Uporabniška programska oprema
2. zvezek

Andrej Brodnik, Iztok Savnik,
Tamara Bertok, Matjaž Kljun

Študijsko gradivo v elektronski obliki

Univerza na Primorskem, Pedagoška fakulteta Koper

2007

Andrej Brodnik, Iztok Savnik, Tamara Bertok, Matjaž Kljun

Uporabniška programska oprema – 2. zvezek

CIP - Kataložni zapis o publikaciji
Narodna in univerzitetna knjižnica, Ljubljana

004.4(075.8)

UPORABNIŠKA programska oprema [Elektronski vir] :
študijsko gradivo v elektronski obliki / Andrej Brodnik ... [et al.]. –
Koper : Pedagoška fakulteta, 2007

Način dostopa (URL): http://www.pef.upr.si/gradiva

ISBN 978-961-6528-64-1 (zv. 2)

1. Brodnik, Andrej

231430144

Kazalo

Predavanja

Vaje

Uporabniška programska oprema

2006/2007

Iztok Savnik

Sistemska analiza

- V naslednjem delu predmeta se bomo ukvarjali s sistemi
- Sisteme bomo analizirali, poiskali njihov ustroj, pravila delovanja
- Na koncu bomo izgradili nove sisteme, ki bodo po funkcijskih specifikacijah enaki prvotnim

Informacijski sistemi

- S kakšnimi sistemi se bomo ukvarjali?
- S sistemi ***informacij*** ali z ***informacijskimi sistemi***
- Kaj je sistem?
- Sistem je zaključena celota, ki sicer lahko sodeluje z okolico
  - prim. fizikalni sistemi

Sistem in entitete

- Sistem vključuje tako fizične elemente kot podatke o le-teh
- Nas bodo zanimali samo ***podatki*** o elementih sistema
- Elementom sistema pravimo tudi ***entitete***
- Pri predmetno usmerjenem programiranju bi entitetam skoraj lahko rekli predmeti
- Tako kot predmeti, imajo tudi entitete ***lastnosti***

Entitete

- Primer entitete je:
  - video kaseta, član video kluba
- Nas ne zanima sama kaseta, ampak podatki o njej:
  - naslov, žanr, trajanje, ...
- Podobno nas ne zanima član, ampak podatki o njem
- Entitete niso neodvisne – tako si član lahko ***sposodi*** video kaseto

Procesi

- Poleg entitet, ki so pasivni elementi sistema, nastopajo še aktivni elementi – procesi
- Vloga procesov je preoblikovanje vrednosti lastnosti entitet
  - v bistvu so podobni metodam oziroma funkcijam

Končni rezultat

- Namen informacijske analize je zadovoljivo opisati:
  - procese v sistemu, da bomo iz njih lahko naredili funkcije, metode – napisali programe
  - entitete v sistemu, da bomo lahko definirali tip (razrede) za hranjenje podatkov – definirali podatkovno bazo

Primer

- Na predavanjih si bomo ogledali primer *videoteke* in vodenja le-te
- Na vajah boste delali vsak svoj majhen projekt in vsi projekti skupaj se bodo sestavili v del spletne predstavitve (portal)
  - pošiljanje SMS sporočil
  - upravljanje sistema za pošiljanje SMS sporočil

Informacijska analiza

- Sestoji iz naslednjih korakov:
  1. zajem podatkov – anketa
  2. izdelavo toka fizičnih podatkov
  3. izdelava toka logičnih podatkov (DTP – diagram toka podatkov, *DFD – Data Flow Diagram*)
  4. izdelava podatkovnega modela
  5. izvedba – definicija tabel in minispecifikacij

Informacijska analiza

- Sami informacijski analizi sledi še cenovna analiza večih možnih izvedb sistema
- Na koncu se naročnik odloči, katero od izvedb bo uporabil in v resnici naredil
- Mi si bomo pogledali samo prvih pet korakov analize

Zajem podatkov

- Pri zajemu podatkov moramo najprej natančno definirati kaj je naš sistem, kakšen je njegov namen in kako se uporablja
  - videoteka, v kateri si člani sposojajo oziroma vračajo posamezne video kasete
  - videoteka pošilja članom, ki zamujajo z vračilom opomine
  - poleg tega videoteka nabavlja nove video kasete
  - naš sistem ne vključuje denarnega poslovanja

Zajem podatkov

- Z zajemom podatkov moramo definirati namen in funkcije našega sistema
- Definirati moramo tudi kakšni podatki prihajajo v sistem in kakšni iz sistema
- prim. s pogodbo pri definiciji funkcij:
  - člen opis – funkcionalni opis sistema
  - členi parametri, rezultat in okolje – kateri podatki prihajajo v sistem in kateri iz njega

Diagram nivoja 0

- Na tem nivoju imamo samo en proces – celoten sistem
- Ob tem navedemo še vse podatke (entitete), ki vstopajo v ali izstopajo iz sistema
  - tem entitetam pravimo ***zunanje entitete***, ker bivajo izven našega sistema in na njih (načeloma) nimamo vpliva
  - zunanje entitete imajo fizično obliko obrazcev, pisem, e-pošt ipd.

Diagram nivoja 0

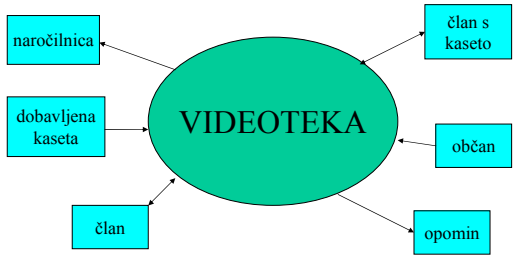


Diagram nivoja 0

- Za vsako od zunanjih entitet zapišemo spisek lastnosti
  - občan: ime, priimek, naslov, telefon
    - le tiste lastnosti, ki so pomembne za naš sistem
  - član: članska številka, ime, priimek
    - v bistvu je to članska izkaznica
  - član s kaseto: članska številka, številka kasete
    - član, ki si je sposodil kaseto

Diagram nivoja 0

- naročilnica: številka naročilnice, spisek naslovov naročenih kaset
  - naročilnica se pošlje v trgovino
  - pozor, imena trgovine ni tu, torej obstaja še nek dodatni proces, ki bo dopisal naslov, recimo vložišče, a to je ***izven*** našega sistema
- dobavljena kaseta: naslov kasete, številka naročilnice, po kateri je dobavljena, ali je dobavljena
  - v primeru, da kasete ni moč dobaviti moramo dobiti obvestilo o tem

Predavanje: 14 UPO 2005/06 16
© A.Brodnik, I.Savnik

Uporabniška programska oprema

2006/2007

Iztok Savnik

Informacijska analiza

- Sestoji iz naslednjih korakov:
  1. zajem podatkov – anketa
  2. izdelavo toka fizičnih podatkov
  3. izdelava toka logičnih podatkov (DTP – diagram toka podatkov, *DFD – Data Flow Diagram*)
  4. izdelava podatkovnega modela
  5. izvedba – definicija tabel in minispecifikacij

Zajem podatkov

- Postopek zajema podatkov nam mora:
  - podati sliko o uporabnosti sistema - izdelati moramo funkcionalne specifikacije sistema
    - črna škatla
  - in sliko o delovanju sistema
    - katere entitete nastopajo v sistemu, kateri procesi so prisotni v sistemu itd.
    - prozorna škatla
- Iz rezultatov ankete moramo narediti diagram toka fizičnih podatkov

Anketna vprašanja - entitete

- Spisek entitet in njihovi prilastki:
  - Koliko je zaposlenih?
  - Kakšni obrazci obstajajo?
  - Kakšni podatki so zapisani na posameznih obrazcih?
  - Ali imate kakšne listke, na katere pišete začasne zabeležke?
  - Obstajajo kakšni posebni dogovori ali šifre?
    - na primer: kratice imen; če vržemo obrazec v eno škatlo pomeni nekaj in če v drugo nekaj drugega itd.

Anketna vprašanja - procesi

- Spisek procesov, njihova vloga, vstopni in izstopni dokumenti
  - Kaj dela posamezen zaposleni?
  - Kdo izpolnjujejo posamezne obrazce, zabeležke itd.?
  - Komu in kako jih preda?
  - Kaj prejemnik podatkov z njimi naredi?

DTP - fizični

- Na podlagi analize ankete izdelamo diagram toka fizičnih podatkov
- Za osnovo vzamemo diagram nivoja 0 in vanj dodamo osnovne procese ter podatke, ki se med njimi izmenjujejo
- Najprej naredimo diagram nivoja 1 in potem, po potrebi vsak proces tega nivoja razčlenimo na naslednjem nivoju

Diagram nivoja 0

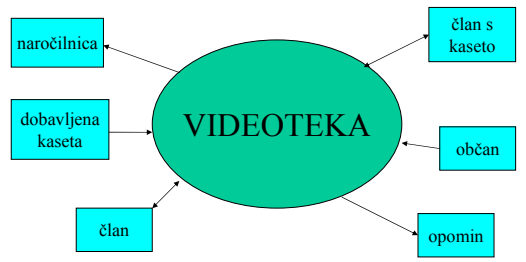


Diagram nivoja 1

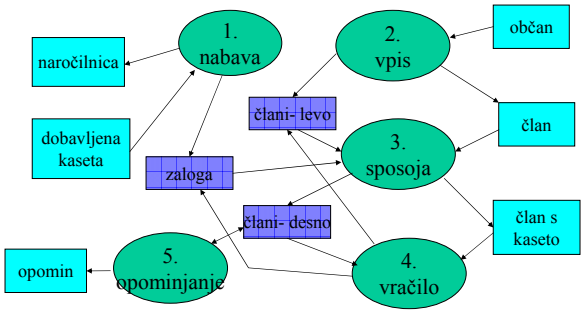


Diagram nivoja 1

- Na diagramu nivoja 1 običajno ni več kot 5 procesov, ker sicer postane vse skupaj nepregledno
- Na primer: recimo, da bi obstajal poseben proces za hkratno vračilo in izposojo kasete
  - združimo procese *izposoja*, *vračilo* in *vračilo s sposojo* v en proces *vračilo in sposoja*, ki ga razčlenimo na naslednjem nivoju
- Na diagramu so se pojavile ***notranje*** entitete *zaloga*, *člani levo* in *člani desno*

Diagram nivoja 1

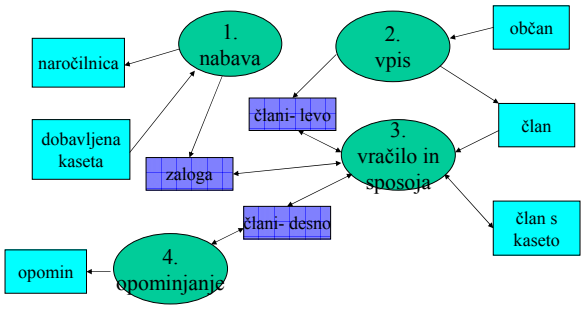
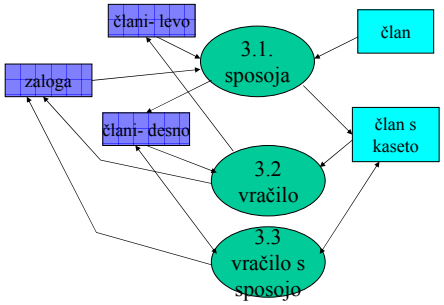
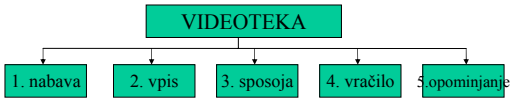


Diagram nivoja 2



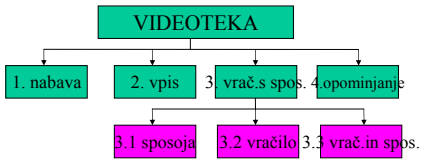
Strukturni diagram

- Strukturo procesov popišemo s ***strukturnim diagramom***
- Strukturni diagram nam poda:
  - sliko vstopnega menija v naš sistem
  - podrobnosti opisov procesov



Strukturni diagram

- druga inačica sistema ima drugačen strukturni diagram



Uporabniška programska oprema

2006/2007

Iztok Savnik

Informacijska analiza

- Sestoji iz naslednjih korakov:
  1. **zajem podatkov – anketa**
  2. **izdelavo toka fizičnih podatkov**
  3. izdelava toka logičnih podatkov
  4. ***izdelava podatkovnega modela***
  5. izvedba – definicija tabel in minispecifikacij

Slovar entitet

- slovar entitet vsebuje vse entitete, ki smo jih definirali v sistemu, njihove prilastke in tipe prilastkov
- v sistemu smo našli naslednje zunanje entitete:
  - naročilnica, dobavljena kaseta, član, opomin, občan in član s kaseto
- ter naslednje notranje entitete:
  - zaloga, listek, člani levo in člani desno
  - listek s kodo sposojene kasete, ki ga pripnemo na entiteto *člani levo* smo opisali kot ločeno entiteto

Slovar entitet

- Za vsako entiteto opišemo vse prilastke. Na primer:
  - zaloga:
    - naslov CHAR 120
    - režiser CHAR 80
    - trajanje INT
    - koda INT
    - lokacija INT
    - sposojeno BOOLEAN

Slovar entitet

  - člani desno:
    - ime CHAR 80
    - naslov CHAR 120
    - številka INT
  - člani levo:
    - ime CHAR 80
    - naslov CHAR 120
    - številka INT

Slovar entitet

  - listek:
    - koda filma INT
    - sposoja DATE
  - opomin:
    - ime CHAR 80
    - naslov CHAR 120
    - film1, film2, ... (naslov, koda, datum)

Odnosi med entitetami

- entitete so med seboj odvisne
  - člani desno ***ima pripet*** listek
  - listek ***opisuje*** zalogo
  - ...
- entitet lahko zaradi večje preglednosti združujemo v entitetne sklope
- odnosi ali relacije povezujejo entitete med seboj
- tudi relacije imajo lahko svoje prilastke

Množice

- entiteta je en element, predmet, ki jih združujemo v ***množice***
  - podobno kot združujemo predmete v razrede
- prav tako je relacija samo povezava med dvema entitetama in jih prav tako združujemo v množice

Množice entitet/relacij

- množica entitet (relacij) sestoji iz veliko entitet, ki jih moramo razlikovati
- dve entiteti bi lahko definirali kot različni
  - če bi se razlikovali v vrednosti vsaj enega prilastka
- ker je to prezahtevno, postavimo načelo ***osnovnega (primarnega) ključa*** (*primary key*) - ključna beseda

Osnovni / primarni ključ

- Nek prilastek entitete (relacije) je primarni ključ, če lahko po njem razlikujemo poljubni entiteti (relaciji) v množici entitet (relacij).
- Če več prilastkov skupaj predstavlja ključ, govorimo o ***sklopljenem ključu***.
- včasih se zgodi, da entiteta nima primarnega ključa in takšna entiteta je ***šibka entiteta***, medtem ko so druge entitete ***močne entitete***

Slovar entitet

- Za vsako entiteto opišemo vse prilastke. Na primer:
  - zaloga:
    - naslov CHAR 120
    - režiser CHAR 80
    - trajanje INT
    - **koda INT**
    - lokacija INT
    - sposojeno BOOLEAN

Slovar entitet

  - člani desno:
    - ime CHAR 80
    - naslov CHAR 120
    - **številka INT**
  - člani levo:
    - ime CHAR 80
    - naslov CHAR 120
    - **številka INT**

Slovar entitet

- listek:
  - koda filma INT
  - sposoja DATE
- opomin:
  - ime CHAR 80
  - naslov CHAR 120
  - film1, film2, ... (naslov, koda, datum)

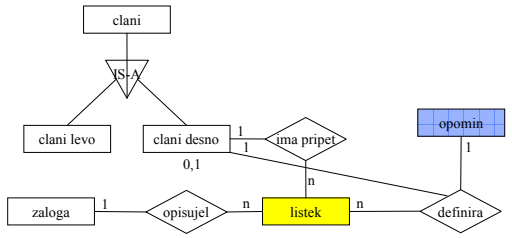
IS-A

- posebni obliki odnosa med entitetami sta ***generalizacija*** in ***specializacija***
- sta podobni kot dedovanje pri predmetno naravnanem programiranju
  - A ***IS-A*** B, C - pomeni, da je A posplošitev B in C-ja; ali da sta B in C posebna primera A-ja
  - v obeh primerih B in C vsebujeta vse prilastke A-ja in morda se kakšne svoje

E-R diagram

- z E-R diagramom popišemo odvisnosti med entitetami sistema na grafičen način
- posamezne entitete so označene s pravokotniki in relacije med njimi z romboidi
- na diagramu označimo večkratnost relacij
- relacije so lahko:
  - 1:1, kar pomeni da vsaki entiteti na eni strani relacije ustreza ena entiteta na drugi strani
  - 1: n ali n:1
  - n:n

Delni E-R diagram



Od entitet k tabelam

- vsaka množica je predstavljena kot tabela, kjer so stolpci prilastki in vrstice posamezne entitete
- vsaka tabela ***mora*** vsebovati ključ
  - z močnimi entitetami ni težav
  - pri šibkih entitetah pa definiramo kot ključ tiste prilastke, ki so ključ entitete, od katere je šibka entiteta odvisna
- zunanje entitete ni nujno, da predstavimo kot tabele - na primer poročila

Tabele

– zaloga:

koda	naslov	režiser	trajanje	lokacija	sposoj.
123	Bla	On	175	231	DA
454	Ble	Ona	143	244	NE

Slovar entitet

– listek – pozor, *številka* je primarni ključ člana:

številka	koda	sposoja
72940	123	10.3.2003

Uporabniška programska oprema

2006/2007

Iztok Savnik

Informacijska analiza

- Sestoji iz naslednjih korakov:
  1. **zajem podatkov – anketa**
  2. **izdelavo toka fizičnih podatkov**
  3. izdelava toka logičnih podatkov
  4. **izdelava podatkovnega modela**
  5. ***izvedba – definicija tabel in minispecifikacij***

Definicija tabel

- Tabele definirajo ***relacije*** med stolpci (prilastki)
- Vse tabele skupaj tvorijo ***podatkovno bazo***
- Nad tabelami so definirane operacije ***relacijske algebre***

Relacijska algebra

- Imamo naslednje tabele s prilastki ter vrednostmi
  - T_1 = {ab, cd} prilastka: p_1 p_2
  - T_2 = {XY, UV} prilastka: p_3 p_4
  - T_3 = {aα, bβ} prilastka: p_1 p_5

Relacijska algebra

T_1 = {ab, cd}
T_2 = {XY, UV}
T_3 = {aα, bβ}

- V relacijski algebri poznamo operacije:
  1. + (seštevanje) je ∪
  2. množenje je kartezični produkt

     $T_A = T_1 \times T_2$ = {abXY, abUV, cdXY, cdUV}
  1. selekcija / izbira: izberemo tiste elemente množice, ki ustrezajo pogoju

$$\sigma_{p1=a}(T_1) = \{ab\alpha\}$$

Relacijska algebra

T_1 = {ab, cd}
T_2 = {XY, UV}
T_3 = {aα, bβ}

1. projekcija: v manj razsežnostni prostor – pri vseh elementih izločimo en prilastek:

$$\pi_{p3}(T_2) = \{Y, V\}$$

– poznamo še pomembno izpeljano operacijo zlitje (*join*):

$$T_B = T_1 \infty_{T1.p1,\ T3.p1}\ T_3 = \pi_{T3.p1}(\sigma_{T1.p1=T3.p1}(T_1 \times T_3))$$

$$T_B = \{ab\alpha\}$$

Definicija tabel

- Tabele definirajo ***relacije*** med stolpci (prilastki)
- Vse tabele skupaj tvorijo ***podatkovno bazo***
- Nad tabelami so definirane operacije ***relacijske algebre***
- Spoštovati moramo nekatera pravila, da zagotovimo:
  - dobro definiranost operacij relacijske algebre in
  - usklajenost (konsistentnost) podatkovne baze

Pravila

- podatki v tabelah se ne smejo ponavljati
  - imejmo dve tabeli: {stevilka, ime, priimek, naslov} in {stevilka, ime, priimek, koda sposojenega filma}
  - ideja: če popravljamo podatke, jih popravljamo na enem samem mestu - v eni sami tabeli
- pravila, ki določajo obliko odnosa med tabelami (funkcionalne odvisnosti)
  - dve entiteti ne smeta biti v odnosu n:n - dodamo novo entiteto
  - dve entiteti ne smeta biti v odnosu 1:1 - entiteti združimo

Definicija procesov

- Ko smo končali z definicijo podatkov in smo definirali vse tabele se vrnemo k procesom - diagramu toka podatkov
- Pripravimo logične podatke (nasprotno od fizičnih), ki uporabljajo nov podatkovni model
- Nov model vsebuje sloni na nivoju 1 diagrama fizičnih podatkov

Definicija procesov

- Posamezne procese drobimo (na naslednji nivo) toliko časa, da jih lahko opišemo s preprostim (pod)programom
- Opisu procesa s podprogramom pravimo ***minispecifikacija***
  - naj bi dovoljeval neposredno kodiranje
  - vključuje že operacije nad podatkovnimi tabelami

Slovar entitet

- slovar entitet vsebuje vse entitete, ki smo jih definirali v sistemu, njihove prilastke in tipe prilastkov
- v sistemu smo našli naslednje zunanje entitete:
  - naročilnica, dobavljena kaseta, član, opomin, občan in član s kaseto
- ter naslednje notranje entitete:
  - zaloga, listek, člani levo in člani desno
  - listek s kodo sposojene kasete, ki ga pripnemo na entiteto *člani levo* smo opisali kot ločeno entiteto

Uporabniška programska oprema

2006/2007

Iztok Savnik

Splet in sodelovanje

- Splet predstavlja infrastrukturo, preko posamezni elementi sodelujejo
- Sodelujejo lahko:
  - na enakovreden način (*peer-to-peer*, P-2-P)
  - ali kot strežnik in odjemalec
- Slednji način je veliko pogostejši in preprostejši za izdelavo

Naslavljanje

- Elementi na Spletu, ki želijo sodelovati:
  - se morajo poznati
  - morajo se enolično razlikovati - imeti morajo enolično ime ali naslov
- Za naše potrebe bo takšen enoličen naslov ***URL*** - *universal resource locator* (RFC1808, glej www.ietf.org)

  <storitev>://<naslov vozla>/<naslov>
  - glej www.w3.org/Addressing

Naslavljanje

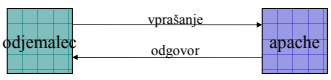
- naslov vozla enolično določa računalnik na Spletu (www.pef.upr.si)
- storitev določa storitev, ki jo ta računalnik nudi (http, https, mailto)
- storitev ima lahko za svoje potrebe dodaten naslov (MARA/2, ur)

Storitev http

- Za nas je zanimiva storitev ***http*** - *hyper text transport protocol*
  - RFC 1945 (1.0); 2058, 2616 (1.1)
- Protokol definira kako lahko elementi na spletu med seboj komunicirajo - jezik komunikacije

Odjemalec in strežnik

- odjemalec postavi vprašanje in strežnik odgovori
- med dvema vprašasnjema ni nobene povezave
- vprašanje ima obliko:
  <ukaz> <parametri>

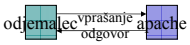


Primer komunikacije

```
Andy@Svarun:16[25]%> telnet www.pef.upr.si 80
Trying 193.2.97.193...
Connected to Perun.pef.upr.si.
Escape character is '^]'.
GET /MARA/
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
 <head>
  <title>Index of /MARA</title>
 </head>
 <body>
<h1>Index of /MARA</h1>
...
```


Osnove protokola

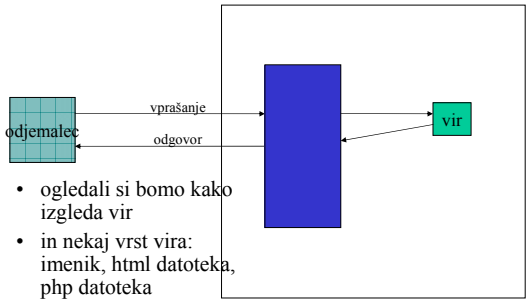
- Osnovni ukaz je **GET**, ki od strežnika zahteva vsebino nekega naslova
- Ta ukaz se sproži, ko izberemo povezavo v brskalniku
- Ukaz je pasiven – strežniku ne moremo ničesar sporočiti
- Pomagamo si lahko s primerno narejeno stranjo, ki v URL skrije dodatne parametre
  - takšne strani lahko tvorimo dinamično



Osnove protokola

- Razumevanje vprašanja je stvar strežnika (apache, httpd)
  - za dokumentacijo glej www.apache.org
  - */MARA/* mu pomeni, da mora poslati vsebino imenika /var/www/html/MARA
  - /usage/ mu pomeni, da mora poslati vsebino datoteke /var/www/html/usage/index.html
  - vse to je zabeleženo v konfiguracijski datoteki

Delovanje strežnika

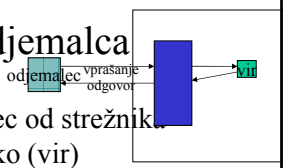


- ogledali si bomo kako izgleda vir
- in nekaj vrst vira: imenik, html datoteka, php datoteka

Kje se kaj dogaja in varnost

- Vse se dogaja na strežniški strani
  - to je naš sistem in naš program teče na njem
  - strežnik ni nujno, da je samo naš, ampak smo lahko samo eden od uporabnikov, zato previdnost
  - **varnost sistema**
- Če želimo, da se kaj dogaja na odjemlačevi strani, nam (našemu programu) mora to dovoliti odjemalec
  - varnost odjemalčevega sistema pred našim programom
  - poseben sistem za poganjanje našega programa - Java

Podatki od odjemalca



- Z ukazom GET odjemalec od strežnika zahteva določeno datoteko (vir)
- Vir je lahko program, ki se izvede na strežnikovi strani
- Doslej smo videli, da samo strežnik posreduje podatke odjemalcu

Osnove protokola

- Včasih želimo, da bi odjemalec posredoval podatke strežniku:
  - želimo, da nam uporabnik preko brskalnika vrne svoje lastne podatke (na primer ime in priimek, vtise, ...)
- V ta namen pozna protokol ukaz **POST**

Zbiranje podatkov

- Da lahko odjemalec posreduje podatke strežniku, jih mora najprej zbrati
- V ta namen obstaja v HTML jeziku posebna značka (in struktura) **FORM** - obrazec

Obrazec

- V obrazcu nastopajo lahko različna vnosna polja (glej www.w3.org/MarkUp/html-spec)
- Vsa vnosna polja so označena z značko **INPUT**
- Razlikujejo se po tipu **TYPE**
- glej še www.htmlhelp.com

Polja obrazca

- V obrazcu lahko nastopa poljubno število vnosnih polj
- Da bo strežnik razlikoval med vrednostmi vnešenimi v posamezna polja, morajo vsa polja imeti prilastek **NAME**
- Prilastek *NAME* predstavlja **ime polja** (prim. ime spremenljivke)

Polja obrazca

- Poleg imena imajo polja lahko tudi (začetno) vrednost - **VALUE**
- To vrednost uporabnik popravlja
- Splošna oblika:

```
<INPUT TYPE="tip"
       NAME= "ime"
       VALUE="vrednost" ...>
```


Tipi vnosnih polj

- TYPE=text: vnos poljubnega besedila
- TYPE=password: enako kot *text*, le da se tipkano besedilo ne vidi

Tipi vnosnih polj

- TYPE=checkbox: stikalo, ki ga lahko vključimo ali izključimo
- TYPE=radio: podobno kot *checkbox*, le da vključimo lahko samo eno od večih stikal
  - stikala, ki so povezana med seboj, imajo enako ime a različne vrednosti
  - eno od polj mora biti na začetku izbrano (začetna vrednost) in ima zato dodatni prilastek **CHECKED**

Tipi vnosnih polj

- TYPE=submit: je samo gumb, ki strežniku pošlje izpolnjen obrazec
  - pošlje mu imena polj in vrednosti
  - prilastek *NAME* tukaj pomeni, kaj se bo izpisalo na gumbu (*prekliči*, ipd.)
  - obstaja lahko več takšnih polj z različnimi imeni in strežnik bo obveščen katerega je uporabnik pritisnil
- TYPE=image: podobno, le da se na gumbu nariše slikica
- TYPE=reset: podobno kot prejšnji gumb, le da se vrednosti obrazca postavijo na začetno vrednost (*VALUE* prilastki)

Tipi vnosnih polj

- TYPE=file: dovoljuje uporabniku, da pošlje (vstavi) celotno datoteko
- TYPE=hidden: v bistvu se nič ne izpiše

Pošiljanje obrazca

- Določeno s prilastkom **METHOD** pri znački *FORM*
- Obstajata dve metodi:
  - **GET**: pošlje vrednosti v obrazcu preko URL-ja
  - **POST**: zahteva dodatni pogovor s strežnikove strani

Pošiljanje obrazca -primer

- Primer na datoteki form-get.gif pošlje naslednjo zahtevo

```
GET
    /nekaj/nekje/nekako?
    name=Ime+in+priimek&
    gender=male&
    family=999&
    city=koper&
    other=Besedilo+je+tukaj%0D%0Ain+se+malo+besedila
      &
    nickname=Ta+pa+je.
HTTP/1.0
```

Uporabniška programska oprema

2006/2007

Iztok Savnik

PHP

- PHP je splošen programski jezik
  - ima podobne konstrukte kot Java ali C
  - je tudi delno predmetno naravnan
- Razvit je bil z namenom podpore razvoja strežniških aplikacij za http strežnike
- zelo pregleden priročnik je na voljo na

  www.php.net
- Za navodila in literaturo glej domačo spletno stran predmeta

Risanje črte

- Narisali bomo črto iz pikic (.)
- Program:

```
crta(dolžina):
  if (dolžina = 0) ne naredi nič
  nariši pikico
  črta(dolžina-1)
```

- Primer programa je v *primer1.php*

Primer 1

- PHP program vstavimo v HTML dokument kot *script*:

```
<SCRIPT language="php">
...
</SCRIPT>
```

- Lahko tudi kot posebno značko:

```
<?php ... ?>
<? ... ?>
```


Modularizacija

- PHP programe lahko modulariziramo
- naredimo lahko knjižnice
- lahko tudi neposredno vključimo kose programa v druge programe:
  - include
  - require
  - *poščite v dokumentaciji razliko!*
- glej *primer2.php* in *crta.php*

Risanje trikotnika

- Radi bi narisali trikotnik:

```
. . . . /
. . . /
. . /
. /
/
```

- Uporabili bomo podprogram za risanje črte

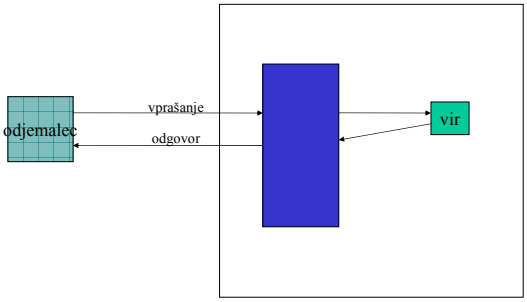
Risanje trikotnika

- Program:

```
trikotnik(višina):
  if (višina < 0) smo zaključili
  črta(višina)
  print /, \n
  trikotnik(višina-1)
```

- Glej *primer3.php* in *trikotnik.php*

Delovanje strežnika



Podatki od brskalnika

- Podatke z brskalnika (od odjemlaca) do strežnika dobimo z uporabo obrazca (*form*)
- V obrazcu uporabnik vnese svoje podatke v polja, ki imajo svoja imena
- Ko pritisne na tipko *submit*, se vrednosti polj prenesejo do strežnika
- Na strežniku imamo naš PHP program, ki lahko ustrezno ukrepa glede na prenešene podatke

Podatki od brskalnika

- Brskalnik lahko pošlje tri vrste podatkov (podatke lahko pošlje na tri načine):
  - s pomočjo POST ukaza
    - <FORM METHOD="POST" ...>
  - s pomočjo GET ukaza
    - <FORM METHOD="GET" ...>
    - izbira <A HREF="...">
  - kot piškotek - *cookie*

Podatki z ukazom POST

- Glej *primer4.html*:
  - obrazec za vnos višine trikotnika, ki pošlje podatke na URL *primer4.php*
  - ime polja je ***visina***
- Program za izris trikotnika *primer4.php*:
  - do vrednosti poslanega polja pridemo na tri načine (*glej primer4.php*):
    - $_POST['visina']
    - $_REQUEST['visina']
    - import_request_variables('p', 'posted_'); $posted_visina;

Podatki z ukazom GET

- Glej *primer5.html*, ki uporabi program *primer5.php*
  - ime polja je ponovno *visina*
- Po pritisku na tipko *submit* se pošlje zahteva strežniku:

```
GET primer5.php?visina=6
```

- Razlika med *GET* in *POST* načinom pošiljanja obrazca, ki jo opazi uporabnik je:
  - da brskalnik pri *GET* načinu ne vpraša ali naj v resnici pošlje izpolnjen obrazec strežniku

Podatki z ukazom GET

- Do vrednosti poslanega polja pridemo na tri načine (*glej primer5.php*):
  - $_GET['visina']
  - $_REQUEST['visina']
  - import_request_variables('g', 'get_'); $get_visina;
- Na podoben način lahko pridemo do podatkov, če so ti poslani kot del URL naslova pri <A ...> znački:

  <A HREF="naslov.php?parameter=vrednost>izbira</A>

Kako do podatkov

- Najsplošnejši način dostopa do podatkov je z uporabo funkcije

  import_request_variables("<tip>", "<predpona>")
- *tip* je lahko:
  - g: GET
  - p: POST
  - c: cookie
- *predpona* je dodana vsem spremenljivkam danih tipov, ki so odslej na voljo modulu

Uporabniška programska oprema

2006/2007

Iztok Savnik

Zaščita dostopa

- Včasih želimo zaščititi vsebin spletne strani

Zaščita dostopa

- Ščitimo lahko imenik in vse njegove podimenike
- Okolje za zaščito nam nudi http strežnik
- V imeniku imamo datoteko *.htaccess*, ki mora biti zaščitena tako, da jo vsi lahko berejo
- Vsebina datoteke je približno takšna (glej httpd.apache.org/docs/howto/htaccess.html):

htaccess

```
AuthType Basic
AuthName "Naslov okna"
AuthUserFile  <datoteka z gesli>
Require user <uporabnik, ki lahko dostopa>
```

- v konfiguracijski datoteki moramo dovoliti omejevanje dostopa

  <Directory /pot/do/imenika/>
  Options Indexes FollowSymLinks MultiViews
  AllowOverride AuthConfig
  </Directory>

Okna

- Okno je del fizičnega zaslona, ki je prirejen skupini procesov
- Ti procesi uporabljajo okno za izpisovanje, branje in ga sploh nadzorujejo
- Eno okno lahko pripada večim procesom
- Enemu procesu lahko pripada več oken
- V enem oknu imamo lahko več oken
- Glej *man window* ali *man screen*

Nadzor nad okni

- Okna lahko povečujemo ali pomanjšujemo
- Lahko se sprehajamo od enega okna k drugemu
- Posebna oblika oken so *grafična okna*
  - okna v X okolju
  - okna v MS okolju
    - vsako okno je ločen proces
    - okno osnovni gradnik ⇒ *windows*

Grafična in navadna okna

- Do razlikovanja pride samo v Unix svetu
- Tam se grafična okna imenujejo tudi okvirji (*frames*)
  - primerjaj emacs in v njem *open frame*
- Okno je vir dogodkov
  - pritisk na tipko / tipkanje
  - v grafičnih okoljih dogodki povezani s kazalčnikom (*miška*)

Okna v oknih

- V grafičnih oknih se lahko pojavljajo druga okna
  - podobno kot pri ukazu *window*
  - nekatera pod-okna so postala precej standardna
    - orodna vrstica
    - menijska vrstica

Okna in njihove lastnosti

- Okno različne lastnosti
  - ime
  - naslov
  - velikost
  - položaj na zaslonu
  - ...
  - glej *man X*, *man xterm*, ...
  - oknu prirejen proces in okno sta običajno nerazdružljiva - primerjaj *xterm*, *xfig* itd.

Okna in HTML

- Brskalnikovo okno predstavlja osnovo okno, v katerem se prikazuje naša vsebina
- Po potrebi lahko odpremo novo okno
- Pokažemo novo vsebino v (drugem) oknu
- Zapremo okno itd.
- Glej *primer.html*
  - za razhroščevanje uporabi *javascript:* kot URL

Okna v oknih in HTML

- Takšnim oknom običajno rečemo ***okvirji*** (***frames***)
- Podobno kot pri tabelah celice, lahko pri okvirjih slednje zlagamo skupaj
- Glej primer na
  wp.netscape.com/assist/net_sites/frame_syntax.html

Okvirji v brskalniku

- Razlike med celicami tabele in okvirji v množici okvirjev:
  - vsebine vseh celic tabele so v isti datoteki
  - zato lahko sodelujejo med seboj - jih lahko uporabimo v istem obrazcu (na primer)
  - pri okvirjih so slednji povsem ločeni - vsak je v svoji datoteki in zato ne morejo sodelovati med seboj

Uporabniška programska oprema

Zakaj bi pisali proste programe?

2006/2007

Aleš Košir

Vsebina

- **Povzetek predstavitve**
- **Komu je predstavitev namenjena?**
- **Razlika med avtomobilom in programom**
- **Kaj je licenca?**
- **Tehnološka matrika Craiga Burtona**
- **Pravica do izbire**
- **Značilne družine licenc, odprta koda in GPL**
- **Lastni razvoj proste programske opreme**
- **Razvoj proste programske opreme v podjetju HERMES SoftLab**
- **'Slovenska scena'**
- **Sklep**

Povzetek

- **Prosta programska oprema je zanimiv tehnološki, pravni in sociološki fenomen, ki je postal posebej znamenit v podobi operacijskega sistema Linux. Morda ni povsem očitno, a razširjanje programske opreme se razlikuje od prodaje avtomobilov.**
- **Ogledali si bomo te razlike, pokazali, zakaj bi kdo pisal prosto programsko opremo, se pogovorili o tem, kako to opremo ustvarjati in pojasnili z njo povezane poslovne modele.**

Komu je predstavitev namenjena?

- **Študentom drugega letnika Pedagoške fakultete,**
- **ki iz rabe poznajo prosto programsko opremo in njene osnovne koncepte**
- **ter znajo uporabljati prosta razvojna orodja.**

Program proti avtu

- **Vrednost programa = uporabna + pričakovana vrednost**
- **Vrednost avtomobila = uporabna + tržna vrednost**

- **Program lahko izgubi vso svojo vrednost, avto ohrani vsaj materialno vrednost**
- **Program: imamo pravico do uporabe**
- **Avto: smo njegov lastnik**
- **Program: ne moremo ga popravljati**
- **Avto: napake reklamiramo**

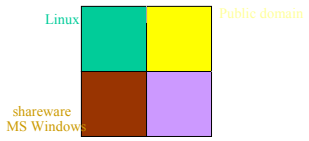
Licenca

- **Pravica (dovoljenje) do uporabe programa**
- **Lastnina programa**
- **Lastnik programa z licenco uresničuje svoj poslovni model**
- **Imetnik pravice do uporabe programa sme program uporabljati**

Nadomestilo za dovoljenje uporabe

- **Brezplačno nadomestilo**
- **Enkratno ob nakupu dovoljenja za uporabo**
- **Letno nadomestilo (15-20% vrednosti) za dovoljenje uporabe**

Tehnološka matrika Craiga Burtona



Pravica do izbire!

- **Za uporabnike**
- **Za izdelovalce (programerje ali podjetja)**

Faze zavedanja svobode in omejitev

- **1950 – 1975: samoumevnost (TMRC)**
- **1975 – 1984: zapiranje (Unix, MS DOS)**
- **1984 – 1997: političnost in filozofskost (GNU, OS)**
- **2001 – ...: pragmatičnost in izbira**

Terminologija

- **Prosta programska oprema (Free Software)**
- **Odprta koda (Open Source)**

Značilne družine licenc

- **Licence zaprte kode**
  - MS Operating System
  - Delphi
  - MS Internet Explorer
- **Pogrami na pokušino (Shareware)**
- **Licence odprte kode**
  - GNU General Public Licence (GPL)
  - GNU Lesser General Public Licence (LGPL)
  - BSD, X Consortium
  - Artistic (perl)

Poslovni model zaprte programske opreme

- **Izkoriščanje pričakovane vrednosti za uporabnika**
  - nekončane začetne različice
  - financiranje prihodnjega razvoja
  - razprodaja

Odprta koda (OpenSource)

- **Ideja:**
  - Programi se razvijajo, če jih programerji lahko berejo, razširjajo in spreminjajo. (Bruce Perens, 1997)

Prosti programi

- **Podeljene pravice:**
  - Uporaba programa
  - Dostopnost izvorne kode
  - Razmnoževanje
  - Razširjanje brez omejitev

Licenca GPL

- Richard Stallman: General Public License

1. **Vsi uporabniki imajo možnost** sodelovati **pri razvoju in** izmenjevati **programsko opremo**
2. **"Kar prejmemo, smo dolžni** pod enakimi pogoji **nuditi naprej"**
3. **Če javno izdamo program, smo dolžni priskrbeti tudi njegovo** izvorno kodo
4. **Za program lahko kot avtor zahtevamo** plačilo**, njegova koda pa mora biti** brezplačna **(lahko zahtevamo le nadomestilo minimalnih stroškov)**
5. **Ne moremo preprečiti, da se program ne razširja tudi** brezplačno
6. **Kot** izključni avtor **pa lahko program in kodo hkrati razširjamo tudi pod drugimi licencami in na druge načine (proti plačilu, pod NDA...)**

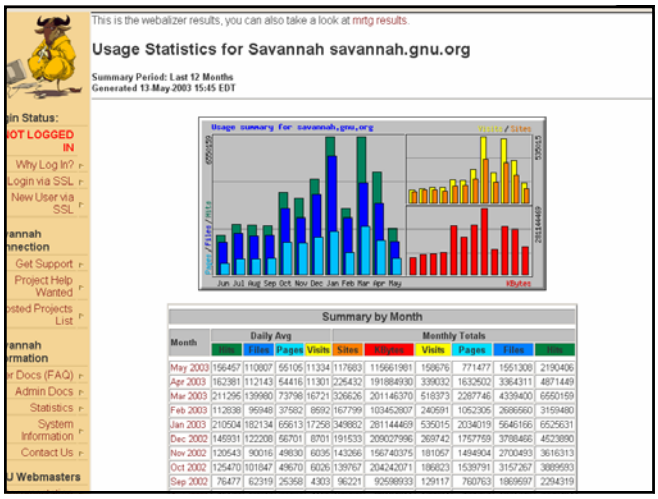
Poslovni model odprte programske opreme

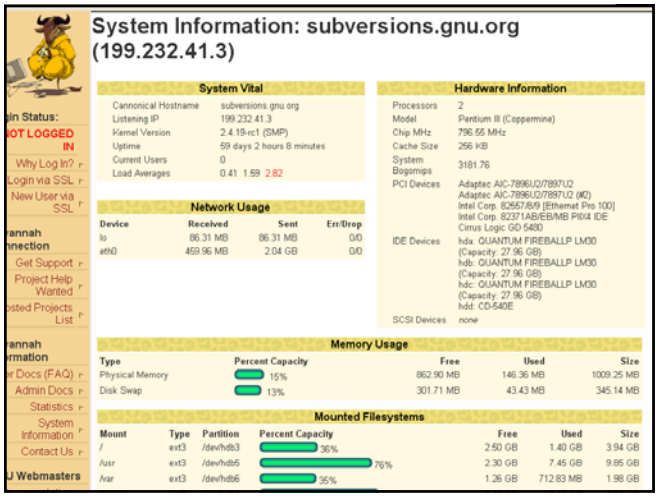
- **Prenos vrednosti**
  - strežniki, odjemniki (Netscape)
  - strojna oprema in gonilniki
  - storitve in programi
  - relikvije in literatura (O'Reilly)
  - nova in starejša različica (Ghostscript)
  - nemotena in motena različica (Opera)

Prosta orodja

- **Razvojno okolje**
  - Prevajalnik gcc: C, C++, java, pascal, fortran...
  - Okolje za prevajanje make
  - Sistem za nadzor različic RCS, CVS...
  - Orodja za lokalizacijo: gettext...
  - Orodja za razhroščevanje: gdb, nadzor puščanja pomnilnika
  - Sistemi za skupinski razvoj in farme za prevajanje http://sourceforge.net, http://savannah.gnu.org
    - sledenje napak
    - forumi za diskusijo
    - domače strani projektov
    - varnostno shranjevanje
    - varne prijave in jasne vloge na projektu
    - projektne metrike (število sprememb, število sodelavcev, popularnost projekta...)


NEWSFORGE
All Technology News, All the time.
my sf.net
software map
foundries
about sf.net
Project: JBoss.org: Summary
Summary | Admin | Home Page | Tracker | Bugs | Patches | RFE | Lists | Tasks | Docs | News | CVS | Files |
The JBoss/Server is the leading Open Source, standards-compliant, J2EE based application server implemented in 100% Pure Java
Developer Info
Project Admins:
juhalindfors
mnf999
patriot1burke
starksm
Developers:
89 [View Members]
Project UNIX name: jboss
Registered: 2001-03-15 09:43
Activity Percentile (last week): 99.9451%
Latest File Releases


This is the webalizer results, you can also take a look at mrtg results.
Usage Statistics for Savannah savannah.gnu.org
Summary Period: Last 12 Months
Generated 13-May-2003 15:45 EDT
Summary by Month
Month
Daily Avg
Monthly Totals
May 2003 156457 110807 55105 11334 117683 115661981 158676 771477 1551308 2190406
Apr 2003 162381 112143 54416 11301 225432 191884930 339032 1632502 3364311 4871449
Mar 2003 211295 139980 73798 16721 326626 201146370 518373 2287746 4339400 6550159
Feb 2003 112838 95948 37582 8692 167799 103452807 240591 1052305 2686560 3159480
Jan 2003 210504 182134 65613 17258 349882 281144469 535015 2034019 5646166 6525631
Dec 2002 145931 122208 56701 8701 191533 209027996 269742 1757759 3788466 4523890
Nov 2002 120543 90016 49830 6035 143266 156740375 181057 1494904 2700493 3616313
Oct 2002 125470 101847 49670 6026 139767 204242071 186823 1539791 3157267 3899593
Sep 2002 76477 62319 25368 4303 96221 92598933 129117 760763 1869597 2294319


System Information: subversions.gnu.org (199.232.41.3)
System Vital
Cannonical Hostname subversions.gnu.org
Listening IP 199.232.41.3
Kernel Version 2.4.19-rc1 (SMP)
Uptime 59 days 2 hours 8 minutes
Current Users 0
Load Averages 0.41 1.59 2.82
Hardware Information
Processors 2
Model Pentium III (Coppermine)
Chip MHz 796.55 MHz
Cache Size 256 KB
System Bogomips 3181.76
PCI Devices
IDE Devices
SCSI Devices none
Network Usage
Device Received Sent Err/Drop
lo 86.31 MB 86.31 MB 0/0
eth0 459.96 MB 2.04 GB 0/0
Memory Usage
Type Percent Capacity Free Used Size
Physical Memory 15% 862.90 MB 146.36 MB 1009.25 MB
Disk Swap 13% 301.71 MB 43.43 MB 345.14 MB
Mounted Filesystems
Mount Type Partition Percent Capacity Free Used Size
/ ext3 /dev/hdb3 36% 2.50 GB 1.40 GB 3.94 GB
/usr ext3 /dev/hdb5 76% 2.30 GB 7.45 GB 9.85 GB
/var ext3 /dev/hdb6 35% 1.26 GB 712.83 MB 1.98 GB

Dodatek

- **Raba Linuxa v podjetju Hermes SoftLab**
- **'Slovenska scena'**

Raba Linuxa v podjetju HERMES SoftLab

- **Osebna delovna postaja**
- **Razvojno okolje**
- **Infrastrukturni strežnik**

Osebna delovna postaja

- **Grafično namizje (KDE, GNOME, IceWM, fwwm2, ...)**
- **Poštni odjemalec**
- **Brskalnik**
- **Razvojno okolje (IDE)**

Razvojno okolje

- **Rabljen pri 11 večjih razvojnih projektih:**
  - varnostno shranjevanje - Omniback,
  - autocad - ME10,
  - strežnik za elektronsko izdajanje kart - E-ticket,
  - vodenje hierarhičnega shranjevanja podatkov - GRAU,
  - vodenje virov za shranjevanje podatkov – StoRM
  - merjenje virov in upravljanje z njimi – RPM

Strežniki

- **Spletni strežnik (Apache)**
- **Internetne storitve (DNS, router, news, ftp...)**
- **Poštni strežnik (sendmail...)**
- **Strežnik za nadzor različic (CVS, RCS, CC)**
- **Aplikacijski strežnik**
- **Strežnik za sledenje napakam (Bugzilla)**
- **Strežnik za avtomatsko testiranje**
- **Grafični strežnik (X server)**
- **Tiskalniški strežnik**
- **Datotečni strežnik (samba)**
- **Strežnik s podatkovno bazo (SQL, Informix...)**
- **Strežnik za varnostno shranjevanje**

Korist za družbo

- **Zaposleni so aktivni člani Lugosa,**
- **Promocija odprtih standardov in proste programske opreme**
- **Sodelovanje pri slovenjenju proste programske opreme**

Povzetek

- **Najobsežnejši razvoj in najbolj vsestranska raba sistema Linux v Sloveniji**
- **Rabljen kot**
  - osebna delovna postaja
  - razvojno okolje z orodji
  - infrastrukturni sistem (strežnik, usmerjevalnik)
- **Uporabljene programe izboljšujemo in jih vračamo v skupno rabo (samba, IceWM…)**

'Slovenska scena'

- **Društva, organizacije:**
  - LUGOS, http://www.lugos.si
  - Slo-tech, http://www.slo-tech.com
  - uporabniki FreeBSD, http://www.si.freebsd.org/
- **Lokalizacija:**
  - GNU,
  - KDE,
  - GNOME,
  - Mozilla
- **Iniciative institucij:**
  - MID, MŠZŠ, CVI

Sklep

- **Pravica do izbire programske opreme**
  - izbiramo funkcionalnost,
  - a licenca je pomembna
- **Boj za odprte standarde**
  - Sobivanje in konkurenca rešitev
  - Pravica in možnost izbire med njimi

Programska oprema – zakonski vidiki

Prosojnice s predavanj za študente Pedagoške fakultete v Kopru
April, 2005

Izr. prof. dr. Cene Bavec
cene.bavec@guest.arnes.si
www2.arnes.si/~bavec

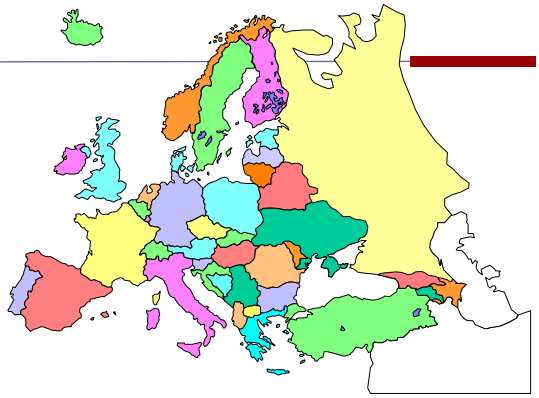
Vsebina

Precej zakonov ureja posamezna področja uporabe informacijskih tehnologij, vendar bomo omenili samo tiste, ki so povezani z razvojem in uporabo programske opreme:

1. zaščita intelektualne lastnine in avtorske pravice
2. elektronsko poslovanje in elektronski podpisi
3. varovanje podatkov in zasebnosti

2

Prof. dr. Cene Bavec, 2005



3

Prof. dr. Cene Bavec, 2005

Slovenija in EU

- EU sprejema direktive (evropske zakone), ki jih posamezne članice uvedejo v svoj pravni sistem.
- Slovenija je članica EU, zato imamo enako zakonodajo, kot jo imajo vse ostale članice EU!

4

Prof. dr. Cene Bavec, 2005

Minimalno zakonsko urejanje

Načelo čim bolj omejenega zakonskega predpisovanja je izjemno pomembno za nastajajočo informacijsko družbo, saj se drugače ne bo mogla dovolj hitro razvijati.

Najboljši primer je internet, ki je brez zakonskega urejanja postal svetovna infratruktura.

5

Prof. dr. Cene Bavec, 2005

Zakonodaja s področja varovanja intelektualne lastnine

6

Prof. dr. Cene Bavec, 2005

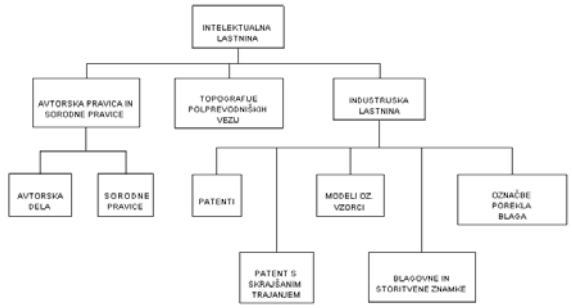
Intelektualna lastnina

Koncept intelektualne lastnine obsega pravice, ki izhajajo iz intelektualne aktivnosti na industrijskem, znanstvenem, literarnem in umetniškem področju.

7

Prof. dr. Cene Bavec, 2005

Koncept intelektualne lastnine

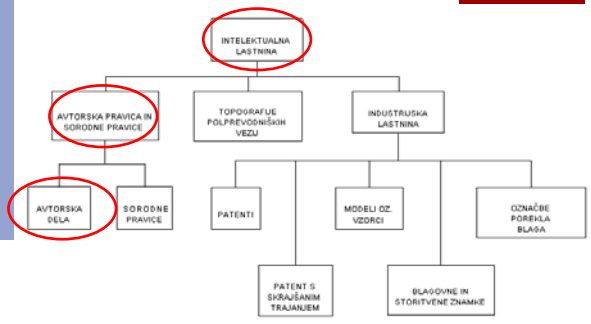


8

Prof. dr. Cene Bavec, 2005

Vir: Urad RS za intelektualno lastnino

Koncept intelektualne lastnine

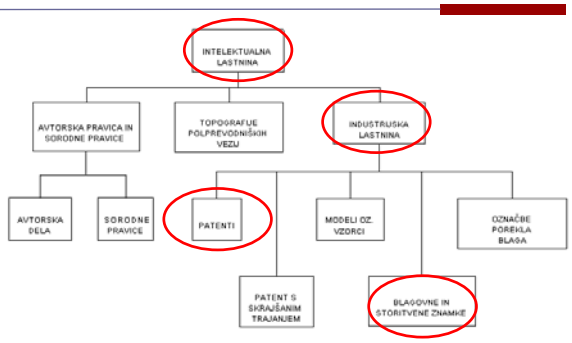


9

Prof. dr. Cene Bavec, 2005

Vir: Urad RS za intelektualno lastnino

Koncept intelektualne lastnine



10

Vir: Urad RS za intelektualno lastnino

Prof. dr. Cene Bavec, 2005

Avtorsko delo

Avtorska dela so individualne intelektualne stvaritve s področja književnosti, znanosti in umetnosti, ki so na kakršenkoli način izražene.

Avtorska pravica pripada avtorju na podlagi same stvaritve dela. To pomeni, da ni potrebna nobena formalnost, kot npr. registracija, da bi bilo delo avtorskopravno varovano.

Računalniški programi so v večini primerov avtorsko delo!

11

Prof. dr. Cene Bavec, 2005

Patent

Patent je izključna pravica fizične ali pravne osebe za izum, ki je nov, dosežen z ustvarjalnim delom na ravni izumiteljstva in je industrijsko uporabljiv.

Evropska unija dovoljuje patentiranje programske opreme (direktiva je še v postopku sprejemanja v Evropskem parlamentu).

12

Prof. dr. Cene Bavec, 2005

Zakon o avtorski in sorodnih pravicah

Vsebina zakona:

- moralne avtorske pravice
- materialne avtorske pravice
- prosta uporaba
- časovne omejitve avtorske pravice (70 let)
- dedovanje
- računalniški programi

13

Prof. dr. Cene Bavec, 2005

Računalniški programi v zakonu

- celota: programi in dokumentacija
- ideje niso predmet varstva
- delojemalec (programer) neomejeno prenese vse avtorske in materialne pravice na delodajalca
- kupec lahko popravi pogram brez avtorjevega dovoljenja
- največ dve varnostni kopiji
- uporabnik lahko testira in proučuje programe in ugotavlja logiko njihovega delovanja
- prepovedana je distribucija programa, za katerega se lahko domneva, da je nedovoljeni primerek
- posest takega primerka za gospodarske namene je kazniva

14

Prof. dr. Cene Bavec, 2005

Avtorske pravice in programska oprema

Kadar računalniški program ustvari delojemalec (programer) pri izpolnjevanju svojih obveznosti ali po navodilih delodajalca, ali ga ustvari avtor po avtorski pogodbi o naročilu, se šteje, da so materialne avtorske pravice in druge pravice avtorja na tem programu izključno in neomejeno prenesene na delodajalca ali naročnika, če ni s pogodbo drugače določeno.

15

Prof. dr. Cene Bavec, 2005

Ideje niso zaščitene

Ideje in načela, ki so osnova nekemu elementu računalniškega programa, vključno s tistimi, ki so osnova njegovim vmesnikom, ne uživajo varstva.

16

Prof. dr. Cene Bavec, 2005

Zakonodaja s področja elektronskega poslovanja in elektronskih podpisov

17

Prof. dr. Cene Bavec, 2005

Zakonska ureditev elektronskih podpisov

- Slovenija ima zakon o elektronskem poslovanju in elektronskem podpisu (junij 2000)
- Temeljni cilj: zagotoviti pravno veljavnosti digitalnih podpisov

18

Prof. dr. Cene Bavec, 2005

Zakon o el. poslovanju in el. podpisu

- Nediskriminacija elektronske oblike dokumentov v primerjavi s klasičnimi (papirnimi) oblikami
- Nekateri dokumenti so zakonsko izvzeti (oporoke itd.)

19

Prof. dr. Cene Bavec, 2005

Elektronski podpisi

Zagotavljajo:

- Identiteto imetnika elektronskega podpisa
- Integriteto (nespremenljivost) podpisanih podatkov

20

Prof. dr. Cene Bavec, 2005

Uporaba elektronskih podpisov

- uradne komunikacije (npr. javni razpisi)
- pogodbene komunikacije (npr. elektronsko kupovanje)
- samo identifikacija (npr. spletne strani)
- uporaba v osebne namene
- identifikacije v intranetu

21

Prof. dr. Cene Bavec, 2005

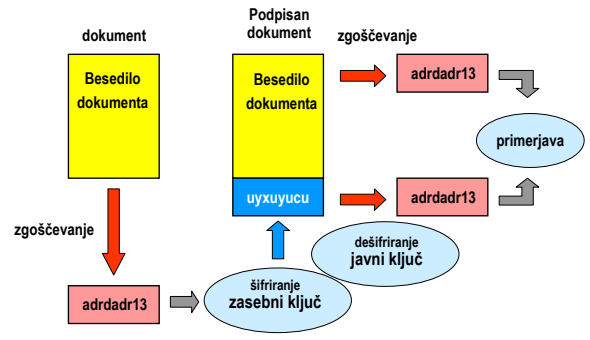
Elektronski podpisi - asimetrična kriptografija

- Dokumente podpisujemo z zasebnim (privatnim) ključem, ki ga poznamo samo mi
- Podpise preverjamo z javnim ključem, ki je na razpolago vsem uporabnikom
- Zgoščevanje je enosmerno (zaradi izgube informacije ne moremo regenerirati originalnega dokumenta)

22

Prof. dr. Cene Bavec, 2005

Elektronsko podpisovanje - primer



23

Prof. dr. Cene Bavec, 2005

Vir: Monitor, 2000

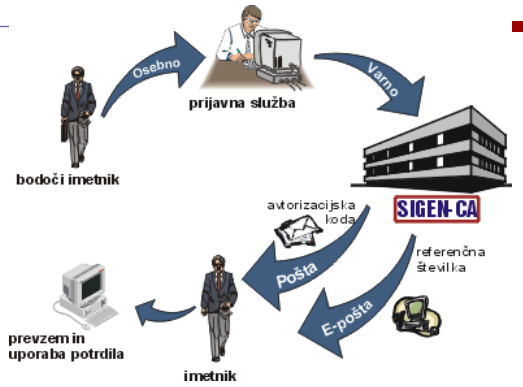
Overjanje javnih ključev

- Zagotoviti je potrebno, da ključ res pripada podpisniku dokumenta!
- To omogočajo posebne ustanove - agencije za overjanje javnih ključev, ki izdajajo lastnikom javnih ključev digitalne certifikate.
- Z digitalnim certifikatom lastnik dokazuje lastništvo javnega ključa.

24

Prof. dr. Cene Bavec, 2005

Pridobivane digitalnih potrdil



25

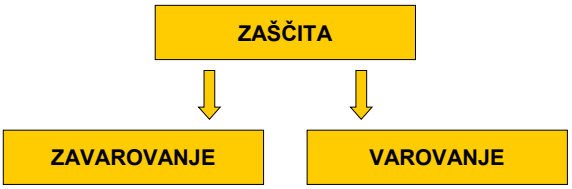
Prof. dr. Cene Bavec, 2005

Zakonodaja s področja varovanja osebnih podatkov in zasebnosti

26

Prof. dr. Cene Bavec, 2005

Zaščita - zavarovanje - varovanje



27

Prof. dr. Cene Bavec, 2005

Zavarovanje podatkov

Cilja zavarovanja je zaščititi podatke pred uničenjem, nepooblaščeno spremembo ali krajo!

V primeru uničenja, nepooblaščene spremembe ali kraje podatkov, ima največjo škodo ravno organizacija, ki ima te podatke (npr. banke). Zato v bistvu z zavarovanjem podatkov organizacije ščitijo same sebe.

28

Prof. dr. Cene Bavec, 2005

Varovanje podatkov

Cilj varovanja podatkov je preprečevanja razkritja podatkov nepooblaščeni osebi.

V primeru nepooblaščenega dostopa do podatkov praviloma nima škode organizacija, ki te podatke zbira, ampak oseba ali organizacija na katere se ti podatki nanašajo. Zato pogosto organizacije niso posebej motivirane za varovanje podatkov in potrebujemo zunanjo prisilo (praviloma zakon).

29

Prof. dr. Cene Bavec, 2005

Cilj varstva osebnih podatkov

Z varstvom osebnih podatkov se preprečujejo nezakoniti in neupravičeni posegi v zasebnost posameznika pri obdelavi osebnih podatkov, varovanju zbirk osebnih podatkov in njihovi uporabi.

30

Prof. dr. Cene Bavec, 2005

Načelo zakonitosti

Osebni podatki se lahko obdelujejo le, če je obdelava osebnih podatkov določena z zakonom ali, če ima upravljavec zbirke osebnih podatkov pisno privolitev posameznika.

Državni organi, organi lokalnih skupnosti in nosilci javnih pooblastil lahko obdelujejo samo tiste osebne podatke, za katere je tako določeno z zakonom.

31

Prof. dr. Cene Bavec, 2005

Katalog zbirk osebnih podatkov

Ministrstvo za pravosodje vodi in objavlja katalog zbirk osebnih podatkov, v katerem so vse zbirke, ki jih določajio področni zakoni.

32

Prof. dr. Cene Bavec, 2005

Pravice posameznika

Upravljavec zbirke osebnih podatkov mora na zahtevo posameznika:

- posamezniku posredovati izpis osebnih podatkov, ki so vsebovani v zbirki osebnih podatkov in se nanašajo nanj;
- posamezniku posredovati seznam tistih, katerim so bili v določenem obdobju posredovani osebni podatki, ki se nanašajo nanj;
- posamezniku omogočiti vpogled v vire, na katerih temeljijo zapisi, ki jih o posamezniku vsebuje zbirka podatkov in metodo obdelave

33

Prof. dr. Cene Bavec, 2005

Varstvo pravic posameznika

- posameznik, ki ugotovi, da so kršene njegove pravice, lahko s tožbo zahteva sodno varstvo
- v postopku, v katerem se odloča o tožbi, je javnost izključena

Pravice posameznika v zvezi z varstvom osebnih podatkov je mogoče omejiti samo izjemoma v primerih, ki jih določa zakon za potrebe nacionalne varnosti, obrambe, javne varnosti, preprečevanja, razkrivanja, odkrivanja in preganjanja kaznivih dejanj ali kršitve etičnih norm.

34

Prof. dr. Cene Bavec, 2005

Inšpekcijsko varstvo

Inšpekcijsko nadzorstvo nad izvajanjem določb tega zakona opravlja Ministrstvo za pravosodje.

Posameznik lahko poda prijavo tudi Republiškemu inšpektorju za varstvo osebnih podatkov.

35

Prof. dr. Cene Bavec, 2005

VAJA 1: Sistemska in Informacijska analiza

Skozi celotne vaje bomo naredili spletni Blog, ki ga bodo lahko uporabljali in vanj vnašali vsi študenti. Šli bomo skozi celoten proces izdelave in na koncu prišli do končnega uporabnega izdelka.

Sistemska analiza:

- določitev entitet, ki nastopajo v Blogu
  Uporabnik (avtenticiran in neavtenticiran)
  Novica, ali vnos v blog
  Kategorija vnosa v blog
  Komentar vnosa

  Entitete pripeljejo do podatkovnega modela v katerem hranimo vse podatke o sodelujočih entitetah v okolju.

- Procesi, ki potekajo v blogu in njihovi vhodni ter izhodni podatki:
  uporabniško ime in geslo -> **Prijava v sistem** -> personalizirano okolje
  prijava v sistem -> **Spreminjanje svjoih podatkov** -> novi podatki zapisani
  prijava v sistem -> **Izbira kategorije vnosa** -> **Vnos bloga** -> vnosi se prikažejo na spletni strani
  Komentiranje vnosov -> nov komentar dodan
  vnosi na spletni strani -> **Branje vnosov**

  Določitev procesov je potrebna za pisanje programov (vnaprej vemo kaj hočemo, da program dela).

Informacijska analiza

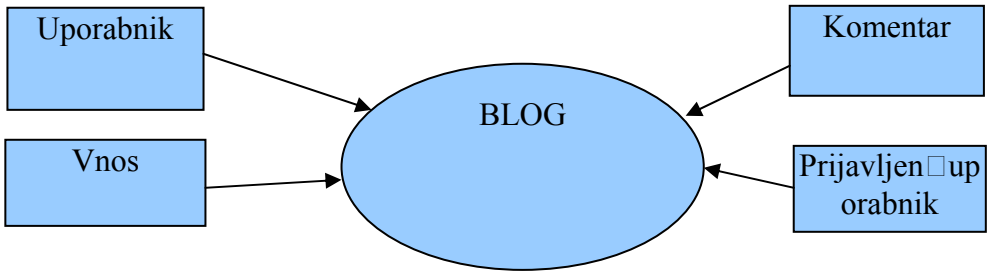
- Anketiranje sodelujoči: izvedeti hočemo, kaj sodelujoči (oziroma bodoči uporabniki) želijo od našega sistema.

  Uporabniki hočejo brati sloge na spletni strani
  Se želijo prijaviti v sistem in imeti personalizirano okolje (jezik)
  Ko je uporabnik avtenticiran, želi vnašati v blog
  Uporabniki želijo komentirati ostale vnose

- Definiranje entitet in njihovih lastnostih (podatki, ki jih bomo potrebovali za delovanje sistema):
  Uporabnik
  ime, priimek, elektronski naslov, jezik, uporabniško ime, geslo
  Vnos
  uporabnik, naslov vnosa, datum vnosa, kategorija, besedilo
  Kategorija vnosov
  kategorija (ime)
  Komentar
  vnos, uporabnik (navtenticiran), datun, besedilo komentarja

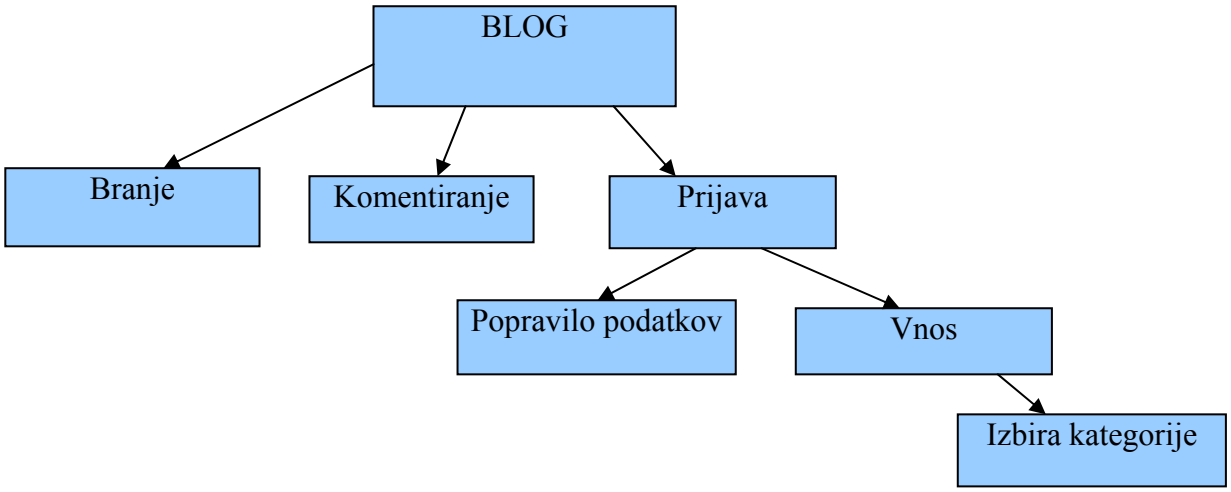
VAJA 2: Diagrami

Diagrami toka podatkov



Poskušajmo narediti bolj podroben diagram! Dodamo procese in podatke, ki v diagramu tečejo.

Strukturni diagram



Ali je še kaj pomembnega?

VAJA 3: ER

Pri Entitetno Relacijskim modelu povežemo med seboj entitete, ki smo jih določili na prvih vajah

Uporabnik
ime, priimek, elektronski naslov, jezik, uporabniško ime, geslo
Vnos
uporabnik, naslov vnosa, datum vnosa, kategorija, besedilo
Kategorija vnosov
kategorija (ime)
Komentar
vnos, uporabnik (navtenticiran), datun, besedilo komentarja

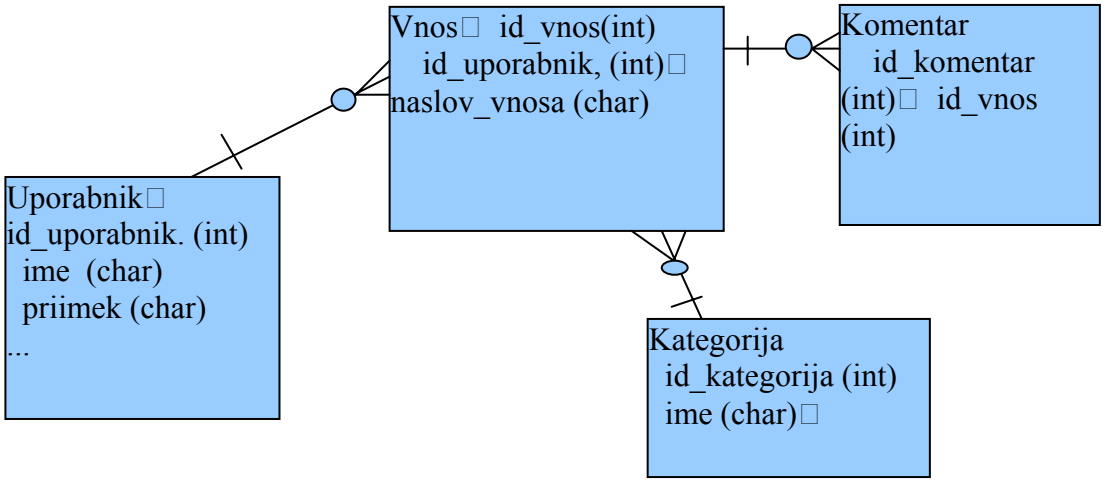
Vnos potrebuje uporabnika in kategorijo
Kategorija ne potrebuje nobene druge entitete
Uporabnik ne potrebuje nobene druge entitete
Komentar potrebuje vnos, ker brez vnosa ne moremo komentarja napisati.

Dodamo ključe in tip podatkov. Vsaka entiteta mora biti taka, da jo lahko točno določimo. Torej, če uporabnik Miha vnese blog z naslovom „Kako naprej“ moramo vedeti povezati Miho in njegov vnos. Kako vemo da ni Jure vnesel blog? Določiti moramo ključe ki bodo vsako posamezni instanco entitete enolično določil.

Uporabnik	Kategorija	Vnos	Komentar
id_uporabnik. (int) ime (char) priimek (char) elektronski (char) naslov (char) jezik (char) uporabniško ime (char) geslo (char)	id_kategorija (int) ime (char)	id_vnos(int) id_uporabnik, (int) naslov_vnosa (char) datum_vnosa (date) id_kategorija (int) besedilo (text)	id_komentar (int) id_vnos (int) uporabnik_ime (char) datum (date) besedilo (text)

Vse identifikatorje smo začeli z id. Kako bi jezik, ki ga ima uporabnik (ko se prijavi si lahko spremeni privzeti jezik v katerem se mu izpisuje stran) dali v novo entiteto?

Relacije med entitetami:
Različne notacije zapisovanja



Kako bi rešili težavo, če bi želeli da vnos v blog spada večim kategorijam? Kaj pa, če bi želeli da ima vnos več avtirjev. Zamislite si, da delate domači sistem za knjige. Kljiga ima lahko poljubno mnogo avtorjev. Ravno tako ima lahko avtor poljubno mnogo knjig. Imamo torej entiteti Knjiga in Avtor. Potrebujemo še vmesno, ki bo povezala obe entiteti.

VAJA 4: tabele in podatkovna baza

Na prejšnjih vajah smo definirali naslednje entitete. Vsaka entiteta predstavlja tabelo v podatkovni bazi:

Uporabnik	Kategorija	Vnos	Komentar
id_uporabnik. (int) ime (char) priimek (char) elektronski (char) naslov (char) jezik (char) uporabniško ime (char) geslo (char)	id_kategorija (int) ime (char)	id_vnos(int) id_uporabnik, (int) naslov_vnosa (char) datum_vnosa (date) id_kategorija (int) besedilo (text)	id_komentar (int) id_vnos (int) uporabnik_ime (char) datum (date) besedilo (text)

Poglejmo primer naših tabel z vnosi za lažje razumevanje:

Uporabnik
id_uporabnik
ime
priimek
...
1
Miha
Kralj
..
2
Janez
Novak
...

Kategorija
id_kategorija
ime
1
Šport
2
Zabava

Vnos
id_vnos
id_uporabnik
Naslov
id_kategorija
...
1
2
Kam sedaj
1
...
2
2
Jutri pri meni
1
...

Komentar
id_komentar
id_vnos
...
1
1

2
1

Kaj lahko povemo o zgornjih tabelah?
- Janez je vnesel v blog dve temi in obe sta kategoriji Šport
- Miha ni vnesel nobene teme
- Prva novica ima dva komentarja medtem ko druga nima nobenega.

Primerjaj te ugotovitve z ER iz prejšnjih vaj.

Podatkovna baza

MySQL

Strežnik MySQL nam omogoča ustvarjanje več različnih baz podatkov, v katerih hranimo in obdelujemo različne podatke. Gre za ustvarjanje relacijskih baz podatkov, ki temeljijo na uporabi razvojnega jezika SQL (Structured Query Language). To je standardni jezik za baze podatkov. Poznavanje SQL-a nam omogoča nemoteno upravljanje baze podatkov MySQL.

Dostop do baze podatkov v MySQL okolju poteka lahko na dva načina:

a) Iz ukazne vrstice
mysql -u uporabnik -p geslo

b) Preko grafičnega vmesnika
http://www.studenti.pef.upr.si/phpmyadmin

Sistemski podatki strežnika MySQL

SHOW DATABASES	Prikaz baz podatkov na strežniku MySQL
SHOW TABLES FROM db	Prikaz tabel v bazi podatkov, v kateri se trenutno nahajamo
SHOW COLUMNS FROM tabela	Prikaz vseh stolpcev (polj) v tabeli
DESCRIBE tabela	Opis vseh polj iz tabele
SHOW GRANTS FOR uporabnik	Prikaz pravic uporabnika
SELECT DATABASE(db)	Zamenjava baze podatkov

Jezik SQL
Predpostavimo da delamo v ukazni vrstici. Tudi preko grafičnih vmesnikov lahko delamo z jezikom SQL, ki ga bomo rabili v ukazni vrstici.

1. Ustvarjanje nove baze podatkov

```
CREATE DATABASE ime_bp;
```
Ukaz ustvari novo bazo podatkov z imenom ime_bp, ki še ne vsebuje tabel.
Primer: CREATE DATABASE upo2007blog;

2. Povezovanje z bazo podatkov

CONNECT ime_bp;
Povežemo se z bazo podatkov ime_bp ter tako lahko to bazo urejamo.
Primer: CONNECT upo2007blog;

3. Brisanje baze podatkov

DROP DATABASE ime_db.
Ukaz pobriše celotno bazo podatkov ime_bp skupaj z njenimi tabelami in podatki.
Primer: DROP upo2007blog;

4. Dodeljevanje pravic drugim uporabnikom

GRANT pravica ON objekt TO uporabnik
Ukaz dodeli pravico pravica za objekt uporabniku uporabnik.
Pravice so lahko:

- ALL PRIVILEGES: uporabnik ima vse provice nad bazo podatkov
- USAGE: uporabnik nima nobene pravice
- ALTER, CREATE, DELETE, DROP, GRANT, INDEX, INSERT, DELETE, SELECT: pravica za izvajanje posameznega ukaza

Primer: GRANT SELECT ON upo2007blog TO upo2007;
(uporabniku upo2007 smo dovolili pravico do pregledovanja na celotni bazi podatkov)

```
GRANT USAGE ON * . * TO 'rp_uporabnik'@'%' IDENTIFIED BY '******';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER,
      CREATE TEMPORARY TABLES ON `rp2007`.* TO 'rp_uporabnik'@'%';
```

5. Ustvarjanje tabel v bazi podatkov

```
CREATE TABLE tab (ime_1stolpca lastnosti,
                  ime_2stolpca lastnosti,
                  ...

                  ime_Nstolpca lastnosti);
```

Ukaz ustvari tabelo z imenom tab, ki vsebuje N stolpcev tako kot smo jih definirali.
Osnovne lastnosti stolpcev (tj. opisi prilastkov) so lahko:

- tip podatka,
- obveznost podatka,
- privzeta vrednost podatka,
- informacija ali gre za ključ.

- Tip podatka je potrebno obvezno določiti za vsak stolpec posebej. Osnovni tipi podatkov:
  - INTEGER: celo število od –231 do 231,
  - FLOAT(N,M): število z največ N znaki pred in največ M znaki po decim. vejici,
  - VARCHAR(N): besedilo največ dolžine N (N znakov)
  - CHAR(N): besedilo natančno dolžine N (N znakov)

- TEXT: Polje z največ 216–1 znaki,
- MEDIUMTEXT: Polje z največ 224– 1 znaki,
- DATE: datum,
- TIMESTAMP: čas (z vključenim datumom).

Ostale lastnosti podatkov niso obvezne in jih določimo le po želji.

- Obveznost podatka:
  - NULL: vrednost v polju je lahko tudi prazna,
  - NOT NULL: podatek v polju je obvezen.
- Privzeta vrednost podatka:
  - AUTO_INCREMENT: lahko uporabimo samo v primeru stolpcev, kjer je tip podatka INTEGER (vrednost v polju se sama povečuje z vnosom novega zapisa v tabelo),
  - DEFAULT: določimo privzeto vrednost polja (ob vnosu zapisa v tabelo se polje s to lastnostjo avtomatsko zapolne s privzeto vrednostjo).
- Informacija ali gre za ključ:
  - PRIMARY KEY: polje je tudi glavni ključ tabele (polje, ki je primarni ključ ne sme biti nikoli prazno, torej vedno zahtevamo še lastnost NOT NULL, in mora biti enolično),
  - INDEX: index lahko vključuje več polj hkrati in omogoča ustvarjanje »indeksa« nad podatki v tabeli po izbranih polji, tako da je iskanje podatkov po teh poljih bistveno hitrejše.

```
 CREATE TABLE `blog` (
     `id_blog` INT NOT NULL AUTO_INCREMENT ,
     `id_uporabnik` INT NOT NULL AUTO_INCREMENT ,
     `naslov` VARCHAR( 50 ) NOT NULL ,
     `datum` DATE NOT NULL ,
     `id_kategorija` VARCHAR( 20 ) NOT NULL ,
     `prispevek` LONGTEXT NOT NULL ,
     PRIMARY KEY ( `id` )
  )
CREATE TABLE `komentar` (
     `id_komentar` INT NOT NULL AUTO_INCREMENT ,
     `id_blog` INT NOT NULL ,
     `ime` VARCHAR( 50 ) NOT NULL ,
     `e-naslov` VARCHAR( 30 ) NOT NULL ,
     `datum` DATE NOT NULL ,
     `komentar` MEDIUMTEXT NOT NULL ,
     PRIMARY KEY ( `id_komentar` )
  )
 CREATE TABLE `uporabnik` (
     `id_uporabnik` INT NOT NULL AUTO_INCREMENT ,
     `ime` VARCHAR( 50 ) NOT NULL ,
     `priimek` VARCHAR( 50 ) NOT NULL ,
     `e-naslov` VARCHAR( 30 ) NOT NULL ,
     `naslov` VARCHAR( 30 ) NOT NULL ,
     `jezik` VARCHAR( 30 ) NOT NULL ,
     `up-ime` VARCHAR( 30 ) NOT NULL ,
     `geslo` MEDIUMTEXT NOT NULL ,
     PRIMARY KEY ( `id_uporabnik` )
  )
CREATE TABLE `komentar` (
     `id_kategorija` INT NOT NULL AUTO_INCREMENT ,
     `ime` VARCHAR( 50 ) NOT NULL ,
     PRIMARY KEY ( `id_kategorija` )
  )
```

Enkrat ko smo izdelali tabel bomo vse ostalo delali z našim programom in pregledovali preko uporabniškega vmesnika.

6. Brisanje tabel v bazi podatkov

DROP TABLE tab;
Ukaz pobriše tabelo z vsemi njenimi podatki.
Primer: DROP TABLE zaposleni;

7. Spreminjanje tabel v bazi podatkov

Lahko dodajamo nova polja v tabelo:
ALTER TABLE tab
ADD (ime_stolpca lastnosti,
....
ime_stolpca lastnosti)

Lahko spreminjamo obstoječa polja v tabeli:
ALTER TABLE tab
MODIFY (ime_stolpca nove_lastnosti,
....
ime_stolpca lastnosti)

Lahko brišemo obstoječa polja v tabeli:
ALTER TABLE tab
DROP (ime_stolpca nove_lastnosti,
....
ime_stolpca lastnosti)

Lahko preimenujemo obstoječa polja v tabeli:
ALTER TABLE tab
RENAME (ime_stolpca nove_lastnosti,
....
ime_stolpca lastnosti)

Primer:
ALTER TABLE zaposleni ADD (OsDohodek float(10,2));
ALTER TABLE zaposleni MODIFY OsDohodek decimal(10,2) ;

8. Vnos podatkov v tabele

INSERT INTO tabela (ime_1stolpca, ime_2stolpca, ...) VALUES (vrednost1, vrednost2, ...)
Ukaz vpiše nov zapis v tabelo in sicer v tista polja, ki smo jih navedli. Ukaz seveda ne uspe, če v polja z lastnostjo NOT NULL nismo vnesli nobene vrednosti.

Primer:
INSERT INTO zaposleni (Priimek,Ime,DatumRojstva) VALUES ('Hmelj','Tone','1968-05.23');

9. Poizvedovanje po tabelah
a. V eni tabeli

SELECT seznam_polj FROM tabela WHERE pogoj
Ukaz izbere iz tabele tabela samo stolpce iz seznam_polj za tiste zapise, ki ustrezajo pogoju pogoj.

Primer: SELECT Ime, Priimek FROM zaposleni WHERE starost>45;

b. V več tabelah hkrati

SELECT seznam_polj FROM seznam_tabela WHERE pogoj
Ukaz izbere iz vseh tabel iz seznam_tabel samo stolpce iz seznam_polj za tiste zapise, ki ustrezajo pogoju pogoj. Pri tem moramo tabele iz seznam_tabel smiselno povezati med seboj.

Primer:
Naj obstaja še tabela Prevozi, ki hrani zapise o načinih prevoza zaposlenih v službo:

```
CREATE TABLE  prevozi (Id integer(5) not null auto_increment primary key,
 IdZaposlenega integer(5),  NacinPrevoza varchar(100));
```

Vstavimo nekaj zapisov za zaposlenega Hmelj Tone, ki ima svoj Id=1 (tega nismo nastavili, baza je avtomatsko zacela z 1, ker je polje Id v tabeli zaposleni auto_increment),

```
INSERT INTO prevozi (IdZaposlenega,NacinPrevoza) VALUES (1, 'avtobus');
INSERT INTO prevozi (IdZaposlenega,NacinPrevoza)VALUES (1, 'avtomobil');
INSERT INTO prevozi (IdZaposlenega,NacinPrevoza) VALUES (1, 'peš');
```

V kolikor želimo dobiti vse načine prevoza in še ime in priimek zaposlenega napišemo naslednj
SQL (ne pozabimo v WHERE pogoju povezati obe tabeli):

```
SELECT zaposleni.Ime, zaposleni.Priimek, prevozi.NacinPrevoza
FROM zaposleni, prevozi WHERE zaposleni.Id=prevozi.IdZaposlenega;
```

Stavek SELECT lahko izkoristimo tudi za vnos podatkov v tabelo:
INSERT INTO tabela (seznam polj) SELECT seznam_polj FROM seznam_tabel
WHERE pogoj

10. Brisanje podatkov v tabelah

DELETE FROM tabela WHERE pogoj
Ukaz pobriše vse zapise v tabeli, ki ustrezajo določenemu pogoju.

Primer: pobriše vse prevoze, katerih opis se prične z pe

```
DELETE FROM prevozi WHERE NacinPrevoza like 'pe%';
```

11. Spreminjanje podatkov v tabelah

UPDATE tabela SET polje=nova_vrednost WHERE pogoj
Ukaz spremeni vrednost v polju za izbrano tabelo vendar le za tiste zapise, ki ustrezajo pogoju. Če
pogoja ne specificiramo, se spremeni vrednost polja v vseh zapisih tabele.

Primer: zaposlenim, ki so starejši od 40 let spremeni osebni dohodek (polje OD)
v vrednost 250000.

```
UPDATE zaposleni SET OD=250000 WHERE Starost>40;
```

VAJA 5: HTML, CSS, PHP

Vnos dnevniškega zapisa (zasnova BLOG sistema)
Obrazce pošiljamo z metodo POST. Drugače pa spremenljivke pošiljamo preko URL naslova z metodo GET. Podrobnosti si bomo ogledali na vajah ki sledijo.

GET - Podatki se pošljejo z URL naslovom (ki je lahko tudi omejen). Uporabljamo ga, ko z njegovo uporabo ne spreminjamo okolja. Primer: iskalniki. Ko vnesemo nekaj v polje in kliknemo „Išči“ s tem ne spremenimo ničesar nikjer.

POST - Podatki poslani preko zaglavja, količina ni omejena. Uporabljamo, ko poslani podatki spremenijo nekaj: bazo podatkov, pošljejo el. naslov.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=UTF-8" http-equiv="content-type">
  <title>Vnos v moj dnevnik</title>
  <link href="blog.css" rel="stylesheet" type="text/css">
</head>
<body>
  <form method="post" action="obdelaj.php" name="prispevek">
    <table width="100%" border="1" cellpadding="2" cellspacing="2">
      <tr>
        <td align="right">Avtor prispevka</td><td> </td>
      </tr>
      <tr>
        <td align="right">Ime: </td><td><input name="ime"></td>
      </tr>
      <tr>
        <td align="right">Priimek: </td><td><input name="priimek"></td>
      </tr>
      <tr>
        <td align="right">Elektronski naslov: </td>
        <td><input name="email"></td>
      </tr>
      <tr>
        <td valign="top" align="right">Prispevek: </td>
        <td><textarea cols="25" rows="10" name="prispevek"></textarea></td>
      </tr>
      <tr>
        <td align="right">Kategorija: </td>
        <td>
          <select name="kategorija">
            <option value="sport">&Scaron;port</option>
            <option value="politika">Politika</option>
            <option value="zabava">Zabava</option>
            <option value="studij">&Scaron;tudij</option>
          </select>
        </td>
      </tr>
      <tr>
        <td align="right"> </td><td><input type="submit" value="Objavi"> </td>
      </tr>
    </table>
  </form>
</body>
</html>
```

CSS:
- Ločitev oblike in vsebine
- Manjši čas nalaganja
- Večja kontrola nad vsebino
- Enakost dokumentov

CSS dokument, ki da zgornji datoteki obliko

```
body {
font-size: 70%;
color:#000000;
background-color:#FFD833;
margin:0px;
}

body, p, h1, h2, h3, table, td, th, ul, ol, textarea, input, a {
font-family: verdana,helvetica,arial,sans-serif;
}

td.menu, td.vsebina {
background-color:#ffffff;
padding:10px;
}

td.locnica {
background-color:#CC9933;
}

div#menu {
font-size:100%;
font-weight:normal;
color:#000000;
}

div#glava {
font-size:120%;
font-variant: small-caps;
letter-spacing: 0.3cm;
font-weight:bold;
color:#660000;
margin-top:10px;
margin-bottom:5px;
text-align: center;
}

div#noga {
font-size:80%;
color:#333333;
background-color:#CC9933;
padding: 2px;
}

a:link    {color:#900B09; background-color:transparent}
a:visited {color:#900B09; background-color:transparent}
a:active  {color:#FF0000; background-color:transparent}
a:hover   {color:#FF0000; background-color:transparent; text-decoration: none;}

div#prispevek {
font-size:100%;
font-weight:normal;
color:#000000;
background-color:#FFFFCC;
margin:20px;
padding:10px;
border-style: solid;
border-color: #C0C0C0;
border-width: 1px
}
```

VAJA 6: HTML, PHP in optimizacija strani

HTML smo spoznali v prvem semestru. Vendar bomo uporabljali tudi druge značke, ki jih bomo sproti razložili. Ravno tako bomo sproti razlagali PHP in njegove funkcije.

Poglejmo si obliko naše spletne strani:

HTML Zaglavje

Prostor, kjer se nahaja logotip	
Meni spletne strani	
Noga kjer damo (c) in pravice, tem kakšne manj pomembne a vendar potrebne podatke	

Edini del, ki se spreminja je sredinski (prazen del). Ta vsebuje vsebino glede na to, kje se nahajamo. En način optimizacije je, da spletno stran zgradimo tako, da posamezne dele, ki bodo stalni na vseh spletnih straneh, damo v svoje datoteke, ki jih nato s PHP funkcijo include() vstavljamo po potrebi.

Poglejmo si sedaj vse naše datoteke.
Datoteke glava.php, logo.php, meni.php in noga.php kličemo na vseh ostalih straneh

glava.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=UTF-8" http-equiv="content-type">
  <title>Naš dnevnik</title>
  <link href="blog.css" rel="stylesheet" type="text/css">
</head>
<body>

<?php // ker se priklapljamo na bazo na več straneh to vključimo na vse strani
$server="localhost";
$uporabnik="rp_uporabnik";
$geslo="2007rp";
$baza="rp2007";
$db = mysql_connect($server,$uporabnik,$geslo);
mysql_select_db($baza,$db)or die( "Unable to select database");
?>
```

logo.php

```
<table border="0" width="100%">
  <tr><td><div id="glava">
       ..-N|a|š   s|p|l|e|t|n|i   d|n|e|v|n|i|k-..
      </div></td></tr>
  <tr><td class="locnica"> </td></tr>
</table>
```

meni.php

```
<div id="menu">

<a href="index.php">Moj blog</a><br>
```

```
<a href="vnos.php">Vnos prispevka</a><br><br>


<?php

$month=date("m");
$year=date("Y");

for($j=$year;$j>($year-3);$j--){
  echo '<hr><p>'.$j.'</p><p align="center">';
  if ($j==$year) {
    for ($i=$month; $i>0 ; $i--) {
      //posiljanje spletni strani podatke preko URL - GET metoda
      echo '|-<a href="index.php?year='.$j.'&month='.$i.'">'.$i.'</a>-';
      if($i==9||$i==5) echo '|<br>';
    }
    echo '|';
  } else {
    for ($i=12; $i>0 ; $i--) {
      //posiljanje spletni strani podatke preko URL - GET metoda
      echo '|-<a href="index.php?year='.$j.'&month='.$i.'">'.$i.'</a>-';
      if($i==9||$i==5) echo '|<br>';
    }
    echo '|';
  }
  echo '</p><hr>';
}
?>

</div>
```

noga.php

```
  <table border="0" width="100%">
   <tr><td colspan="2">
       <div id="noga">&copy; RP - Vse pravice pridržane</div></td>
   </tr>
  </table>
</body>
</html>
```

Vse ostale datoteke uporabljajo skupne datoteke tako, da jih na pravih mestih kličejo. Torej spletne strani gradimo sproti, ko uporabnik kliče posamezne datoteke.

index.php – glavna spletna stran, ki se odpre, ko uporabnik pride na blog

```
<?php include('glava.php'); ?>
<?php include('logo.php'); ?>

  <table border="0" width="100%">
   <tr><td class="menu" width="150" valign="top">
       <?php include('meni.php'); ?></td>
    <td class="vsebina" valign="top">
      <?php

      if ($_POST["submit"] && $_POST["id_blog"]) {
        //vnesi komentar, če kdo klikne na gumb
        $datum=date("Y-m-d"); //danasnji datum
        $sql="INSERT INTO komentar
              VALUES ('','".$_POST["id_blog"]."',
                      '".$_POST["ime"]."',
```

```
                    '".$_POST["email"]."',
                    '$datum',
                    '".$_POST["komentar"]."')";
      $result = mysql_query($sql,$db);
    }

    //kaj izpisemo je odvisno od kod prihajamo (na kaj smo kliknili)
    if ($_POST["id_blog"]) { // ce je poslan id zapisa ga najdi
      $sql="SELECT * FROM blog WHERE id_blog=".$_POST["id_blog"];
    } elseif ($_GET["id_blog"]) {// ce je poslan id zapisa ga najdi
      $sql="SELECT * FROM blog WHERE id_blog=".$_GET["id_blog"];
    } elseif ($_GET["month"]&&$_GET["year"]) { // ce imamo mesec in leto
      //najdemo vse zapise tega leta in meseca
      $sql="SELECT * FROM blog WHERE datum
            LIKE '".$_GET["year"]."-".$_GET["month"]."%'";
    } else { // ce ne pa najdi zadnjega zapisanega
      $sql="SELECT * FROM blog ORDER BY datum DESC LIMIT 1";
      $prazen=1; // tega potrebujemo za izpis komentarjev zadnjega
    }

    $result = mysql_query($sql,$db);
    while ($myrow = mysql_fetch_row($result)) { //dokler je se kaj najdenega
      echo '<div id="noga">'.$myrow[5].'</div>';
      echo '<div id="prispevek">
        <a href="mailto:'.$myrow[3].'">'.$myrow[1].' '.$myrow[2].'</a><br>';
      echo '<a href="index.php?id_blog='.$myrow[0].'">'.$myrow[4].'</a>
        <font color="grey">('.$myrow[6].')</font><br>';
      echo '<br>'.$myrow[7].'
        </div>';

      if ($_POST["id_blog"] || $_GET["id_blog"] || $prazen==1) {
        // izpis komentarjev
        $db2 = mysql_connect($server,$uporabnik,$geslo);
        mysql_select_db($baza,$db2)or die( "Unable to select database");
        $sql2="SELECT * FROM komentar WHERE id_blog=".$myrow[0];
        $result2 = mysql_query($sql2,$db2);
        while ($myrow2 = mysql_fetch_row($result2)) {
          echo '<hr><p><a href="mailto:'.$myrow2[3].'">'.$myrow2[2].'</a>
               ('.$myrow2[4].'): '.$myrow2[5].'</p>';
        }
        // obrazec za vpis komentarja
        echo '<hr>
           <form method="post" action="index.php">
           <input type="hidden" name="id_blog" value="'.$myrow[0].'">
           Ime: <input name="ime"> <br>
           E-naslov: <input name="email"> <br>
           Komentar: <br>
           <textarea cols="50" rows="4" name="komentar"></textarea><br>
           <input type="submit" name="submit" value="Objavi">';
      }
    }

    mysql_close();
    ?>
    </td>
   </tr>
  </table>

<?php include('noga.php'); ?>
```

vnos.php – stran z obrazcem za vnos v blog;

```
<?php include('glava.php'); ?>
<?php include('logo.php'); ?>

  <table border="0" width="100%">
   <tr><td class="menu" width="150" valign="top">
       <?php include('meni.php'); ?></td>
    <td class="vsebina">
    <form method="post" action="obdelaj.php" name="prispevek">
    <table width="100%" border="0" cellpadding="2" cellspacing="2">
      <tr>
        <th align="right">Avtor prispevka</td><td> </th>
      </tr>
      <tr>
        <td align="right">Ime: </td><td><input name="ime"></td>
      </tr>
      <tr>
        <td align="right">Priimek: </td><td><input name="priimek"></td>
      </tr>
      <tr>
        <td align="right">Elektronski naslov: </td>
        <td><input name="email"></td>
      </tr>
      <tr>
        <td align="right">Naslov prispevka: </td>
        <td><input name="naslov"></td>
      </tr>
      <tr>
        <td align="right">Kategorija: </td>
        <td>
          <select name="kategorija">
            <option value="sport">&Scaron;port</option>
            <option value="politika">Politika</option>
            <option value="zabava">Zabava</option>
            <option value="studij">&Scaron;tudij</option>
          </select>
        </td>
      </tr>
      <tr>
        <th valign="top" align="right">Prispevek: </th>
        <td><textarea cols="25" rows="10" name="prispevek"></textarea></td>
      </tr>
      <tr>
        <td align="right"> </td>
        <td><input type="submit" name="submit" value="Objavi"> </td>
      </tr>
    </table>
    </form>
    </td>
   </tr>
  </table>

<?php include('noga.php'); ?>
```

obdelaj.php – stran, ki doda vnos v bazo podatkov

```
<?php include('glava.php'); ?>
<?php include('logo.php'); ?>
```

```
<table border="0" width="100%">
 <tr><td class="menu" width="150" valign="top">
       <?php include('meni.php'); ?></td>
  <td class="vsebina">
  <?php

      if ($_POST["submit"]=="Objavi") {
         // ce smo kliknili na gumb - da kdo ne pride
         // direktno na spletno stran brez
         //izpisovanje napak
         if (trim($_POST["ime"])=="" ||
             trim($_POST["priimek"])=="" ||
             trim($_POST["email"])=="" ||
             trim($_POST["naslov"])=="" ||
             trim($_POST["prispevek"])=="") { //ce je kaksno polje prazno
                $error[]="Niste vnesli vseh podatkov. Vsa polja so obvezna.";
         }
         if ((strlen($_POST["ime"])<2) || (strlen($_POST["priimek"])<2)) {
                // ce sta ime ali priimek krajsa od dveh znakov
                $error[]="Ime ali priimek sta krajša od dveh znakov..";
         }
         if (!ereg("([A-Z]{1})[[:alpha:]]", $_POST["ime"])) {
                $error[]="Ime vsebuje ne-alfanumerične znake.";
         }
         if (!ereg("([A-Z]{1})[[:alpha:]]", $_POST["priimek"])) {
                $error[]="Priimek vsebuje ne-alfanumerične znake.";
         }
         if (!ereg("^[-!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+"."@".
                   "[-!#$%&\'*+\\/0-9=?A-Z^_`a-z{|}~]+\.".
                   "[-!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+$", $_POST["email"])){
               $error[]="Elektrosnki naslov ni pravilne oblike.";
         }

         if (count($error) == 0) { // ce ni napak izpisi podatke
            echo 'Vnesli ste naslednje podatke: <br>';
            echo '<div id="prispevek">Avtor:
                 '.$_POST["ime"].' '.$_POST["priimek"].'<br>';
            echo 'elektronski naslov: '.$_POST["email"].'<br>';
            echo 'Naslov: '.$_POST["naslov"].'<br>';
            echo 'Kategorija: '.$_POST["kategorija"].'<br>';
            echo 'Prispevek:<br>'.$_POST["prispevek"].'<div>';

            $datum=date("Y-m-d"); //danasnji datum
            $sql="INSERT INTO blog
                    VALUES ('','".$_POST["ime"]."',
                            '".$_POST["priimek"]."',
                            '".$_POST["email"]."',
                            '".$_POST["naslov"]."',
                            '$datum',
                            '".$_POST["kategorija"]."',
                            '".$_POST["prispevek"]."')";
             echo $sql;
             $result = mysql_query($sql,$db);
             mysql_close();

         } elseif (is_array($error)) { // izpisi vse najdene napake
             echo '<p>';
             while (list($error_id,$error_text)=each($error)) {
               echo $error_text.'<br>';
               unset ($error[$error_id]);
             }
             echo '</p>';
         }
      } else {
```

```
            echo "Na spletno stran niste prišli preko obrazca
                  za vnos objav v dneviški sistem";
          }
      ?>
      </td>
     </tr>
    </table>

<?php include('noga.php'); ?>
```

VAJA 7: avtentikacija

Ker želimo, da se uporabnik, preden vnese nov vnos v blog, avtenticira moramo poskrbeti za avtentikacijski sistem. To je lahko kar Apache (spletni strežnik) .htaccess datoteka v našem direktoriju, kjer se nahaja spletna stran, ki jo želimo zaščititi. Lahko avtenticiramo uporabnike iz LDAP imenika (light directory access protocol) ali pa preko mySQL podatkovne baze. Apachejev .htaccess se ravno tako lahko avtenticira iz LDAP imenika, poleg privzete lastne datoteke z gesli. Za uporabnike lahko torej poskrbimo sami, ali pa uporabimo drugi že obstoječi vir. Konec koncev lahko uporabimo tudi IMAP ali POP poštne strežnike.

Primer .htaccess datoteke

datoteka .htaccess

```
AuthUserFile /home/osebje/pef/mkljun/.htpasswd
AuthGroupFile /dev/null
AuthName Potrebna prijava
AuthType Basic
require valid-user
```

Ko dodamo prvega uporabnika uporabimo -c stikalo (create). Za naslednje uporabnike to ni potrebno. ~$ `htpasswd -c .htpasswd` *`username`*

Primer .htaccess datoteke, ki se avtenticira iz LDAP imenika

```
AuthType               basic
AuthName               "Potrebna prijava"
AuthLDAPAuthoritative  on
AuthLDAPEnabled        on
AuthLDAPURL            ldap://localhost/dc=upr,dc=si
AuthLDAPGroupAttributeIsDN off
AuthLDAPGroupAttribute "memberUid"
require valid-user

(require group cn=mara,ou=Group,dc=upr,dc=si)
```

Primer php skripte, ki se avtenticira iz LDAP imenika
Seje v nasprotju s piškotki, se nahajajo na strežniku. Piškotki pa na računalniku uporabnika. Oboje uporabljamo za shranjevanje uporabnikovih podatkov, s katerimi porsonaliziramo spletne strani. Le da se seja ob zaprtju brskalnika zbriše. Piškotek pa ostane na računalnik uporabnika do naslednjega obiska nedotaknjen. Seje so bolj varne (skorajda nemogoče je vdreti v sejo), medtem ko so piškotki pisani v besedilnih datotekah in jih je zato lahko prebrat. V sejah ne moremo trajno hranit podatkov, v piškotkih pa. Pri avtentikaciji bomo uporabljali sejo.

```
<?
  session_start();
  if(!session_is_registered("language_id")) {
     session_register("language");
     $_SESSION['language_id']="slo.inc";
  }


  if ($_POST["submit"]) {
    $conn= ldap_connect("localhost",389);
    ldap_set_option( $conn, LDAP_OPT_PROTOCOL_VERSION, 3);
    ldap_set_option( $conn, LDAP_OPT_REFERRALS, 0);
    $ldap_pot="ou=PEFStudenti,ou=People,dc=upr,dc=si";
    $res = ldap_bind($conn,"uid=".$_POST["username"].",".$ldap_pot,
```

```
$_POST["pass"]);
    if ($res){
      $sr=ldap_search($conn, "dc=upr,dc=si","(uid=".$_POST["username"].")");
      $info = ldap_get_entries($conn, $sr);
      if (isset($info[0])){
        $prijavljen=true;
        session_register("prijavljen");
        session_register("username");
        session_register("email");
        $_SESSION['prijavljen']=true;
        $_SESSION['username']=$_POST["username"];
        $_SESSION['email']=$info[0]["mail"][1];
        $_SESSION['name']=$info[0]["cn"][0];
      }
    }
  }

  if (!session_is_registered("username")) {
        echo '<hr>
        <form method="post" action="'.$_SERVER['PHP_SELF'].'">
        Uporabniško ime: <input name="username"> <br>
        GEslo: <input type="password" name="pass"> <br>
        <input type="submit" name="submit" value="Prijavi">';
  } else {
    echo $_SESSION['name']." ".$_SESSION['username']." ".$_SESSION['email'];
  }

?>
```

Popravimo sedaj našo **vnos.php** in **glava.php** datoteko. Če uporabnik ni registriran, ne more vnesti bloga. Torej se mora prej registrirati.

glava.php

```
<?php
//začnemo sejo na vsaki strani, da lahko dostopamo do spremenljivk v seji
session_start();
if(!session_is_registered("language_id")) {//če jezik ni izbran izberemo slo
   session_register("language");
   $_SESSION['language_id']="slo.inc";
}
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=UTF-8" http-equiv="content-type">
  <title>Naš dnevnik</title>
  <link href="blog.css" rel="stylesheet" type="text/css">
</head>
<body>

<?php
// ker se priklapljamo na bazo na več straneh to vključimo na vse strani
$server="localhost";
$uporabnik="rp_uporabnik";
$geslo="2007rp";
$baza="rp2007";
$db = mysql_connect($server,$uporabnik,$geslo);
mysql_select_db($baza,$db)or die( "Unable to select database");
?>
```

vnos.php – stran z obrazcem za vnos v blog;

```
<?php include('glava.php'); ?>
<?php include('logo.php'); ?>

  <table border="0" width="100%">
   <tr><td class="menu" width="150" valign="top">
       <?php include('meni.php'); ?></td>
    <td class="vsebina">

    <?php //obrazec izpisemo le ce je uporabnik prijavljen

    if ($_POST["submit"]) {
      $conn= ldap_connect("localhost",389);
      ldap_set_option( $conn, LDAP_OPT_PROTOCOL_VERSION, 3);
      ldap_set_option( $conn, LDAP_OPT_REFERRALS, 0);
      $ldap_pot="ou=PEFGosti,ou=People,dc=upr,dc=si";
      $res = ldap_bind($conn,"uid=".$_POST["username"].",".$ldap_pot,
$_POST["pass"]);
      if ($res){
        $sr=ldap_search($conn, "dc=upr,dc=si","(uid=".$_POST["username"].")");
        $info = ldap_get_entries($conn, $sr);
        if (isset($info[0])){
          session_register("prijavljens");
          session_register("usernames");
          session_register("emails");
          session_register("names");
          $_SESSION['prijavljen']=true;
          $_SESSION['usernames']=$_POST["username"];
          $_SESSION['emails']=$info[0]["mail"][1];
          $_SESSION['names']=$info[0]["cn"][0];
        }
      }
    }

    if (!session_is_registered("usernames")) {
        echo '<hr>
        <form method="post" action="'.$_SERVER['PHP_SELF'].'">
        <table border="0">
          <tr><td>Uporabniško ime: </td><td><input name="username"></td></tr>
          <tr><td>Geslo: </td><td><input type="password" name="pass"></td></tr>
          <tr><td>  </td><td>
              <input type="submit" name="submit" value="Prijavi"></td></tr>
        </table>';
    } else {
    ?>
    <form method="post" action="obdelaj.php" name="prispevek">
    <table width="100%" border="0" cellpadding="2" cellspacing="2">
      <tr>
        <th align="right">Avtor prispevka</td><td> </th>
      </tr>
      <tr>
        <td align="right">Ime: </td><td>
            <input name="ime" value="<?php echo $_SESSION['names'] ?>"></td>
      </tr>
      <tr>
        <td align="right">Priimek: </td><td>
            <input name="priimek" value="<?php echo $_SESSION['names'] ?>"></td>
```

```
      </tr>
      <tr>
        <td align="right">Elektronski naslov: </td>
        <td><input name="email" value="<?php echo $_SESSION['emails'] ?>"></td>
      </tr>
      <tr>
        <td align="right">Naslov prispevka: </td>
        <td><input name="naslov"></td>
      </tr>
      <tr>
        <td align="right">Kategorija: </td>
        <td>
          <select name="kategorija">
            <option value="sport">&Scaron;port</option>
            <option value="politika">Politika</option>
            <option value="zabava">Zabava</option>
            <option value="studij">&Scaron;tudij</option>
          </select>
        </td>
      </tr>
      <tr>
        <th valign="top" align="right">Prispevek: </th>
        <td><textarea cols="25" rows="10" name="prispevek"></textarea></td>
      </tr>
      <tr>
        <td align="right"> </td>
        <td><input type="submit" name="submit" value="Objavi"> </td>
      </tr>
    </table>
    </form>
    <?php
    }
    ?>
    </td>
   </tr>
  </table>

<?php include('noga.php'); ?>
```

meni.php – odjava

```
<div id="menu">

<a href="index.php">Moj blog</a><br>
<a href="vnos.php">Vnos prispevka</a><br><br>

<?php
//odjava iz sistema, ce smo prijavljeni

if ($_SESSION['usernames']!="") {
  echo '<a href="'.$_SERVER['PHP_SELF'].'?odjava=1">Odjava</a><br>';
}
if ($_GET["odjava"]==1){
  session_unregister("usernames");
  session_unregister("emails");
  session_unregister("names");
  unset($_SESSION['usernames']); //zbrišemo tudi vrednost
  unset($_SESSION['emails']);
  unset($_SESSION['names']);
  unset($usernames); //za slučaj če imamo GLOBALS on
  unset($emails);
  unset($names);
  $_SESSION['prijavljen']=false;
```

```
  $_GET["odjava"]=0; //sicer ni potrebno, pa vendar
}

//izpis koledarjev po mescih
$month=date("m");
$year=date("Y");

for($j=$year;$j>($year-3);$j--){
  echo '<hr><p>'.$j.'</p><p align="center">';
  if ($j==$year) {
    for ($i=$month; $i>0 ; $i--) {
      //posiljanje spletni strani podatke preko URL - GET metoda
      echo '|-<a href="index.php?year='.$j.'&month='.$i.'">'.$i.'</a>-';
      if($i==9||$i==5) echo '|<br>';
    }
    echo '|';
  } else {
    for ($i=12; $i>0 ; $i--) {
      //posiljanje spletni strani podatke preko URL - GET metoda
      echo '|-<a href="index.php?year='.$j.'&month='.$i.'">'.$i.'</a>-';
      if($i==9||$i==5) echo '|<br>';
    }
    echo '|';
  }
  echo '</p><hr>';
}
?>

</div>
```

VAJA 8: lokalizacija

Izdelali bomo preprost primer lokalizacije našega bloga s pomočjo sej in besedilnih datotek, ki hranijo vse nize, ki se pojavljajo na strani.

slo.inc.php

```
<?php
$GLOBAL['Nas_Dnevnik']= "Naš dnevnik";
$GLOBAL['logo']="..-
N|a|š   s|p|l|e|t|n|i   d|n|e|v|n|i|k-..";
$GLOBAL['Moj_blog']="Moj blog";
$GLOBAL['Vnos_prispevka']="Vnos_prispevka";
$GLOBAL['Odjava']="Odjava";
$GLOBAL['Noga']="&copy; RP - Vse pravice pridržane";
$GLOBAL['Avtor_prispevka']="Avtor prispevka";
$GLOBAL['Ime']="Ime";
$GLOBAL['Priimek']="Priimek";
$GLOBAL['Elektronski_naslov']="Elektronski naslov";
$GLOBAL['Naslov_prispevka']="Naslov prispevka";
$GLOBAL['Kategorija']="Kategorija";
$GLOBAL['Sport']="&Scaron;port";
$GLOBAL['Politika']="Politika";
$GLOBAL['Zabava']="Zabava";
$GLOBAL['Studij']="&Scaron;tudij";
$GLOBAL['Prispevek']="Prispevek";
$GLOBAL['Objavi']="Objavi";
$GLOBAL['Vnesli_ste']="Vnesli ste naslednje podatke";
$GLOBAL['Error_6']="Na spletno stran niste prišli preko obrazca za vnos objav v
dneviški sistem";
$GLOBAL['Error1']="Niste vnesli vseh podatkov. Vsa polja so obvezna.";
$GLOBAL['Error2']="Ime ali priimek sta krajša od dveh znakov.";
$GLOBAL['Error3']="Ime vsebuje ne-alfanumerične znake.";
$GLOBAL['Error4']="Priimek vsebuje ne-alfanumerične znake.";
$GLOBAL['Error5']="Elektrosnki naslov ni pravilne oblike.";
$GLOBAL['Komentar']="Komentar";
?>
```

Sedaj je potrebno popraviti vse datoteke, ki vsebujejo prevedene nize

en.inc.php

```
<?php
$GLOBAL['Nas_Dnevnik']= "Our blog";
$GLOBAL['logo']="..-
O|u|r   w|e|b   l|o|g   B|l|o|g-..";
$GLOBAL['Moj_blog']="My blog";
$GLOBAL['Vnos_prispevka']="New entry";
$GLOBAL['Odjava']="Logout";
$GLOBAL['Noga']="&copy; RP - all rights reserved";
$GLOBAL['Avtor_prispevka']="Author";
$GLOBAL['Ime']="Name";
$GLOBAL['Priimek']="Surname";
$GLOBAL['Elektronski_naslov']="E-mail";
$GLOBAL['Naslov_prispevka']="Title";
$GLOBAL['Kategorija']="Category";
$GLOBAL['Sport']="Sport";
$GLOBAL['Politika']="Politics";
$GLOBAL['Zabava']="Fun";
$GLOBAL['Studij']="Study";
$GLOBAL['Prispevek']="Entry";
$GLOBAL['Objavi']="Submit";
$GLOBAL['Vnesli_ste']="You enetered following data";
$GLOBAL['Error_6']="If you see this messages you did not eneter this site properly";
$GLOBAL['Error1']="All fields must be filled in.";
```

```
$GLOBAL['Error2']="Your name or surname is only onecharacter long.";
$GLOBAL['Error3']="Your name contains numeric or other characters which are not
allowed.";
$GLOBAL['Error4']="Your surname contains numeric or other characters which are not
allowed.";
$GLOBAL['Error5']="Your email is not right format.";
$GLOBAL['Komentar']="Comment";
?>
```

blog.css

```
body {
font-size: 70%;
color:#000000;
background-color:#FFD833;
margin:0px;
}
body, p, h1, h2, h3, table, td, th, ul, ol, textarea, input, a {
font-family: verdana,helvetica,arial,sans-serif;
}
td.menu, td.vsebina {
background-color:#ffffff;
padding:10px;
}
td.locnica {
background-color:#CC9933;
}
div#menu {
font-size:100%;
font-weight:normal;
color:#000000;
/*background-color:#ffffff;*/
}
div#glava {
font-size:120%;
font-variant: small-caps;
letter-spacing: 0.3cm;
font-weight:bold;
color:#660000;
margin-top:10px;
margin-bottom:5px;
text-align: center;
}
div#noga {
font-size:80%;
color:#333333;
background-color:#CC9933;
padding: 2px;
}
a:link    {color:#900B09; background-color:transparent}
a:visited {color:#900B09; background-color:transparent}
a:active  {color:#FF0000; background-color:transparent}
a:hover   {color:#FF0000; background-color:transparent; text-decoration: none;}

div#prispevek {
font-size:100%;
font-weight:normal;
color:#000000;
background-color:#FFFFCC;
margin:20px;
padding:10px;
border-style: solid;
border-color: #C0C0C0;
border-width: 1px
}
```

glava.php

```
<?php
//začnemo sejo na vsaki strani, da lahko dostopamo do spremenljivk v seji
```

```
session_start();
if(!session_is_registered("language_id")) {//če jezik ni izbran izberemo slo
   session_register("language");
   $_SESSION['language_id']="slo.inc.php";
} else if ($_GET['lang']=="slo.inc.php") {
   $_SESSION['language_id']="slo.inc.php";
} else if ($_GET['lang']=="en.inc.php") {
   $_SESSION['language_id']="en.inc.php";
}

include($_SESSION['language_id']);

?>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=UTF-8" http-equiv="content-type">
  <title><?php echo $GLOBAL['Nas_Dnevnik'] ?></title>
  <link href="blog.css" rel="stylesheet" type="text/css">
</head>
<body>

<?php
// ker se priklapljamo na bazo na več straneh to vključimo na vse strani
$server="localhost";
$uporabnik="rp_uporabnik";
$geslo="2007rp";
$baza="rp2007";
$db = mysql_connect($server,$uporabnik,$geslo);
mysql_select_db($baza,$db)or die( "Unable to select database");
?>
```

logo.php

```
  <table border="0" width="100%">
   <tr><td><div id="glava">
        <?php echo $GLOBAL['logo'] ?>
        </div></td></tr>
   <tr><td class="locnica">
       <?php // izbira jezika
           echo '<a href="'.$_SERVER['PHP_SELF'].'?lang=slo.inc.php">sLo</a> -
                 <a href="'.$_SERVER['PHP_SELF'].'?lang=en.inc.php">eN</a>';
       ?>
       </td></tr>
  </table>
```

meni.php

```
<div id="menu">

<a href="index.php"><?php echo $GLOBAL['Moj_blog'] ?></a><br>
<a href="vnos.php"><?php echo $GLOBAL['Vnos_prispevka'] ?></a><br><br>

<?php
//odjava iz sistema, ce smo prijavljeni

if ($_SESSION['usernames']!="") {
  echo '<a href="'.$_SERVER['PHP_SELF'].'?odjava=1">'.$GLOBAL['Odjava'].'</a><br>';
}
if ($_GET["odjava"]==1){
  session_unregister("usernames");
  session_unregister("emails");
  session_unregister("names");
  unset($_SESSION['usernames']); //zbrišemo tudi vrednost
  unset($_SESSION['emails']);
  unset($_SESSION['names']);
  unset($usernames); //za slučaj če imamo GLOBALS on
  unset($emails);
  unset($names);
  $_SESSION['prijavljen']=false;
```

```
  $_GET["odjava"]=0; //sicer ni potrebno, pa vendar
}

//izpis koledarjev po mescih
$month=date("m");
$year=date("Y");

for($j=$year;$j>($year-3);$j--){
  echo '<hr><p>'.$j.'</p><p align="center">';
  if ($j==$year) {
    for ($i=$month; $i>0 ; $i--) {
      //posiljanje spletni strani podatke preko URL - GET metoda
      echo '|-<a href="index.php?year='.$j.'&month='.$i.'">'.$i.'</a>-';
      if($i==9||$i==5) echo '|<br>';
    }
    echo '|';
  } else {
    for ($i=12; $i>0 ; $i--) {
      //posiljanje spletni strani podatke preko URL - GET metoda
      echo '|-<a href="index.php?year='.$j.'&month='.$i.'">'.$i.'</a>-';
      if($i==9||$i==5) echo '|<br>';
    }
    echo '|';
  }
  echo '</p><hr>';
}
?>

</div>
```

noga.php

```
  <table border="0" width="100%">
   <tr><td colspan="2"><div id="noga"><?php echo $GLOBAL['Noga'] ?></div></td></tr>
  </table>
</body>
</html>
```

index.php

```
<?php include('glava.php'); ?>
<?php include('logo.php'); ?>

  <table border="0" width="100%">
   <tr><td class="menu" width="150" valign="top"><?php include('meni.php'); ?></td>
    <td class="vsebina" valign="top">
         <?php

           $server="localhost";
           $uporabnik="rp_uporabnik";
           $geslo="2007rp";
           $baza="rp2007";
           $db = mysql_connect($server,$uporabnik,$geslo);
           mysql_select_db($baza,$db)or die( "Unable to select database");

           if ($_POST["submit"] && $_POST["id_blog"]) {
                //vnesi komentar, če kdo klikne na gumb
                $datum=date("Y-m-d"); //danasnji datum
                $sql="INSERT INTO komentar
                      VALUES ('','".$_POST["id_blog"]."',
                              '".$_POST["ime"]."',
                              '".$_POST["email"]."',
                              '$datum',
                              '".$_POST["komentar"]."')";
                $result = mysql_query($sql,$db);
           }

           //kaj izpisemo je odvisno od kod prihajamo (na kaj smo kliknili)
           if ($_POST["id_blog"]) { // ce je poslan id zapisa ga najdi
             $sql="SELECT * FROM blog WHERE id_blog=".$_POST["id_blog"];
           } elseif ($_GET["id_blog"]) {// ce je poslan id zapisa ga najdi
```

```
            $sql="SELECT * FROM blog WHERE id_blog=".$_GET["id_blog"];
          } elseif ($_GET["month"]&&$_GET["year"]) { // ce imamo mesec in leto
            //najdemo vse zapise tega leta in meseca
            $sql="SELECT * FROM blog WHERE datum LIKE '".$_GET["year"]."-
".$_GET["month"]."%'";
          } else { // ce ne pa najdi zadnjega zapisanega
           $sql="SELECT * FROM blog ORDER BY datum DESC LIMIT 1";
            $prazen=1;
          }
          $result = mysql_query($sql,$db);

          while ($myrow = mysql_fetch_row($result)) { //dokler je se kaj najdenega
             echo '<div id="noga">'.$myrow[5].'</div>';
            echo '<div id="prispevek">
                   <a href="mailto:'.$myrow[3].'">'.$myrow[1].' '.$myrow[2].'</a><br>';
            echo '<a href="index.php?id_blog='.$myrow[0].'">'.$myrow[4].'</a>
                   <font color="grey">('.$myrow[6].')</font><br>';
            echo '<br>'.$myrow[7].'
                     </div>';
             if ($_POST["id_blog"] || $_GET["id_blog"] || $prazen==1) {
               // izpis komentarjev
               $db2 = mysql_connect($server,$uporabnik,$geslo);
               mysql_select_db($baza,$db2)or die( "Unable to select database");
               $sql2="SELECT * FROM komentar WHERE id_blog=".$myrow[0];
               $result2 = mysql_query($sql2,$db2);
               while ($myrow2 = mysql_fetch_row($result2)) {
                   echo '<hr><p><a href="mailto:'.$myrow2[3].'">'.$myrow2[2].'</a>
                        ('.$myrow2[4].'): '.$myrow2[5].'</p>';
               }
               // obrazec za vpis komentarja
               echo '<hr>
                    <form method="post" action="index.php">
                    <input type="hidden" name="id_blog" value="'.$myrow[0].'">
                    '.$GLOBAL['Ime'].': <input name="ime"> <br>
                    '.$GLOBAL['Elektronski_naslov'].': <input name="email"> <br>
                    '.$GLOBAL['Komentar'].': <br>
                        <textarea cols="50" rows="4" name="komentar"></textarea><br>
                    <input type="submit" name="submit" value="'.$GLOBAL['Objavi'].'">';
            }
          }

          mysql_close();
        ?>
    </td>
   </tr>
  </table>

<?php include('noga.php'); ?>
```

vnos.php

```
<?php include('glava.php'); ?>
<?php include('logo.php'); ?>

  <table border="0" width="100%">
   <tr><td class="menu" width="150" valign="top">
       <?php include('meni.php'); ?></td>
    <td class="vsebina">

    <?php //obrazec izpisemo le ce je uporabnik prijavljen

    if ($_POST["submit"]) {
      $conn= ldap_connect("localhost",389);
      ldap_set_option( $conn, LDAP_OPT_PROTOCOL_VERSION, 3);
      ldap_set_option( $conn, LDAP_OPT_REFERRALS, 0);
      $ldap_pot="ou=PEFGosti,ou=People,dc=upr,dc=si";
      $res = ldap_bind($conn,"uid=".$_POST["username"].",".$ldap_pot, $_POST["pass"]);
      if ($res){
        $sr=ldap_search($conn, "dc=upr,dc=si","(uid=".$_POST["username"].")");
        $info = ldap_get_entries($conn, $sr);
        if (isset($info[0])){
```

```
      session_register("prijavljens");
      session_register("usernames");
      session_register("emails");
      session_register("names");
      $_SESSION['prijavljen']=true;
      $_SESSION['usernames']=$_POST["username"];
      $_SESSION['emails']=$info[0]["mail"][1];
      $_SESSION['names']=$info[0]["cn"][0];
    }
  }
}

if (!session_is_registered("usernames")) {
    echo '<hr>
    <form method="post" action="'.$_SERVER['PHP_SELF'].'">
    <table border="0">
      <tr><td>Uporabniško ime: </td><td><input name="username"></td></tr>
      <tr><td>Geslo: </td><td><input type="password" name="pass"></td></tr>
      <tr><td>  </td><td>
          <input type="submit" name="submit" value="Prijavi"></td></tr>
    </table>';
} else {
?>
<form method="post" action="obdelaj.php" name="prispevek">
<table width="100%" border="0" cellpadding="2" cellspacing="2">
  <tr>
    <th align="right"><?php echo $GLOBAL['Avtor_prispevka'] ?></td><td> </th>
  </tr>
  <tr>
    <td align="right"><?php echo $GLOBAL['Ime'] ?>: </td>
       <td><input name="ime" value="<?php echo $_SESSION['names'] ?>"></td>
  </tr>
  <tr>
    <td align="right"><?php echo $GLOBAL['Priimek'] ?>: </td>
       <td><input name="priimek" value="<?php echo $_SESSION['names'] ?>"></td>
  </tr>
  <tr>
    <td align="right"><?php echo $GLOBAL['Elektronski_naslov'] ?>: </td>
    <td><input name="email" value="<?php echo $_SESSION['emails'] ?>"></td>
  </tr>
  <tr>
    <td align="right"><?php echo $GLOBAL['Naslov_prispevka'] ?>: </td>
    <td><input name="naslov"></td>
  </tr>
  <tr>
    <td align="right"><?php echo $GLOBAL['Kategorija'] ?>: </td>
    <td>
      <select name="kategorija">
        <option value="sport"><?php echo $GLOBAL['Sport'] ?></option>
        <option value="politika"><?php echo $GLOBAL['Politika'] ?></option>
        <option value="zabava"><?php echo $GLOBAL['Zabava'] ?></option>
        <option value="studij"><?php echo $GLOBAL['Studij'] ?></option>
      </select>
    </td>
  </tr>
  <tr>
    <th valign="top" align="right"><?php echo $GLOBAL['Prispevek'] ?>: </th>
    <td><textarea cols="25" rows="10" name="prispevek"></textarea></td>
  </tr>
  <tr>
    <td align="right"> </td>
    <td>
    <input type="submit" name="submit" value="<?php echo $GLOBAL['Objavi'] ?>"> </td>
  </tr>
</table>
</form>
<?php
}
?>
</td>
</tr>
```

```
  </table>

<?php include('noga.php'); ?>
```

obsdelaj.php

```
<?php include('glava.php'); ?>
<?php include('logo.php'); ?>

  <table border="0" width="100%">
   <tr><td class="menu" width="150" valign="top"><?php include('meni.php'); ?></td>
    <td class="vsebina" valign="top">
    <?php

        if ($_POST["submit"]) { // ce smo kliknili na gumb - da kdo ne pride
                                // direktno na spletno stran brez
             //izpisovanje napak
           if (trim($_POST["ime"])=="" ||
               trim($_POST["priimek"])=="" ||
               trim($_POST["email"])=="" ||
               trim($_POST["naslov"])=="" ||
               trim($_POST["prispevek"])=="") { //ce je kaksno polje prazno
                  $error[]=$GLOBAL['Error1'];
           }
           if ((strlen($_POST["ime"])<2) || (strlen($_POST["priimek"])<2)) {
                  // ce sta ime ali priimek krajsa od dveh znakov
                  $error[]=$GLOBAL['Error2'];
           }
           if (!ereg("([A-Z]{1})[[:alpha:]]", $_POST["ime"])) {
                  $error[]=$GLOBAL['Error3'];
           }
           if (!ereg("([A-Z]{1})[[:alpha:]]", $_POST["priimek"])) {
                  $error[]=$GLOBAL['Error4'];
           }
           if (!ereg("^[-!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+"."@".
                        "[-!#$%&\'*+\\/0-9=?A-Z^_`a-z{|}~]+\.".
                        "[-!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+$", $_POST["email"])){
                 $error[]=$GLOBAL['Error5'];
           }

           if (count($error) == 0) { // ce ni napak izpisi podatke
              echo $GLOBAL['Vnesli_ste'].': <br>';
              echo '<div id="prispevek">'.$GLOBAL['Avtor_prispevka'].':
                     '.$_POST["ime"].' '.$_POST["priimek"].'<br>';
              echo $GLOBAL['Elektronski_naslov'].': '.$_POST["email"].'<br>';
              echo $GLOBAL['Naslov_prispevka'].': '.$_POST["naslov"].'<br>';
              echo $GLOBAL['Kategorija'].': '.$_POST["kategorija"].'<br>';
              echo $GLOBAL['Prispevek'].':<br>'.$_POST["prispevek"].'<div>';

                $datum=date("Y-m-d"); //danasnji datum
                $server="localhost";
                $uporabnik="rp_uporabnik";
                $geslo="2007rp";
                $baza="rp2007";
             $db = mysql_connect($server,$uporabnik,$geslo);
             mysql_select_db($baza,$db)or die( "Unable to select database");
                //$result = mysql_query("SET NAMES utf-8",$db);
                $sql="INSERT INTO blog
                      VALUES ('','".$_POST["ime"]."',
                              '".$_POST["priimek"]."',
                              '".$_POST["email"]."',
                              '".$_POST["naslov"]."',
                              '$datum',
                              '".$_POST["kategorija"]."',
                              '".$_POST["prispevek"]."')";
                echo $sql;
                $result = mysql_query($sql,$db);
             mysql_close();

         } elseif (is_array($error)) { // izpisi vse najdene napake
              echo '<p>';
```

```
                while (list($error_id,$error_text)=each($error)) {
                  echo $error_text.'<br>';
                  unset ($error[$error_id]);
                }
                echo '</p>';
             }
          } else {
            echo "Na spletno stran niste prišli preko obrazca za vnos objav v dneviški
sistem";
          }
      ?>
      </td>
     </tr>
    </table>

<?php include('noga.php'); ?>
```

mysql.sql

```
CREATE DATABASE `rp2007`;

GRANT USAGE ON * . * TO 'rp_uporabnik'@'%' IDENTIFIED BY '******';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER,
      CREATE TEMPORARY TABLES ON `rp2007`.* TO 'rp_uporabnik'@'%';

CREATE TABLE `blog` (
       `id_blog` INT NOT NULL AUTO_INCREMENT ,
       `ime_avtor` VARCHAR( 20 ) NOT NULL ,
       `priimek_avtor` VARCHAR( 20 ) NOT NULL ,
       `e-naslov` VARCHAR( 30 ) NOT NULL ,
       `naslov` VARCHAR( 50 ) NOT NULL ,
       `datum` DATE NOT NULL ,
       `kategorija` VARCHAR( 20 ) NOT NULL ,
       `prispevek` LONGTEXT NOT NULL ,
       PRIMARY KEY ( `id` )
) TYPE = MYISAM ;

CREATE TABLE `komentar` (
       `id_komentar` INT NOT NULL AUTO_INCREMENT ,
       `id_blog` INT NOT NULL ,
       `ime` VARCHAR( 50 ) NOT NULL ,
       `e-naslov` VARCHAR( 30 ) NOT NULL ,
       `datum` DATE NOT NULL ,
       `komentar` MEDIUMTEXT NOT NULL ,
       PRIMARY KEY ( `id_komentar` )
) TYPE = MYISAM ;
```

VAJA 9: projekt

Pri vajah smo izdelali preprost projekt. Je še daleč od končnega izdelka, ki bi moral biti brez napak in hroščev. Naš blog brez napak in hroščev ni. Vsaj nekaj izboljšav je potrebnih:

- brisanje in popravljanje vnosov
- iskalnik
- brskanje po vnosih enega uporabnika
- brskanje po kategoriji vnosov
- dodatek slik k vnosom
- lokalizacija glede na okolje uporabnika brez njegove izbire
- omogočanje prevajanje vsebine
- ...

Do konca vaj morate vi izdelati podoben projekt, v katerem boste izvedli vse korake in predvideli vse funkcionalne in sistemske zahteve vašega IS. Izdelati morate diagrame, izvesti med seboj (v skupini)ankete, da ugotovite, kaj vi kot uporabniki želite od sistema. Pričakovati morate, da bodo nove funkcionalne zahteve prihajale sproti, ko boste sredi programiranja. Vse zahteve zapisujte. Projekt mora biti dokumentiran.

Če imate svojo idejo za projekt se javite. V nasprotnem primeru bova ideje in skupine določila s profesorjem.