primjer04_1 – Generiranje pogrešaka

Evidencija pogrešaka / 1.

```
bool error_log ( string message [, int message_type [, string
destination [, string extra_headers]]]
```

Funkcijom `error_log` može se evidentirati pogreška u dnevniku web poslužitelja, poslati na TCP port ili u vlastitu datoteku.
Vrste:
0 – poruka se šalje PHP sistemskom dnevniku
1 – poruka se šalje kao email na adresu
2 - poruka se šalje na vezu PHP debugera
3 – poruka se dodaje na kraj datoteke

Evidencija pogrešaka / 2.

```
if($_SERVER["REQUEST_METHOD"] != "GET") {
  echo "Samo za GET metodu!";
  exit;
}
if(! isset($_GET["a"])) {
  $a = 5;
} else {
  $a = $_GET["a"];
}
if(! isset($_GET["b"])) {
  $b = 1;
} else {
  $b = $_GET["b"];
}
```


Evidencija pogrešaka / 2.

```
if($b == 0) {
  error_log("Djeljenje s nulom!");

  error_log(__FILE__ . " " . __LINE__ . " Djeljenje s nulom! " .
       date("d.m.Y H:i:s") . "\n", 3, "dnevnik_pogresaka.txt");


  error_log(" Djeljenje s nulom! " . date("d.m.Y H:i:s") . "\n",
       1, "pero@localhost", "Subject: Pero s WebDiP-a");

  exit;
}
echo "Rezultat: " . $a / $b
```

PHP sistemski dnevnik

aplikacijski dnevnik

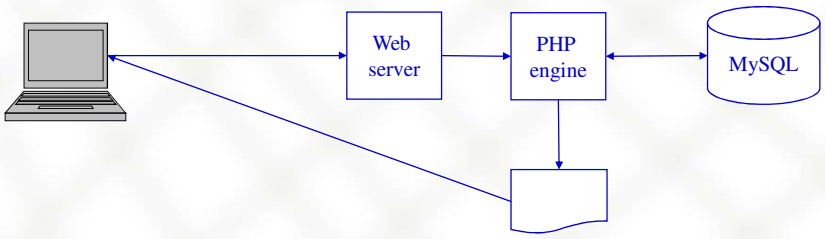
email adresa

Primjer05 – Evidencija pogrešaka

RAD S BAZAMA PODATAKA

Složeni projekti uglavnom su temeljeni na primjeni baza podataka. U posljednje vrijeme kada se spominje ozbiljna poslovna primjena WWW-a tada on nije iznimka od nepisanog pravila primjene baza podataka.
Otvorenost WWW donosi znatno više problema na području sigurnosti i rada na zaštiti baza podataka.

Na predmetu "Web dizajn i programiranje" koristi se baza podataka MySQL kojoj se pristupa iz PHP-a.



Korištenje MySQL-a

Postoje 3 načina korištenje baze podataka MySQL:

-mysql - od PHP 5.5.0 označeno kao "deprecated" pa treba izbjegavati

-mysqli - poboljšana verzija, objektno orijentirana, preporučuje se za korištenje u novim projektima

-PDO MySQL - PHP Data Objects - apstraktni sloj za bazu podataka (API) koji je namijenjen za PHP aplikacije a cilj je da se može mijenjati baza podataka uz minimalnu promjenu u programskom kodu

Korištenje MySQL-a

Poboljšanja koje donosi mysqli:

- **objektno orijentirano sučelje**
- **podrška za pripremljene instrukcije**
- **podrška za višestruke instrukcije (više SQL instrukcija u jednoj instrukcija, a odvojene su s ';')**
- **podrška za transakcije**
- **dopunjenje mogućnosti debug-a**
- **podrška za ugrađeni poslužitelj.**

Može se koristi proceduralni i objektno orijentirani način.

Proceduralno korištenje MySQL-a

Prvi korak je otvaranja veze prema poslužitelju baze podataka:

```
mysqli mysqli_connect ( [string host [, string username [,
string passwd [, string dbname [, int port [, string
socket]]]]]] )
```

Svi paremetri su opcionalni tako da se uzimaju pretpostavljene vrijednosti, a to su "localhost" za server, username se preuzima od vlasnika procesa i password je prazan, osnovna baza podataka, port 3306.

Drugi korak je obično postavljenje skupa znakova (charset):

```
mysqli_set_charset(veza, znakovi);
```


Izvršavanje operacija na BP kod MySQL-a

Operacije se mogu izvršiti na nekoliko načina pri čemu izbor ovisi o operaciji i načinu rada s dobivenim podacima.

```
int mysqli_query (int link_identifier, string query)
```

```
mysqli_fetch_row(...),
mysqli_fetch_array(...),
mysqli_fetch_object(...).
```


PHP funkcije kod proceduralnog korištenja

mysqli_affected_rows	- broj redaka koji su obuhvaćeni prethodnom operacijom
mysqli_change_user	- promjena korisnika aktivne veze
mysqli_close	- zatvaranje veze
mysqli_connect	- otvaranje veze prema MySQL poslužitelju
mysqli_create_db	- kreiranje baze podataka
mysqli_data_seek	- pomak internog pokazivača rezultata
mysqli_db_query	- slanje SQL upita
mysqli_drop_db	- brisanje baze podataka
mysqli_errno	- vraća vrijednost poruke pogreške prethodne operacije
mysqli_error	- vraća tekst poruke pogreške prethodne operacije
mysqli_fetch_array	- preuzima redak rezultata kao asocijativno polje
mysqli_fetch_field	- vraća informaciju o stupcu iz rezultata kao objekt
mysqli_list_tables	- lista tablica u MySQL DMBS-u
mysqli_num_fields	- vraća broj polja u rezultatu
mysqli_num_rows	- vraća broj redaka u rezultatu
mysqli_result	- vraća rezultat
mysqli_tablename	- vraća naziv tablice

Izbjegavanje čvrstih veza u kodu

Izbjegavanje čvrstih podataka u programskom kodu najbolje se postiže uvođenjem datoteke s parametarskim podacima.

```
<?php
$server = 'localhost';
$korisnik = 'WebDiP';
$lozinka = 'FOI';
$baza = 'PzaWeb';
$znakovi = 'utf8';
?>
```

baza_podataka – Parametarski podaci za rad s bazom podataka

SQL SELECT kod MySQL-a

```
<?
require("bazaPodataka.php");
$dbc = mysqli_connect($server, $korisnik, $lozinka, $baza);
if(! $dbc) {
        echo "Problem kod povezivanja na bazu podataka!";
        exit;
}
mysqli_set_charset($dbc, $znakovi);

$sql = "select prezime, ime, maticni_broj FROM POLAZNICI " .
        "order by prezime, ime";

$rs = mysqli_query($dbc, $sql);
if(! $rs) {
        echo "Problem kod upita na bazu podataka!";
        exit;
}
```


SQL SELECT kod MySQL-a / 1.

```
 print "<table border=1><tr><td>Prezime</td><td>Ime</td>" .
        "<td>Maticni_broj</td></tr>\n";

        while (list($prezime, $ime, $maticni_broj) =
                        mysqli_fetch_array($rs)) {
            print "<tr><td>$prezime</td><td>$ime</td>" .
                "<td>$maticni_broj</td></tr>\n";
        }
        print "</table>\n";

mysqli_close($dbc);
?>
</table>
```

Primjer42 – SQL SELECT kod PK MySQL-a

SQL INSERT kod MySQL-a

```
<?
require("bazaPodataka.php");
$dbc = mysqli_connect($server, $korisnik, $lozinka, $baza);
if(! $dbc) {
        echo "Problem kod povezivanja na bazu podataka!";
        exit;
}

mysqli_set_charset($dbc, $znakovi);

$sql = "insert into POLAZNICI " .
        "(maticni_broj, prezime, ime, lozinka, email_adresa) " .
        "values ('00001/00-R', 'Kermek', 'Dragutin', '123456',
        'dkermek@foi.hr')";

$rs = mysqli_query($dbc, $sql);
```


SQL INSERT kod MySQL-a / 1.

```
if(! $rs) {
        echo "Problem kod upisa u bazu podataka!<br>";
        echo mysqli_error($dbc);
        exit;
}

mysqli_close($dbc);
?>
```

Primjer42_1 – SQL INSERT kod MySQL-a

OO korištenje MySQL-a

Prvi korak je otvaranja veze prema poslužitelju baze podataka pomoću objekta klase mysqli:

```
new mysqli ( [string host [, string username [, string passwd
[, string dbname [, int port [, string socket]]]]]] )
```

Svi paremetri su opcionalni tako da se uzimaju pretpostavljene vrijednosti, a to su "localhost" za server, username se preuzima od vlasnika procesa i password je prazan, osnovna baza podataka, port 3306.

Kod proceduralnog korištenja kod kojeg se treba u funkcije prenositi veza prema bazi podataka.
Kod OO korištenja koristi se objekt klase mysqli te se na njemu izvršavaju funkcije.

PHP funkcije kod OO korištenja

Funkcije su uglavnom iste kao i kod proceduralnog korištenja samo što nemaju prefix mysqli_ i argument veze na bazu podataka. One se pozivaju nad objektom klase mysqli uz operator ->.

Npr.

```
mysqli_close($dbc);
```

postaje

```
$mysqli->close();
```


SQL SELECT kod MySQL-a

```
<?
$mysqli = new mysqli($server, $korisnik, $lozinka, $baza);
 if($mysqli->connect_errno) {
        echo "Problem kod povezivanja na bazu podataka! " .
                $mysqli->connect_error;
        exit;
}

$mysqli->set_charset($znakovi);

$sql = "select prezime, ime, maticni_broj FROM POLAZNICI " .
        "order by prezime, ime";

$rs = $mysqli->query($sql);

if(! $result) {
        echo "Problem kod upita na bazu podataka!";
        exit;
}
```


SQL SELECT kod MySQL-a

```
print "<table><tr><td>Prezime</td><td>Ime</td>
                <td>Maticni_broj</td></tr>\n";
while( list($prezime, $ime, $maticni_broj) = $rs->fetch_array()) {
        print "<tr><td>$prezime</td><td>$ime</td>
                <td>$maticni_broj</td></tr>\n";
}
$rs->close();
$mysqli->close();
```

Primjer42 – SQL SELECT kod OOK MySQLI-a

Vlastita klasa za rad s bazom podataka

```
<?php

class Baza {

    const server = "localhost";
    const korisnik = "WebDiP";
    const lozinka = "FOI";
    const baza = "PzaWeb";

    private $veza = null;
    private $greska = '';
```


Vlastita klasa za rad s bazom podataka

```
    function spojiDB() {
        $this->veza = new mysqli(self::server, self::korisnik,
                        self::lozinka, self::baza);
        if ($this->veza->connect_errno) {
            echo "Neuspješno spajanje na bazu: " .
               $this->veza->connect_errno . ", " .
            $this->veza->connect_error;
            $this->greska = $this->veza->connect_error;
        }
        $this->veza->set_charset("utf8");
        if ($this->veza->connect_errno) {
            echo "Neuspješno postavljanje znakova za bazu: " .
               $this->veza->connect_errno . ", " .
            $this->veza->connect_error;
            $this->greska = $this->veza->connect_error;
        }
        return $this->veza;
    }
```


Vlastita klasa za rad s bazom podataka

```
function selectDB($upit) {
    $rezultat = $this->veza->query($upit);
    if ($this->veza->connect_errno) {
        echo "Greška kod upita: {$upit} - " .
           $this->veza->connect_errno . ", " .
        $this->veza->connect_error;
        $this->greska = $this->veza->connect_error;
    }
    if (!$rezultat) {
        $rezultat = null;
    }
    return $rezultat;
}
```


Vlastita klasa za rad s bazom podataka

```
function updateDB($upit, $skripta = '') {
    $rezultat = $this->veza->query($upit);
    if ($this->veza->connect_errno) {
        echo "Greška kod upita: {$upit} - " .
           $this->veza->connect_errno . ", " .
        $this->veza->connect_error;
        $this->greska = $this->veza->connect_error;
    } else {
        if ($skripta != '') {
            header("Location: $skripta");
        }
    }

    return $rezultat;
}
```


Vlastita klasa za rad s bazom podataka

```
    function pogreskaDB() {
        if ($this->greska != '') {
            return true;
        } else {
            return false;
        }
    }

}

?>
```

baza.class.php– Vlastita klasa za rad s BP

Vlastita klasa za rad s bazom podataka

```
<?php
        require("baza.class.php");
        $bp = new Baza();
        $sql = "select prezime, ime, maticni_broj FROM POLAZNICI " .
                "order by prezime, ime";

        $bp->spojiDB();
        $rs = $bp->selectDB($sql);

        if ($bp->pogreskaDB()) {
            echo "Problem kod upita na bazu podataka!";
            exit;
        }
```


Vlastita klasa za rad s bazom podataka

```
        print "<table border=1>
               <tr><td>Prezime</td><td>Ime</td>
                   <td>Maticni_broj</td></tr>\n";

        while (list($prezime, $ime, $maticni_broj) =
                                $rs->fetch_array()) {
            print "<tr><td>$prezime</td><td>$ime</td>
               <td>$maticni_broj</td></tr>\n";
        }
        print "</table>\n";

        $rs->close();
        $bp->zatvoriDB();
        ?>
    </table>
```

Primjer42_1 – Vlastita klasa za rad s BP

Otvaranje veze prema poslužitelju PostgreSQL-a

Prvi korak je otvaranja veze prema poslužitelju baze podataka s izborom baze podataka:

```
resource pg_connect ( string connection_string [, int
connect_type] )
```

Connection_string sadrži elemente/parametre za uspostavljanje veze:
`"host= port= dbname= user= password="`

Svi paremetri su opcionalni tako da se uzimaju pretpostavljene vrijednosti. Potrebno je u konfiguraciji PHP-a (php.ini) dopustiti korištenje PostgreSQL-a.

Izvršavanje operacija na BP kod MySQL-a

Operacije se mogu izvršiti na nekoliko načina pri čemu izbor ovisi o operaciji i načinu rada s dobivenim podacima.

```
int pg_query (resource connection, string query )
```

```
pg_fetch_row(),
pg_fetch_array(),
pg_fetch_object().
```


SQL SELECT kod PostgreSQL-a

```
<?
$dbc = pg_connect("host=localhost port=5432 dbname=PzaWeb
user=PzaWeb2002 password=FOI");
if(! $dbc) {
        echo "Problem kod povezivanja na bazu podataka!";
        exit;
}
$sql = "select prezime, ime, maticni_broj FROM POLAZNICI " .
        "order by prezime, ime";
$rs = pg_query($dbc, $sql);
if(! $rs) {
        echo "Problem kod upita na bazu podataka!";
        exit;
}
print "<TABLE><TR><TD>Prezime<TD>Ime<TD>Maticni_broj\n";
while( list($prezime, $ime, $maticni_broj) = pg_fetch_array($rs)) {
        print "<TR><TD>$prezime<TD>$ime<TD>$maticni_broj\n";
}
pg_close($dbc);
```

Primjer42_3 – SQL SELECT kod PostgeSQL-a

Kreiranje baze podataka - MySQL

Za kreiranje baze podataka treba imati dozvolu za tu vrstu operacije.

- kreiranje baze podataka

```
CREATE DATABASE FOI_PzaWeb;
```

- kreiranje korisnika s lozinkom

```
CREATE USER FOI_PzaWeb IDENTIFIED BY 'foi2007';
```

- pridruživanje dozvola korisniku za rad s bazom podataka

```
GRANT DELETE, INSERT, SELECT, UPDATE
  ON `FOI_PzaWeb`.*
  TO 'PzaWeb2007'@'localhost' IDENTIFIED BY 'foi2007';
```

- mogućnost korištenja stare verzije lozinke (bez kripriranja)

```
SET PASSWORD FOR 'FOI_PzaWeb'@'localhost' = OLD_PASSWORD('foi2007');
```


Kreiranje tablice - MySQL

Za kreiranje tablice u bazi podataka treba imati dozvolu za tu vrstu operacije.

```
CREATE TABLE POLAZNICI (
       maticni_broj CHAR(10) NOT NULL,
       prezime CHAR(25) NOT NULL,
       ime CHAR(25) NOT NULL,
       lozinka CHAR(20) NOT NULL,
       email_adresa CHAR(40) NOT NULL,
       datum_kreiranja datetime NOT NULL,
       datum_promjene datetime NOT NULL,
       PRIMARY KEY (maticni_broj),
       INDEX prezime_ime (prezime, ime)
);
```


SQL

Skup naredbi za upravljenje i rad s podacima baze podataka.

DDL - Data Definition Language (DDL)
DML - Data Manipulation Language (DML)
DCL - Data Control Language (DCL)
TCL - Transaction Control (TCL)

SQL - DDL

Data Definition Language (DDL) – za definiranje strukture baze statements ili sheme.

CREATE
ALTER
DROP
TRUNCATE
COMMENT
RENAME

SQL - DML

Data Manipulation Language (DML) – za upravljenje/manipulaciju s podacima u shemi.

```
SELECT
INSERT
UPDATE
DELETE
```


SQL - DCL

Data Control Language (DCL) – za postavljenje dozvola.

```
GRANT
REVOKE
```


SQL - TCL

Transaction Control (TCL) – za upravljenje promjenama koje DML proizvode, formiranje grupa naredbi - transakcije.

```
COMMIT
SAVEPOINT
ROLLBACK
SET TRANSACTION
```


SQL DCL – MySQL 1.

Dozvola	Značenje
ALL [PRIVILEGES]	Sve dozvole osim GRANT OPTION
ALTER	Dozvola za ALTER TABLE
ALTER ROUTINE	Dozvola promjene ili brisanja pohranjenih procedura
CREATE	Dozvola za CREATE TABLE
CREATE ROUTINE	Dozvola kreiranja pohranjenih procedura
CREATE TEMPORARY TABLES	Dozvola za CREATE TEMPORARY TABLE
CREATE USER	Dozvola za CREATE USER, DROP USER, RENAME USER, and REVOKE ALL PRIVILEGES.
CREATE VIEW	Dozvola za CREATE VIEW
DELETE	Dozvola za DELETE
DROP	Dozvola za DROP TABLE
EVENT	Dozvola kreiranja događaja za upravljača događaja
EXECUTE	Dozvola korisniku da izvršava pohranjene procedure
FILE	Dozvola za SELECT ... INTO OUTFILE i LOAD DATA INFILE
INDEX	Dozvola za CREATE INDEX i DROP INDEX
INSERT	Dozvola za INSERT

SQL DCL – MySQL 2.

Privilege	Meaning
LOCK TABLES	Dozvola za LOCK TABLES za tablice za koje korisnik ima dozvolu SELECT
PROCESS	Dozvola da korisnik vidi pohranjenje procedura s SHOW PROCESSLIST
REFERENCES	Nije implementirano
RELOAD	Dozvola za FLUSH
REPLICATION CLIENT	Omogućava korisniku da pita gdje su glavni i pomoćni serveri
REPLICATION SLAVE	Potrebno za replikaciju pomoćnog servera
SELECT	Dozvola za SELECT
SHOW DATABASES	SHOW DATABASES prikazuje sve baze podataka
SHOW VIEW	Dozvola za SHOW CREATE VIEW
SHUTDOWN	Dozvola za **mysqladmin shutdown**
SUPER	Dozvola za CHANGE MASTER, KILL, PURGE MASTER LOGS, i SET GLOBAL, the **mysqladmin debug** komande, itd
TRIGGER	Omogućava korisniku da kreira ili obriše okidače
UPDATE	Dozvola za UPDATE
USAGE	Sinonim za "bez privilegija"
GRANT OPTION	Omogućava pridruživanje dozvola

Imenički servisi – LDAP

LDAP definira kako klijenti trebaju pristupiti do podataka na poslužitelju. On ne određuje kako podaci trebaju biti spremljeni na poslužitelju.

LDAP je organiziran u stablo pod nazivom DIT – Director Information Tree. Svaki list u DIT-u naziva se element (entry). Prvi element u DIT-u naziva se korijenski element .

Element se sastoji od DN – Distinguished Name i bilo kojeg broja parova atribut/vrijednost. DN je naziv elementa i mora biti jednoznačan.

LDAP poslužitelji podržavaju uputnice tj. kazaljke na druge LDAP direktorije tako da jedan LDAP poslužitelj može pretražiti milijun elemenata temeljem jednog zahtjeva korisnika.

LDAP – CMU – AAI@EduHr

Imenički servisi kao što je LDAP sve više se koriste zbog centralizacije autentifikacijskih podataka, a time i uklanjanja redundancije podataka za lozinku. Tada se koriste kao dodatak bazi podataka koja u tablici sadrži skup korisnika kojima se dozvoljava rad, a može se i u LDAP-u uvesti atribut koji sadrži domene za koje korisnik ima pristup.

Prednost LDAP je što omogućava kreiranje vlastite sheme (atributi, ponavljanje atributa i sl.) a ujedno je podržana autentikacija i autorizacija – na razini atributa. LDAP ima ugrađen vrlo snažan sigurnosni model korištenjem ACL za zaštitu podataka na poslužitelju i podržava SSL – Secure Socket Layer protokol.

Za CMU i FOI Moodle koristi se AAI@EduHr, shema na http://schema.aaiedu.hr/shema/

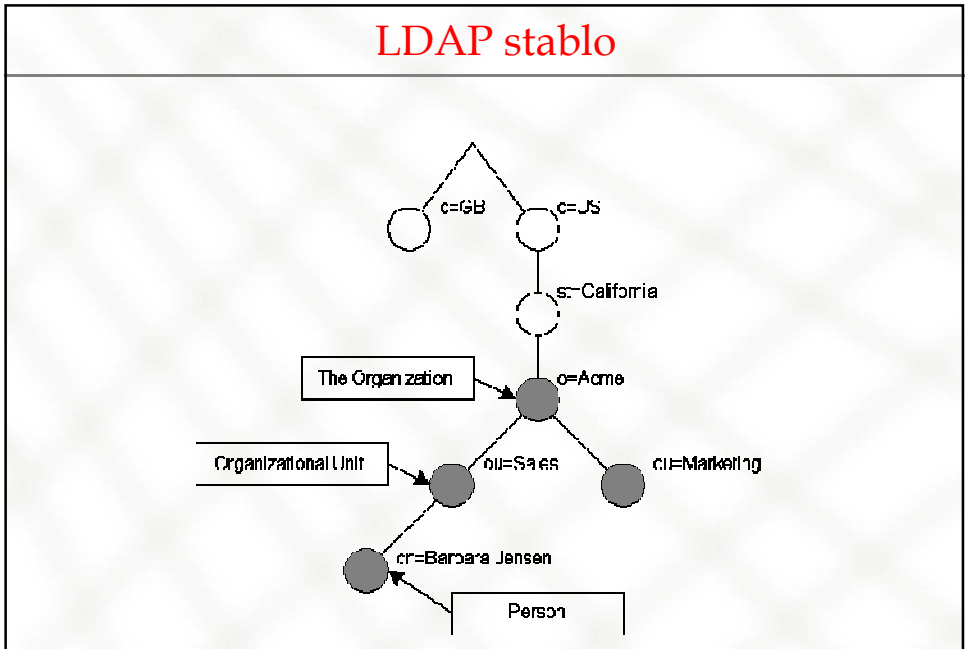
Potrebno je u konfiguraciji PHP-a (php.ini) dopustiti korištenje LDAP-a.

LDAP klase

Klasa	Roditelj	Zahtjevani atribut(i)
top	nema	ObjectClass
country	top	c
locality	top	nema
organization	top	o
organizaitionalUnit	top	ou
person	top	sn, cn
organizationalPerson	top	nema

LDAP atributi

Atribut	Značenje
objectClass	Naziv klase objekta i njegovih superklasa
dc	domain context – dio naziva domene
cn	common name
sn	surname - prezime
c	kod države (2 znaka)
l	lokalno
st	područje
o	organizacija
mail	e-mail adresa



LDAP stablo

LDAP – pretraživanje, listanje i ispitivanje

LDAP imenički servisi omogućava pretraživanje po raznim kriterijima te izlistavanje izabranih atributa za zadovoljene element. Može se ispitivati vrijednost izabranog atributa i sl.

```
$ds=ldap_connect("barok.foi.hr");
$r=ldap_bind($ds);
$sr=ldap_search($ds, "dc=foi,dc=hr", "sn=K*");
$nr = ldap_count_entries($ds, $sr)
$info = ldap_get_entries($ds, $sr);
for ($i=0; $i < $info["count"]; $i++) {
        echo $info["count"]["dn"];
}
ldap_close($ds);
```

Primjer01 – LDAP listanje 1 Primjer02 – LDAP listanje 2

Primjer03 – LDAP ispitivanje atributa

Korištena i dodatna literatura

- http://www.sitepoint.com/books/Kevs-php-mysql.pdf
- http://www.devshed.com/Server_Side/PHP/DB_Basics/DB_Basics.pdf
- http://www.devshed.com/Server_Side/PHP/PHPMySQLPublishing/PHPMySQLPublishing.pdf