Andmebaas veebilehestiku juures

Nagu eelnenud näidetest näha, pole veebi koostamisel andmebaas sugugi hädavajalik lisandus. Küll aga võimaldab veebilehe andmebaasitabeliga ühendamine mitmete toimetustega mugavamal ja/või ohutumalt hakkama saada. Hulk võimalusi, mis muidu tuleks ise kodeerida, on juba andmebaasiprogrammidesse sisse ehitatud. Tüüpilised kohad, kus andmebaaside eelised välja tulevad on järgmised:

* samas veebilehestikus kasutatavad mitmesuguse struktuuriga seotud andmed
* otsing parameetrite järgi
* andmete muutmine veebi kaudu - eriti mitme administraatori korral

Eelnevalt nähtud suhteliselt staatilised lehed seega, kus kasutatakse lihtsalt ühiseid päiseid ning sarnase struktuuriga pikematekstilisi andmeid - selliste lehtede puhul pole andmebaas sugugi hädavajalik.

Kui aga soovida veebi kaudu muudetavat/täiendatavat lehestikku teha, siis on andmebaasi abi kindlasti omal kohal.

Enne näite juurde minemist märgime siia edaspidi kasutatavad mõisted.

* Andmebaasiprogramm, andmeohjur - masinasse installeeritud programm või draiver andmebaasidega ümber käimiseks. Nt. MySQL, Oracle, PostgreSQL, Access.

* Andmebaas - andmeohjuri poolt hallatav tabelite komplekt. Nt. konkreetse veebipoe jaoks tarvilikud andmetabelid. Ühes masinas võib olla (enamasti ongi) mitu andmebaasi. Igas andmebaasis saab olla palju andmetabeleid. Ühe andmebaasi piires on eri andmetabelitest andmeid mugavam/kiirem siduda kui eri andmebaaside vahel.

* Andmetabel - ridade ja veergudega andmete hoidmise koht. Igal veerul ehk tulbal on nimi ja tüüp - nt. tekst, kuupäev, täisarv, reaalarv.

* SQL - tõenäoliselt levinuim keel andmebaasile käskude andmiseks. Selle abil saavad andmebaasiga suhelda nii inimesed kui andmebaasivälised rakendusprogrammid (nt. PHP). Inimeste jaoks tehakse vahel ka graafilisi haldusvahendeid, kuid teised programmid suhtlevad üldjuhul andmebaasiga SQLi kaudu.

Siinses õppematerjalis kasutame andmebaasina MySQLi. Ta sobib PHPga hästi kokku, kuna on suunatud enamvähem sarnasele sihtgrupile: lihtsamate veebirakenduste lihtne loomine tasuta tarkvara abil. Eestis vähemasti 2009nda aastani tõenäoliselt levinuim moodus veebirakenduste loomiseks. (Üli)suurtes süsteemides nagu nt. pangalehed jääb vahenditesse sisseehitatud kontrolle ja stuktueeritust väheks. Aga enamike veebirakenduste nagu foorumid, uudistelehed, registreerumisvormid jaoks on võimalusi piisavalt. Arendamine ja ülespanek nõuab riistvaraliselt vähe ressursse, lehtede ülespanekuks kasutatavaid teenusepakkujaid (nt zone.ee, elkdata.ee) on piisavalt. Täiesti korraliku PHP ja MySQLi veebimajutuse leiab 2009nda aasta seisuga alates 100st kroonist kuus.

Ühe andmetabeliga seotud veebilehestik

MySQLis ja ka teistes relatsioonilistes ehk tabelipõhistes andmebaasides hoitakse ja väljastatakse pea kõiki andmeid tabelitena. Ning viimastel aastakümnetel üle ¾ andmebaasidest ongi relatsioonilised.

Andmebaasis toimetamiseks peab kõigepealt sellesse sisenema. Oma arvutisse nt. XAMPP abil veebiloomiskomplekti installides on MySQL kohe kättesaadav. Avalikumas serveris tuleb sisenemiseks enne administraatorilt kasutajakonto paluda. Ning mõnes teenusepakkuja keskkonnas saab ligi vaid graafilise kasutajaliidese (nt. PHP MyAdmin) kaudu.

Käsurealt sisenemine näeb välja ligukaudu järgmine. Võti -ujaagup näitab, et kasutajaks (user) on jaagup. Järgmine jaagup tähistab andmebaasi nime. Ning -p teatab, et parool küsitakse eraldi nõnda, et selle tähti ekraanile ei kuvata. Kui kõik õnneks läheb, siis ilmub lõpuks ette käsuviip mysql >

```
jaagup@tigu:~$ mysql -ujaagup jaagup -p
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 47574
Server version: 5.0.51a-24+lenny2-log (Debian)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Tabeli loomiseks käsklus CREATE TABLE. Käsu nimele järgneb tabeli nimi (praegusel juhul "lehed"). Ning siis sulgudes komadega eraldatuna tulpade nimed ning nende parameetrid. Iga tabeli esimeseks tulbaks on üldjuhul id - identifikaator, mille abil hiljem ridu eristada ja neile viidata. Parameetrid võivad lihtsamate rakenduste puhul enamasti samaks jääda. Selgitused:

INT - täisarv
NOT NULL - väärtus ei tohi puududa
AUTO_INCREMENT - server arvutab lisamisel ise juurde sobiva seni veel kasutamata väärtuse
PRIMARY KEY - selle tulba väärtust kasutatakse edaspidi tabeli vastavale reale viitamisel (näiteks muutmise või kustutamise juures).

Tulp pealkiri siin näites tüübiga VARCHAR(50) ehk siis tekst pikkusega kuni 50 tähte. Sisu tüübiks TEXT, mis tähendab, et pikkust ei piirata.

Kokku siis lause järgmine, mis tasub valmis kirjutada ning MySQLi käsuviibale kopeerida:

```
CREATE TABLE lehed(
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  pealkiri VARCHAR(50),
  sisu TEXT
);
```

Kui vastuseks tuli Query OK, siis järelikult ettevõtmine õnnestus. Muidu tuleb veateateid uurida ja uuesti proovida.

Andmete lisamiseks on loodud käsklus INSERT INTO. Järgneb tabeli nimi, siis sulgudes tulpade nimed kuhu lisatavad andmed tulevad. Edasi sõna VALUES ning sulgude sisse komadega eraldatult

igale tulbale vastav väärtus. Tekstilised andmed paigutatakse ülakomade vahele.

```
INSERT INTO lehed (pealkiri, sisu) VALUES ('Ilmateade', 'Kuiv ilm');
```

Tahtes rohkem andmeid lisada, tuleb INSERT lauset lihtsalt mitu korda käivitada, igal korral eraldi andmed sisse pannes.

```
mysql> INSERT INTO lehed (pealkiri, sisu) VALUES ('Korvpall', 'Treening reedel kell 18');
Query OK, 1 row affected (0.00 sec)
```

Kui mõned sees, siis on hea vaadata ja kontrollida, et mis sinna täpsemalt sai. Andmete küsimiseks on SQLis loodud käsklus SELECT. Tärn tähendab, et kuvatakse kõikide olemasolevate tulpade andmed. Sõnale FROM järgneb tabeli nimi ning käsu lõppu käivitamiseks semikoolon. Lehtede tabeli sisu tuleb siis välja järgnevalt.

```
mysql> SELECT * FROM lehed;
+----+-----------+-------------------------+
| id | pealkiri  | sisu                    |
+----+-----------+-------------------------+
|  1 | Ilmateade | Kuiv ilm                |
|  2 | Korvpall  | Treening reedel kell 18 |
+----+-----------+-------------------------+
2 rows in set (0.00 sec)
```

Nagu näha, id-väärtused on automaatselt ise pandud, kuna vastaval tulbal on juures omadus AUTO_INCREMENT.

Tahtes andmeid veel juurde panna, tuleb taas käivitada INSERT-lause sobivate andmetega. Teksti sisestamisele vastab kõige korrasoleku puhul MySQL taas "Query OK", lisades sinna vahel ka mõjutatud ridade arvu ja kulunud aja - tähtis pigem suuremate andmestike korral.

```
mysql> INSERT INTO lehed (pealkiri, sisu) VALUES ('Matemaatika', 'Homme tunnikontroll');
Query OK, 1 row affected (0.00 sec)
```

SQLi selecti abil saab andmeid kergesti sobivas järjekorras ja kujul välja küsida. Kui lause lõppu lisatakse ORDER BY koos vastava tulba nimega, siis tulevad andmed välja selle tulba järgi tähestiku järjekorda panduna (kui vastav tulp oli tekstitulp).

```
mysql> SELECT * FROM lehed ORDER BY sisu;
+----+-------------+-------------------------+
| id | pealkiri    | sisu                    |
+----+-------------+-------------------------+
|  3 | Matemaatika | Homme tunnikontroll     |
|  1 | Ilmateade   | Kuiv ilm                |
|  2 | Korvpall    | Treening reedel kell 18 |
+----+-------------+-------------------------+
3 rows in set (0.00 sec)
```

Saab küsida ka ainult ühe rea väärtusi:

```
mysql> SELECT pealkiri, sisu FROM lehed WHERE id=3;
+-------------+---------------------+
| pealkiri    | sisu                |
+-------------+---------------------+
| Matemaatika | Homme tunnikontroll |
+-------------+---------------------+
```

Kui leitakse, et rida pole enam vajalik, siis selle kustutamiseks sobib käsklus DELETE, kus soovitavalt id järgi määratakse ära, milline rida kustutada.

```
mysql> DELETE FROM lehed WHERE id=3;
Query OK, 1 row affected (0.00 sec)
```

Uue SELECT-päringuga saab kontrollida, mis siis sinna tegelikult alles jäi. Kui nüüd juhtutaks INSERT-lausega taas andmeid lisama, siis sellele reale enam id väärtust 3 välja ei antaks - välistamaks näiteks olukorda, kus vanale teatele pandud kommentaarid satuksid uue külge. Primaarvõtmetulba id väärtuseks tuleks uue rea lisamisel vähemasti 4.

```
mysql> SELECT * FROM lehed;
+----+-----------+-------------------------+
| id | pealkiri  | sisu                    |
+----+-----------+-------------------------+
|  1 | Ilmateade | Kuiv ilm                |
|  2 | Korvpall  | Treening reedel kell 18 |
+----+-----------+-------------------------+
```

MySQList saab välja käsuga `quit`

Ülesandeid

* Tee näited läbi
* Välju MySQList, sisene uuesti.
* Lisa teadete tabelisse veel mõned read.
* Väljasta teated sorteerituna pealkirja järgi

* Loo tabel "veised" tulpadega id, veisenimi, mass ja vanus
* Lisa mõned andmed
* Väljasta andmed sordituna vanuse järgi
* Väljasta ühe veise andmed id järgi.
* Kustuta see veis.
* Väljasta veised, kelle mass on 200-500kg (kahe tingimuse vahel AND).

Andmetabeli sisu kuvamine PHP abil.

MySQLi ja PHP ühendamiseks on aegade jooksul kasutatud mitmesuguseid vahendeid. Levinuimad on tõenäoliselt käsud mysql_connect, mysql_select_db ja mysql_query, millede abil on ka suurem osa kasutatavaid rakendusi kirjutatud. Selle komplekti puuduseks on aga sisendandmete suhteliselt vaba sattumine SQLi lausesse, mis teeb rakenduse kergemini haavatavaks. Aastaid on abiks kasutusel olnud eraldi loodud teek nimega PEAR (http://pear.php.net/). PHP 5. versiooniga aga tuli kaasa andmebaasipäringute poolest sarnaseid võimalusi pakkuv pakett MySQL Improved, mille abil ka siinse õppematerjali näited tehakse.

Järgnev PHP-leht kuvab andmetabelis olnud teated ekraanile. MySQL Improved Extensioni (http://ee.php.net/mysqli) võimaluste kasutamiseks tuleb kõigepealt vastav objekt luua.

```
$yhendus=new mysqli("localhost", "jaagup", "xxxxxx", "jaagup");
```

Edasi tehakse valmis käsklus.

```
$kask=$yhendus->prepare("SELECT id, pealkiri, sisu FROM lehed");
```

Siis määratakse, millistesse muutujatesse satuvad tulpadest tulevad andmed ning käivitatakse käsklus andmebaasiserveris. Muutujatesse pannakse andmed samas järjekorras, kui need tulevad välja SELECT-lausest.

```
  $kask->bind_result($id, $pealkiri, $sisu);
  $kask->execute();
```

Kui andmed käes, siis võib neid ühe rea kaupa ette võtma hakata. Iga `fetch()` täidab `bind_result`-käsklusega määratud muutujad uue rea andmetega. Tsükkel while jätkab senikaua kuni päringu vastuseks veel ridu on, ehk kuni `$kask->fetch` tagastab tõese väärtuse kursori edasiliikumise õnnestumise kohta.

Veebilehel näitamiseks pannakse pealkirjale ja sisule ümber käsklus htmlspecialchars - see asendab HTML-koodis tekstiosas lubamatud sümbolid (nt. < ja >) vastavate asenduskombinatsioonidega (`<` ja `>` nende märkide puhul).

```
       while($kask->fetch()){
         echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
         echo "<div>".htmlspecialchars($sisu)."</div>";
       }
```

Lõpuks on viisakas ühendus kinni panna. Ilma vastava käsuta küll ühendus suletakse ka mingi aja pärast, kuid jääb siiski mõneks ajaks rippuma ning tiheda kasutusega serveris võib tekkida olukord, kus vabade andmebaasiühenduste limiit saab otsa ja mõnda aega pole võimalik baasipäringuid kasutada. Kui aga ühendus selgesõnaliselt kinni panna, siis on vastav kanal vaba ning selle võib avada uue päringu tarbeks.

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

Tabeli andmeid veebilehel näitav kood tervikuna:

```
<?php
  $yhendus=new mysqli("localhost", "jaagup", "xxxxxx", "jaagup");
  $kask=$yhendus->prepare("SELECT id, pealkiri, sisu FROM lehed");
  $kask->bind_result($id, $pealkiri, $sisu);
  $kask->execute();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
     <title>Teated lehel</title>
  </head>
  <body>
     <h1>Teadete loetelu</h1>
     <?php
       while($kask->fetch()){
         echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
         echo "<div>".htmlspecialchars($sisu)."</div>";
       }
     ?>
  </body>
</html>
<?php
  $yhendus->close();
?>
```

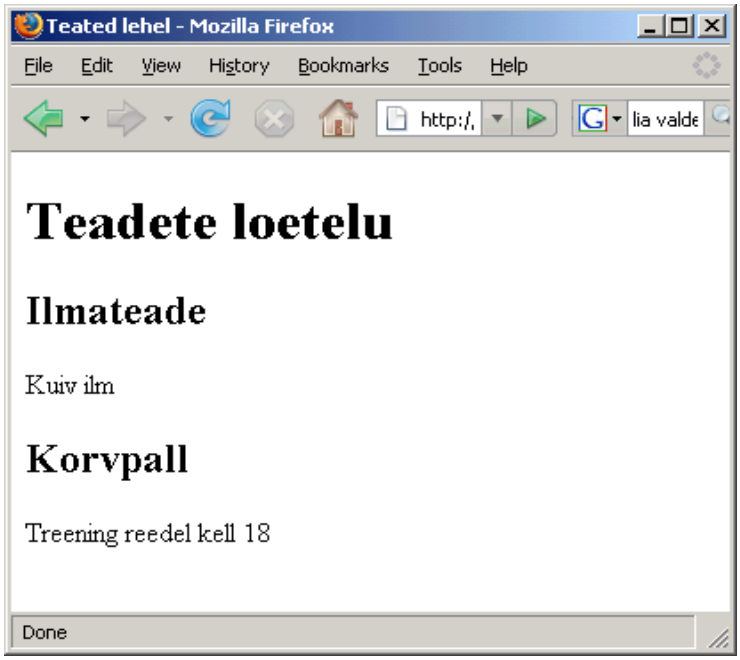
Käivitamise tulemusena valmis HTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
     <title>Teated lehel</title>
  </head>
  <body>
```

```
     <h1>Teadete loetelu</h1>
     <h2>Ilmateade</h2><div>Kuiv ilm</div>
<h2>Korvpall</h2><div>Treening reedel kell 18</div>  </body>

</html>
```

Ning selle põhjal järgmise väljanägemisega veebileht:



Ülesandeid

* Hangi omale võimalus MySQL-andmebaasile käsklusi saata - olgu siis PHPMyAdmini, käsurea või mõne muu koha kaudu.

* Loo/otsi veiste andmetabel (id, veisenimi, mass ja vanus), lisa mõned andmed
* Näita andmed veebilehele
* Vorminda andmed nummerdamata loeteluna (

```
<ul><li>Maasu, 400 kg, 5a</li><li>Tipa, 500 kg, 3a</li></ul>
```

)
* Vorminda andmed tabelina (

```
<table>
  <tr>
   <td>Veisenimi</td>
   <td>Mass</td>
   <td>Vanus</td>
  </tr>
  <tr>
   <td>Maasu</td>
   <td>400</td>
   <td>5</td>
  </tr>
  <tr>
   <td>Tipa</td>
   <td>500</td>
   <td>3</td>
  </tr>
```

```
  </table>
)
```

* veiste puhul, kelle mass ületab 500 kg, määra tabeli lahtri klassiks ”rammus”. Stiilikäsklusega näita lahtrid klassist ”rammus” rasvases kirjas (font-weight: bold).

* Koosta tabel koerte andmetega: id automaatselt suurenev primaarivõti, koeranimi kuni 30 sümboli pikkune tekst (VARCHAR) ning koera sünniaasta (INT). Lisa mõned andmed.

* Küsi tabelist koerad kord nime, kord sünniaastate järgi järjestatuna välja-
* Näita vaid koeri, kelle sünniaasta on suurem kui 2000.

* Koosta veebileht, kus on näha kõikide andmetabelis olevate koerte nimed ja sünniaastad.
* Kujunda leht kasutades võimalusel koerte pilte.

Teadete valik

Lühema sisuga teadete puhul sobib kõigi andmete korraga välja näitamine täiesti - vajalikud andmed on hea ülevaatlikult leida. Kui aga teated palju või nad pikemad, siis kipub kõige korraga nägemine tülikas olema. Esmaseks abiks sobib, kui algul paistavad välja vaid pealkirjad ning hiljem nendele vajutades saab ühe teate kaupa vaadata teate sisu teksti.

Selleks tuleb lehele andmete vaatamiseks luua kaks suhteliselt eraldi osa: menüü teadete loetelu jaoks ning keskel olev sisuosa ühe valitud teate näitamiseks.

Menüü kood on suhteliselt sarnane eelmise näite teadete loetelu omale. Lihtsalt loetelus näidatakse vaid teate pealkirja, sisu jäetakse näitamata. Pealkirjad muudetakse viideteks siiasamale failile nimega teadetevalik.php. Aadressiribale pannakse küsimärgi järel kaasa lähemalt vaadata soovitava teate id. Selle järgi on võimalik hiljem teate andmed küsida ning neid kasutada.

```
<ul>
  <?php
     $kask=$yhendus->prepare("SELECT id, pealkiri FROM lehed");
     $kask->bind_result($id, $pealkiri);
     $kask->execute();
     while($kask->fetch()){
       echo "<li>  <a href='teadetevalik.php?id=$id'>".
                  htmlspecialchars($pealkiri)."</a></li>";
     }
  ?>
</ul>
```

Teine plokk on `id` järgi konkreetsete andmete küsimine baasist ning näitamine veebilehele. See võetakse ette vaid juhul, kui aadressiribale on lisatud parameeter nimega "id". SELECT-lausele tuleb selle väärtus `bind_param`-käsu kaudu sisse ajada. Jutumärkide vahel olev "i" näitab, et parameetrina etteantavad id-d loetakse integeri ehk täisarvuna. Muude sisendite puhul jõuab päringulausesse selle koha peale lihtsalt arv 0.

Andmete kätte saamiseks määratakse bind_result-käskluse juures muutujate nimed, kuhu iga fetch-käsuga andmed sisestatakse. Endise `while($kask->fetch)` asemel on siin `if($kask->fetch)`, sest mitut kirjet sama id põhjal nagunii küsida ei saa. Kui aga ühtegi ei anta, siis tõenäoliselt on keegi

aadressiribale kirjutanud katsetamise huvides olematu id-numbri ning talle on viisakas teatada, et ta andmed on vigased.

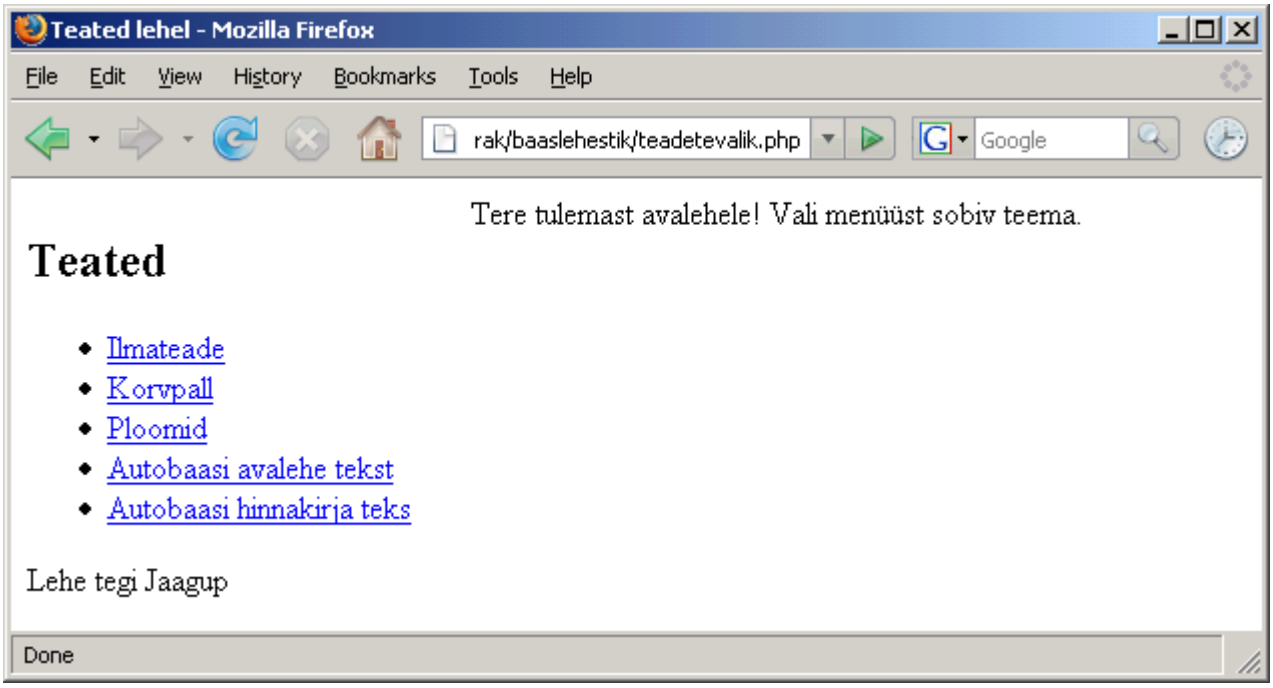
```
<?php
  if(isSet($_REQUEST["id"])){
     $kask=$yhendus->prepare("SELECT id, pealkiri, sisu FROM lehed
        WHERE id=?");
        //Kysim2rgi asemele pannakse aadressiribalt tulnud id,
        //eeldatakse, et ta on tyybist integer (i). (double - d, string - s)
     $kask->bind_param("i", $_REQUEST["id"]);
     $kask->bind_result($id, $pealkiri, $sisu);
     $kask->execute();
     if($kask->fetch()){
       echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
       echo htmlspecialchars($sisu);
     } else {
       echo "Vigased andmed.";
     }
  } else {
    echo "Tere tulemast avalehele! Vali men&uuml;&uuml;st sobiv teema.";
  }
?>
```

Kui `id`-väärtust aadressiribal pole üldse märgitud, siis tõenäoliselt on tegemist lehe esmaavamisega, kus pole veel teate id-d määratud. Viisakuse poolest pannakse avatud ühendus lehe lõpul ka kinni. Kujundust enam eraldi failidesse paigutada pole mõtet, sest kogu andmete näitamise töö teebki ära üks ja seesama fail.

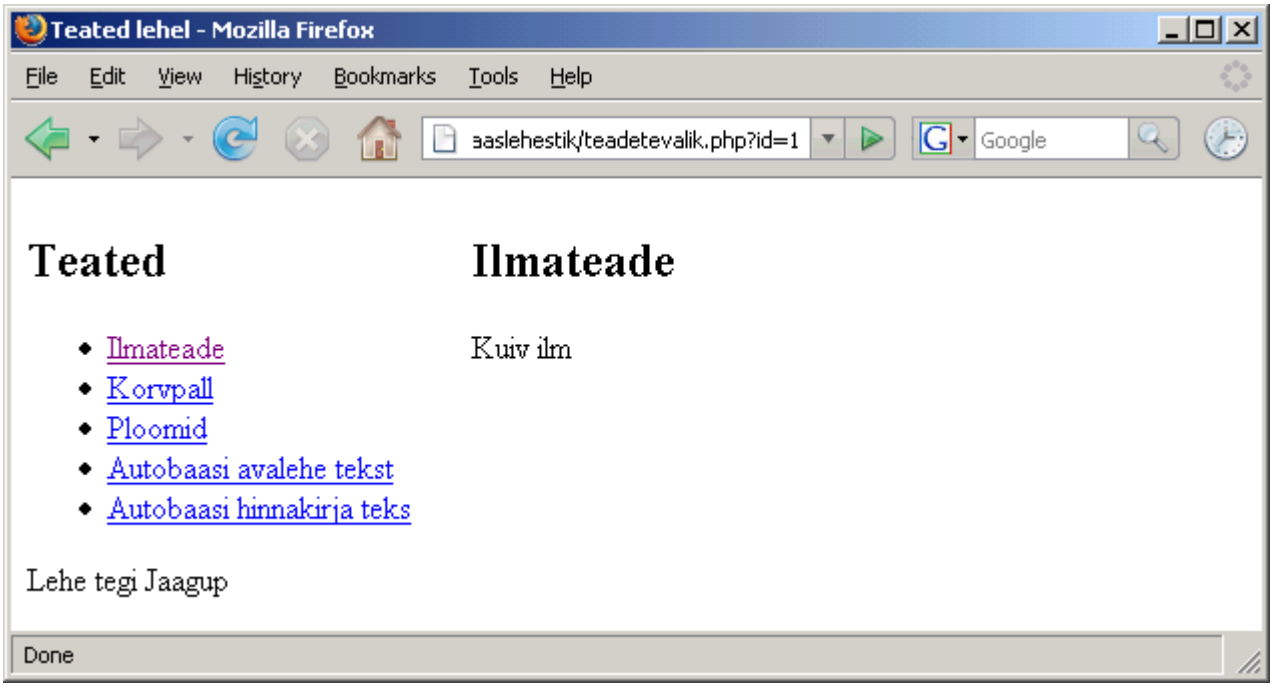
```
<?php
  $yhendus=new mysqli("localhost", "jaagup", "xxxxxx", "jaagup");
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
    <title>Teated lehel</title>
    <style type="text/css">
       #menyykiht{
         float: left;
         padding-right: 30px;
       }
       #sisukiht{
         float:left;
       }
       #jalusekiht{
         clear: left;
       }
    </style>
  </head>
  <body>
    <div id="menyykiht">
        <h2>Teated</h2>
        <ul>
          <?php
             $kask=$yhendus->prepare("SELECT id, pealkiri FROM lehed");
             $kask->bind_result($id, $pealkiri);
             $kask->execute();
             while($kask->fetch()){
               echo "<li><a href='teadetevalik.php?id=$id'>".
                     htmlspecialchars($pealkiri)."</a></li>";
             }
          ?>
        </ul>
    </div>
    <div id="sisukiht">
```

```
        <?php
          if(isSet($_REQUEST["id"])){
             $kask=$yhendus->prepare("SELECT id, pealkiri, sisu FROM lehed
                WHERE id=?");
                //Kysim2rgi asemele pannakse aadressiribalt tulnud id,
                //eeldatakse, et ta on tyybist integer (i). (double - d, string - s)
             $kask->bind_param("i", $_REQUEST["id"]);
             $kask->bind_result($id, $pealkiri, $sisu);
             $kask->execute();
             if($kask->fetch()){
               echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
               echo htmlspecialchars($sisu);
             } else {
               echo "Vigased andmed.";
             }
          } else {
            echo "Tere tulemast avalehele! Vali men&uuml;&uuml;st sobiv teema.";
          }
        ?>
      </div>
      <div id="jalusekiht">
         Lehe tegi Jaagup
      </div>
    </body>
</html>
<?php
  $yhendus->close();
?>
```

Avalehel näidatav teatepealkirjade loetelu



Valitud teate andmete kuvamine. Aadressiribal on näha valitud teate id.



Ülesandeid

* Mõelge/vaadake eelnenud näide läbi, vajadusel pange tööle.

* Koosta/otsi andmetabel koerte/veiste andmete hoidmiseks
* Loetelus on näha tabelisse sisestatud nimed
* Koera nimele vajutamisel näidatakse muid andmeid selle looma kohta.

Andmete lisamine ja kustutamine

Baasist tulevaid andmeid on hea ja ilus vaadata, ent kuidagi peavad need andmed baasi saama. Kui andmestik kuigivõrd ei muutu, siis võib ju ka andmebaasi enese haldusliidese abil nad sinna kirja panna. Kui aga tahta, et tavakasutaja saaks mugavasti sättida, mis tal lehe peal kirjas, siis tema hea meelega PHP MyAdmini või otse SQL käsuviiba taha ei lähe. Lihtne arusaadav veebileht andmete sisestamiseks ja muutmiseks sobib palju paremini.

Andmete lisamiseks tuleb kõigepealt toiminguga algust teha: kasutajale antakse pealkirjade menüü lõpus võimalus valida toiming "Lisa ...". Tehniliselt suunatakse kasutaja samale veebilehele, aga andes lehele kaasa parameetri `lisamine=jah`.

```
<a href='<?="$_SERVER[PHP_SELF]?lisamine=jah" ?>'>Lisa ...</a>
```

Selle peale kuvatakse lehel välja sisestusvorm. Vormi kohustuslik parameeter on action, mis näitab lehe aadressi, kuhu andmed jõuavad. Kui kogu haldus toimub sama lehe juures, siis on mugavaks mooduseks kirjutada action'iks `$_SERVER["PHP_SELF"]` - siis pannakse sinna viide lehele enesele. Ning oma juurde saab ligi ka juhul, kui lehe nime vahetatakse. Hilisema toimingu eristamise huvides lisan varjatud väljana (hidden) ka teate uusleht=jah.

```
if(isSet($_REQUEST["lisamine"])){
  ?>
    <form action='<?=$_SERVER["PHP_SELF"] ?>'>
     <input type="hidden" name="uusleht" value="jah" />
     <h2>Uue teate lisamine</h2>
     <dl>
       <dt>Pealkiri:</dt>
       <dd>
        <input type="text" name="pealkiri" />
       </dd>
       <dt>Teate sisu:</dt>
       <dd>
         <textarea rows="20" name="sisu"></textarea>
       </dd>
      </dl>
      <input type="submit" value="sisesta">
    </form>
  <?php
}
```

Selle järgi saab lehe algusosa koodis kontrollida, kas saadeti uued andmed, ehk kas muutuja `$_REQUEST["uusleht"]` on olemas. Tavajuhul seda pole ning plokist minnakse lihtsalt üle. Kui aga lehele saabuti vormist andmeid sisestades, siis varjatud väli andis teada, et sealtkaudu tuldi.

MySQL Improved-laienduse abil tuleb kõigepealt sisestamise SQL-lause kokku panna ning siis sinna andmed sisse paigutada. Pealkiri ja sisu on mõlemad tekstid ehk stringid - sellest ka `bind_param`-käsu parameetrite alguses "ss" ehk kaks stringi. Käsuga `execute` jõuavad andmed baasi.

Iseenesest on siiamaani toimetusega andmed sisestatud. Kui aga kood niimoodi jätta ja pärast `execute`-käsklust muude toimetustega edasi minna, sellisel juhul tekiks refresh-nuppu vajutades üllatus: iga vajutuse korral lisatakse veel kord tabelisse rea jagu samu andmeid. Seda seetõttu, et uuendusnupule vajutades jõuavad aadressireale (või ka post-meetodi kaudu) samad andmed ning nendega tehakse sama toiming ehk andmete lisamine tabelisse.

Soovimatust lisandreast vabanemiseks on üheks võimaluseks suunata lehe näitamine ümber. Päisekäsk `header("Location: failinimi.php")` suunab veebilehitseja edasi nimetatud failile. Kui failinimeks on `$_SERVER["PHP_SELF"]`, siis on tegelikult tegemist sama PHP failiga. Erinevuseks ainult, et kaasa ei anta tabelisse sisestamiseks mõeldud parameetreid. Seda meil aga just vaja ongi.

Pahatihti kiputakse selle päisekäskluse saatmisega piirduma, sest näiliselt oleks just nagu kõik korras. Samas on siiski viisakas andmebaasiühendus kinni panna. Ning mis vahel veel tähtsam - `exit`-käsklusega ülejäänud lehe näitamine katkestada. Tüüpiliseks probleemiks näiteks lehe turvamise juures on, et valede kasutajatunnustega sisenenud inimese puhul saadetakse küll brauserile käsklus lehe vahetamiseks, kuid tegeliku lehe serveerimist ei katkestata. Sellisel juhul on veebilehitseja asemel mõne telneti-sarnase programmiga veebiserveri pordiga suheldes võimalik vabalt ka ülejäänud lehe sisu näha. Nii et meelde jätmiseks - vaatamise ümber suunamisel kindlasti järgi käsklus `exit()`.

```
if(isSet($_REQUEST["uusleht"])){
  $kask=$yhendus->prepare("INSERT INTO lehed (pealkiri, sisu) VALUES (?, ?)");
  $kask->bind_param("ss", $_REQUEST["pealkiri"], $_REQUEST["sisu"]);
  $kask->execute();
  header("Location: $_SERVER[PHP_SELF]");
  $yhendus->close();
  exit();
```

```
  }
```

Andmete kustutamise juures tuleb kõigepealt kasutajale anda võimalus teate kustutamiseks. Sobib see näiteks teate vaatamise lehele, kus vastava teate andmed juba nagunii olemas on. Sealt loon viite samale lehele, andes aadressireal küsimärgi järele parameetri nimega "kustutusid", mille väärtuseks saab vastava teate id-number.

```
if($kask->fetch()){
  echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
  echo htmlspecialchars($sisu);
  echo "<br /><a href='$_SERVER[PHP_SELF]?kustutusid=$id'>kustuta</a>";
} else {
  echo "Vigased andmed.";
}
```

Lehe uuel avamisel saab kontrollida, kas esineb parameeter `kustutusid`. Kui mitte, siis pole vaja kustutamisega tegelda. Kui aga leidub, siis tuleb sobiv DELETE-lause käivitada. Primaarvõtmeks olev id-number siin täisarvuna, sellest ka "i" `bind_param` käskluse esimese parameetrina.

Kustutuse juures praegusel juhul korduva lehe avamise kontrolli ei ole, sest korduval sama id-ga teate kustutamisel ei juhtu midagi - vähemasti senikaua, kui pole eelnevat kontrolli tehtud, kas kustutatav asi üldse olemas on. Kui aga eeldan heauskset kasutajat, kes aadressireale käske ei kirjuta, vaid viisakasti viidete järgi rakendusega suhtleb, siis tal üksi oma lehestikku administreerides ei tohiks sellist üllatust juhtuda.

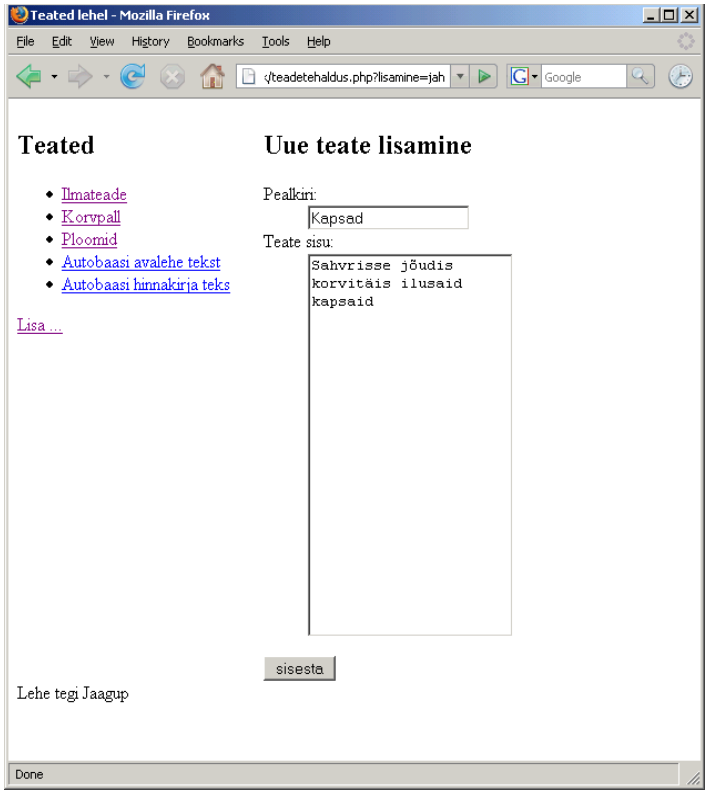
```
  if(isSet($_REQUEST["kustutusid"])){
    $kask=$yhendus->prepare("DELETE FROM lehed WHERE id=?");
    $kask->bind_param("i", $_REQUEST["kustutusid"]);
    $kask->execute();
  }
```

Edasi juba lisamis- ja kustutusvõimelise lehe kood tervikuna.

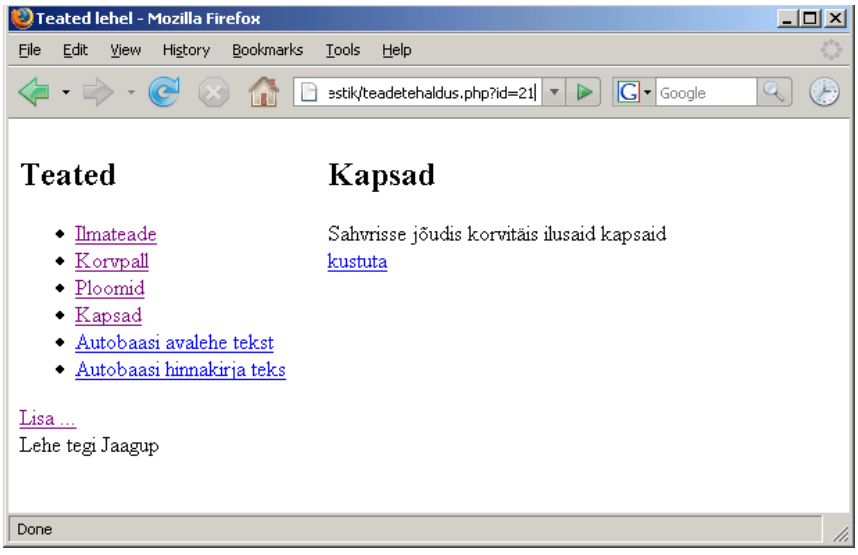
```
<?php
  $yhendus=new mysqli("localhost", "jaagup", "xxxxxx", "jaagup");
  if(isSet($_REQUEST["uusleht"])){
    $kask=$yhendus->prepare("INSERT INTO lehed (pealkiri, sisu) VALUES (?, ?)");
    $kask->bind_param("ss", $_REQUEST["pealkiri"], $_REQUEST["sisu"]);
    $kask->execute();
    header("Location: $_SERVER[PHP_SELF]");
    $yhendus->close();
    exit();
  }
  if(isSet($_REQUEST["kustutusid"])){
    $kask=$yhendus->prepare("DELETE FROM lehed WHERE id=?");
    $kask->bind_param("i", $_REQUEST["kustutusid"]);
    $kask->execute();
  }
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
    <title>Teated lehel</title>
    <style type="text/css">
       #menyykiht{
         float: left;
         padding-right: 30px;
       }
       #sisukiht{
         float:left;
       }
       #jalusekiht{
         clear: left;
```

```
      }
    </style>
  </head>
  <body>
    <div id="menyykiht">
        <h2>Teated</h2>
        <ul>
          <?php
             $kask=$yhendus->prepare("SELECT id, pealkiri FROM lehed");
             $kask->bind_result($id, $pealkiri);
             $kask->execute();
             while($kask->fetch()){
               echo "<li><a href='$_SERVER[PHP_SELF]?id=$id'>".
                           htmlspecialchars($pealkiri)."</a></li>";
             }
          ?>
        </ul>
        <a href='<?="$_SERVER[PHP_SELF]?lisamine=jah" ?>'>Lisa ...</a>
    </div>
    <div id="sisukiht">
       <?php
         if(isSet($_REQUEST["id"])){
            $kask=$yhendus->prepare("SELECT id, pealkiri, sisu FROM lehed
               WHERE id=?");
            $kask->bind_param("i", $_REQUEST["id"]);
            $kask->bind_result($id, $pealkiri, $sisu);
            $kask->execute();
            if($kask->fetch()){
              echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
              echo htmlspecialchars($sisu);
              echo "<br /><a href='$_SERVER[PHP_SELF]?kustutusid=$id'>kustuta</a>";
            } else {
              echo "Vigased andmed.";
            }
         }
         if(isSet($_REQUEST["lisamine"])){
           ?>
             <form action='<?=$_SERVER["PHP_SELF"] ?>'>
              <input type="hidden" name="uusleht" value="jah" />
              <h2>Uue teate lisamine</h2>
              <dl>
                <dt>Pealkiri:</dt>
                <dd>
                 <input type="text" name="pealkiri" />
                </dd>
                <dt>Teate sisu:</dt>
                <dd>
                  <textarea rows="20" name="sisu"></textarea>
                </dd>
               </dl>
               <input type="submit" value="sisesta">
             </form>
           <?php
         }
       ?>
    </div>
    <div id="jalusekiht">
       Lehe tegi Jaagup
    </div>
  </body>
</html>
<?php
  $yhendus->close();
?>
```

Andmete lisamine veebilehel:



Lisatud teate andmete vaatamine:



Ülesandeid

* Tee näide läbi

* Koosta/otsi andmetabelipõhine veebileht koerte/veiste andmete vaatamiseks
* Võimalda loomi lisada
* Võimalda loomi kustutada

Matka leht

* Loo leht kasutajate lisamiseks matkale: eesnimi, perekonnanimi, telefon, elektronpost

* Koosta leht matka tarbeks: marsruut, ajakava. Kujunda koos kasutajate lehega ühtseks lehestikuks.

* Loo eraldi administraatorileht vigaste sisestuste kustutamiseks
* Muuda ajakavalehte nõnda, et administraator saaks sinna sündmusi lisada ja sealt eemaldada.

Võimalikke lisaülesandeid matkalehele:

* Sisestuskontroll andmete korrektsuse kohta
* Mitu kujundusfaili, mille vahetamisega saab üldkujundust (paigutus, pildid) märgatavalt muuta
* Osalejate arvu teatamine avalehel
* Teemade kaupa kataloogis olevate piltide näitamine galeriis samuti teemade kaupa

Andmete muutmine veebilehel

Lihtsaim andmete muutmine sageli koosnebki kustutamisest ja uuest lisamisest. Nii on mõnigi kord kergem teha, kui viisakamat muutmisvahendit luua. Pikkade ja keerukate andmete puhul pole aga kasutajad kuigivõrd rahul, kui väikese trükivea või täienduse pärast peaks terve rea jagu andmeid uuesti sisse panema. Samuti on kustutamiseta muutmine tähtis oludes, kus andmeid hakatakse juba omavahel siduma. Kui näiteks koeral on kindel omanik, ehk viide inimeste andmetabeli reale. Siis selle rea juures nt. telefoninumbri muutmisel jääb koera juurde ikka sama omanik. Kui aga telefoninumbri muutmiseks inimese rida ära kustutatakse ja uuesti luuakse, siis on juba keerukam hoolitseda, et koera juures andmebaasis kindel sama omanik oleks.

Järgneva näite pealt näebki, kuidas PHP abil andmeid muuta nõnda, et ei peaks kõike maha kustutama ja uuesti sisestama, vaid saab paranduse paigutada vaid sobivasse kohta.

Kõigepealt lisame kustutamise viite kõrvale viite ka muutmise kohta. Anname lehe uuel avamisel kaasa muudetava teate id, samuti parameetri "muutmine" väärtusega "jah".

```
echo "<br /><a href='$_SERVER[PHP_SELF]?kustutusid=$id'>kustuta</a> ";
echo "<a href='$_SERVER[PHP_SELF]?id=$id&amp;muutmine=jah'>muuda</a>";
```

Nende põhjal saab siis järgmisel avamisel kontrollida, kas on seatud parameeter nimega "muutmine".

```
if(isSet($_REQUEST["muutmine"])){
```

Kui jah, siis luuakse vorm ja pannakse sinna kaasa muutmisid - mille järgi siis muudatuste salvestamise juures teab, milllise teate juures muutus tuleb kirja panna.

```
<input type='hidden' name='muutmisid' value='$id'/>
```

Väljadele antakse sisse olemasolevad väärtused, et neid saaks siis soovi järgi täiendada/parandada.

```
<input type='text' name='pealkiri'
value='".htmlspecialchars($pealkiri)."'/>
```

Muutmisvormi loov koodilõik siit tervikuna silma ette. Tekstiväljale ja tekstialale saab vana sisu suhteliselt lihtsalt sisse panna. Rippenüüde puhul tuleb näiteks sobiv rida kavalasti ette kerida - aga sellest juba edaspidi. Ning eks peab jääma ka lõik tavalise teate näitamiseks - ilma muutmata. Ning viisakuse poolest ka kontroll selle kohta, et kui soovitakse olematut teadet - olgu siis aadressirea muutmise tõttu või selle poolest, et keegi vahepeal serverist teate ära kustutas - siis antakse ka sõnadega teada, et nende andmete põhjal teadet kätte ei saa.

```
            if($kask->fetch()){
             if(isSet($_REQUEST["muutmine"])){
                echo "
                   <form action='$_SERVER[PHP_SELF]'>
                     <input type='hidden' name='muutmisid' value='$id'/>
                     <h2>Teate muutmine</h2>
                     <dl>
                       <dt>Pealkiri:</dt>
                       <dd>
                         <input type='text' name='pealkiri'
                                value='".htmlspecialchars($pealkiri)."'/>
                       </dd>
                       <dt>Teate sisu:</dt>
                       <dd>
                         <textarea rows='20' cols='30'
                            name='sisu'>".htmlspecialchars($sisu)."</textarea>
                       </dd>
                     </dl>
                     <input type='submit' value='Muuda' />
                   </form>
                ";
             } else {
              echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
              echo htmlspecialchars($sisu);
              echo "<br /><a href='$_SERVER[PHP_SELF]?kustutusid=$id'>kustuta</a> ";
              echo "<a href='$_SERVER[PHP_SELF]?id=$id&amp;muutmine=jah'>muuda</a>";
             }
            } else {
              echo "Vigased andmed.";
            }
```

Kui uued andmed sees, vajutatakse submit-nuppu ning andmed jõuavad taas lehele. Parameetri "muutmisid" olemasolu järgi teab, et nüüd tasub olemasoleva teate andmeid muutma hakata. Eks jällegi tuleb kokku panna SQL-lause. Kõik muudetavad väärtused saavad SET-osas omale veebilehelt tulnud andmete põhjal uue sisu ning lõpus WHERE osas määratakse, millise teate kohta see muutus käib. Siit kusjuures oht: kui kogemata unustatakse WHERE osa märkimata, siis kirjutatakse nende andmetega üle kõik tabelis leiduvad read ilma midagi eelnevalt küsimata ega kontrollimata.

Parameetrite tüüpideks siin "ssi", sest pealkiri ja sisu on stringid, tabelirida määrav id aga integer. Käsk taas käima ning andmed muudetud. Siingi ei ole lehe ümbersuunamist tehtud, sest kui ka juhtutaks lehe värskendusnupule vajutama, siis kirjutataks uued andmed lihtsalt veel korra tabelisse ning midagi muud sellega ei kaasne.

```
  if(isSet($_REQUEST["muutmisid"])){
    $kask=$yhendus->prepare("UPDATE lehed SET pealkiri=?, sisu=? WHERE id=?");
    $kask->bind_param("ssi", $_REQUEST["pealkiri"], $_REQUEST["sisu"],
$_REQUEST["muutmisid"]);
    $kask->execute();
  }
```

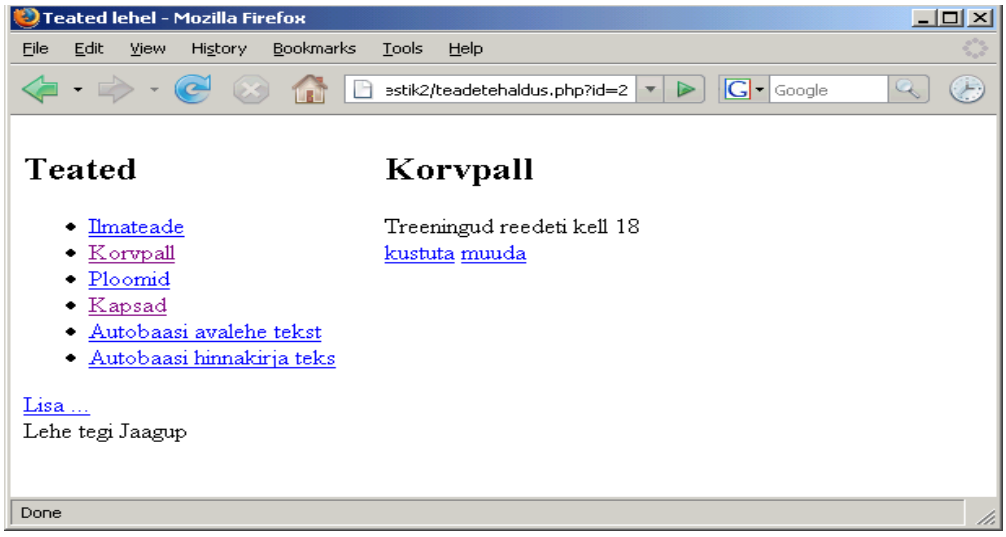
Edasi muutmisvõimelise lehe lähtekood tervikuna.

```
<?php
  $yhendus=new mysqli("localhost", "jaagup", "xxxxxx", "jaagup");
  if(isSet($_REQUEST["uusleht"])){
    $kask=$yhendus->prepare("INSERT INTO lehed (pealkiri, sisu) VALUES (?, ?)");
    $kask->bind_param("ss", $_REQUEST["pealkiri"], $_REQUEST["sisu"]);
    $kask->execute();
    header("Location: $_SERVER[PHP_SELF]");
    $yhendus->close();
    exit();
  }
  if(isSet($_REQUEST["kustutusid"])){
    $kask=$yhendus->prepare("DELETE FROM lehed WHERE id=?");
    $kask->bind_param("i", $_REQUEST["kustutusid"]);
```

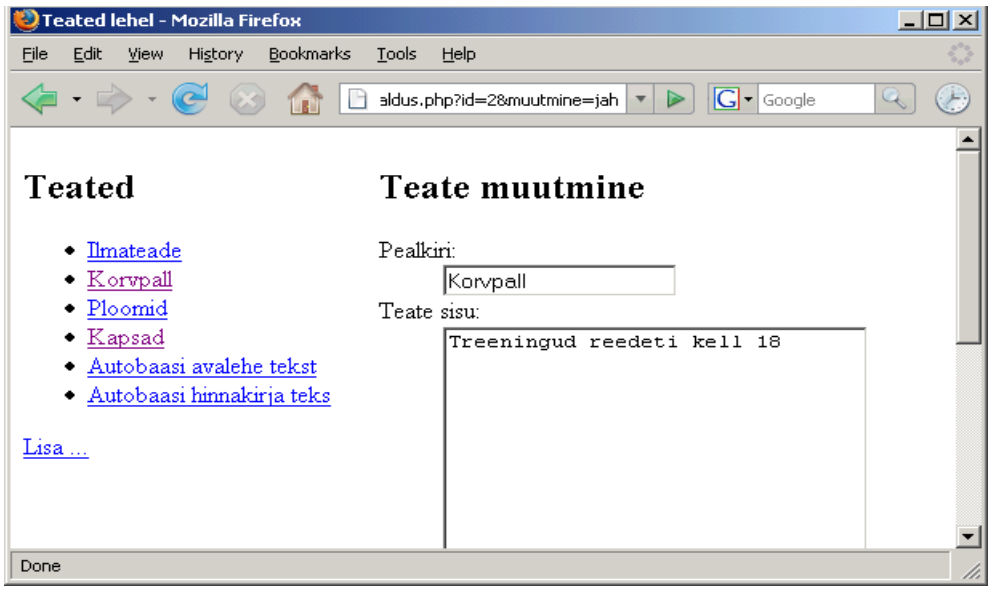
```
    $kask->execute();
  }
  if(isSet($_REQUEST["muutmisid"])){
    $kask=$yhendus->prepare("UPDATE lehed SET pealkiri=?, sisu=? WHERE id=?");
    $kask->bind_param("ssi", $_REQUEST["pealkiri"], $_REQUEST["sisu"],
$_REQUEST["muutmisid"]);
    $kask->execute();
  }
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Teated lehel</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
    <style type="text/css">
       #menyykiht{
         float: left;
         padding-right: 30px;
       }
       #sisukiht{
         float:left;
       }
       #jalusekiht{
         clear: left;
       }
    </style>
  </head>
  <body>
    <div id="menyykiht">
        <h2>Teated</h2>
        <ul>
          <?php
             $kask=$yhendus->prepare("SELECT id, pealkiri FROM lehed");
             $kask->bind_result($id, $pealkiri);
             $kask->execute();
             while($kask->fetch()){
               echo "<li><a
href='$_SERVER[PHP_SELF]?id=$id'>".htmlspecialchars($pealkiri)."</a></li>";
             }
          ?>
        </ul>
        <a href='<?="$_SERVER[PHP_SELF]?lisamine=jah" ?>'>Lisa ...</a>
    </div>
    <div id="sisukiht">
       <?php
         if(isSet($_REQUEST["id"])){
            $kask=$yhendus->prepare("SELECT id, pealkiri, sisu FROM lehed WHERE id=?");
            $kask->bind_param("i", $_REQUEST["id"]);
            $kask->bind_result($id, $pealkiri, $sisu);
            $kask->execute();
            if($kask->fetch()){
             if(isSet($_REQUEST["muutmine"])){
                echo "
                   <form action='$_SERVER[PHP_SELF]'>
                     <input type='hidden' name='muutmisid' value='$id'/>
                     <h2>Teate muutmine</h2>
                     <dl>
                       <dt>Pealkiri:</dt>
                       <dd>
                         <input type='text' name='pealkiri' value='".
                                   htmlspecialchars($pealkiri)."'/>
                       </dd>
                       <dt>Teate sisu:</dt>
                       <dd>
                         <textarea rows='20' cols='30' name='sisu'>".
                            htmlspecialchars($sisu)."</textarea>
                       </dd>
                     </dl>
                     <input type='submit' value='Muuda' />
                   </form>
```

```
                ";
              } else {
               echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
               echo htmlspecialchars($sisu);
               echo "<br /><a href='$_SERVER[PHP_SELF]?kustutusid=$id'>kustuta</a> ";
               echo "<a href='$_SERVER[PHP_SELF]?id=$id&amp;muutmine=jah'>muuda</a>";
              }
             } else {
               echo "Vigased andmed.";
             }
          }
          if(isSet($_REQUEST["lisamine"])){
            ?>
              <form action='<?=$_SERVER["PHP_SELF"] ?>'>
               <input type="hidden" name="uusleht" value="jah" />
               <h2>Uue teate lisamine</h2>
               <dl>
                 <dt>Pealkiri:</dt>
                 <dd>
                  <input type="text" name="pealkiri" />
                 </dd>
                 <dt>Teate sisu:</dt>
                 <dd>
                   <textarea rows="20" cols="30" name="sisu"></textarea>
                 </dd>
                </dl>
                <input type="submit" value="sisesta" />
              </form>
            <?php
          }
        ?>
    </div>
    <div id="jalusekiht">
       Lehe tegi Jaagup
    </div>
  </body>
</html>
<?php
  $yhendus->close();
?>
```

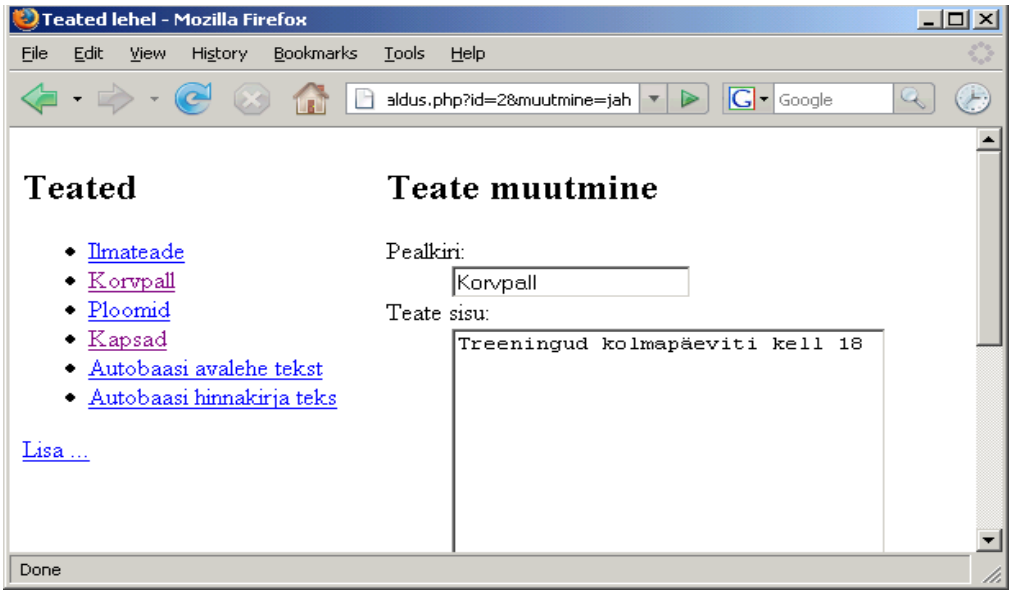
Piltidelt on näha, kuidas kõigepealt avatakse id järgi sobiv teade.



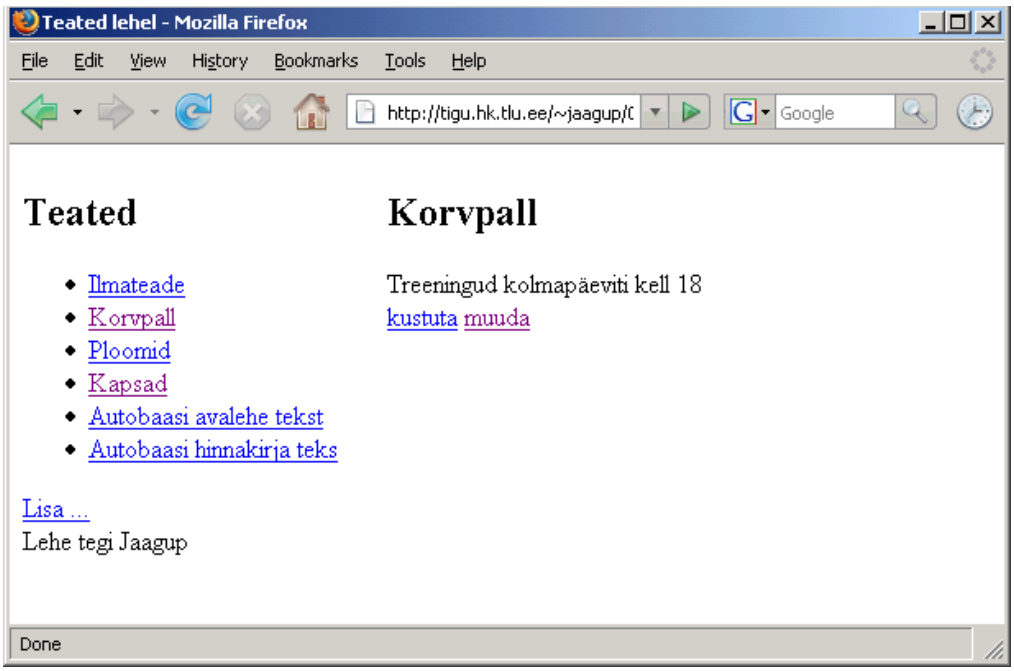
Edasi muutmisviitele vajutades avatakse leht koos vormiga andmete muutmiseks:



Siis saab parandused sisse viia



Ning lõpuks tasub muudetud lehte imetleda.



Ülesandeid

* Tee näide läbi.
* Anna muutmisel hoiatus juhul, kui sisestatud pealkirja pikkus ületab 50 sümbolit

* Loo/otsi koerte/veiste andmeid näitav veebirakendus
* Lisa rakendusele andmete muutmise võimalus.

Väikefirma veebilehestik

* Mõtle välja / lepi kaaslasega kokku firma, kellele veebileht teha (näiteks pagariäri, autoremonditöökoda)
* Koosta kujundatud tutvustav leht.
* Loo andmebaasitabel toodete hinnakirja tarbeks. Tutvustuslehelt pääseb hinnakirja vaatama.
* Loo eraldi ligipääsuga andministraatorileht toodete hinnakirja haldamiseks (lisamine, muutmine, kustutamine).
* Lisa veebilehestikule toodete/tööde pildigalerii, kuhu ilmuvad kõik kindlasse kataloogi kopeeritud pildid.

Graafiline tekstiredaktor

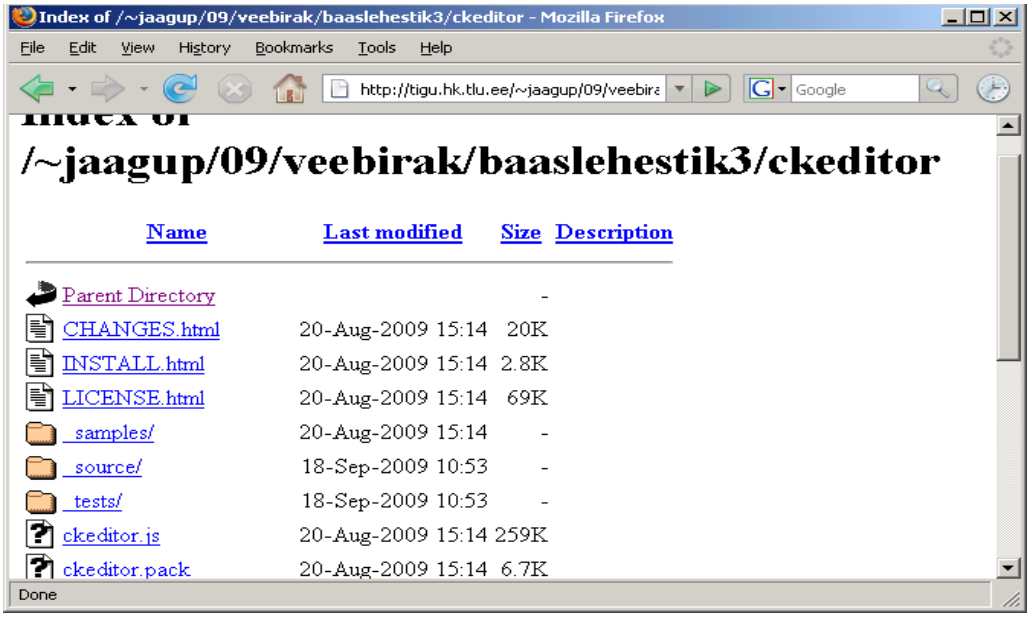
Aastaid oli veebikoostajate märgatavaks osaks tööst lehekülgede kaupa HTMLi kirjutada ja hoolitseda, et selle abil kirja pandud tekst rahvale viisakalt loetav oleks. Eks sellist kujundamist ole praegugi - kui tahtmist oma leht pikslipealt paika joonistada. Aga veebipõhised redaktorid aitavad küllaltki mugavalt tavakasutajatel omaloodud sisu kujundada ilma, et arvutiabilist pidevalt kõrval oleks.

Graafilisi ehk WYSIWYG (What You See Is What You Get) tekstiredaktoreid veebilehel on tekkinud hulgem, tuleb vaid omale sobiv leida. Väiksematel piisab mõnekilobaidisest Javaskripti failist, keerukamatel võib allalaetav skript koos abipiltidega mitmesaja kilobaidini ulatuda. Samuti

suudab mõni hakkama saada vaid üksikute brauserite ja versioonidega, teisel on paindlikkust rohkem.

Kirjutise autor on juhtunud kasutama Javaskriptipõhiseid redaktoreid TinyMCE ja Kupu. Ning subjektiivselt parima mulje on jätnud CKEditor (endine FCKEditor). Seda kasutame ka siinses näites.

Redaktor tuleb kõigepealt alla laadida ja lahti pakkida. Tasub vaadata, et kus kaustas asub fail ckeditor.js - see tuleb varsti meie lehe külge haakida.



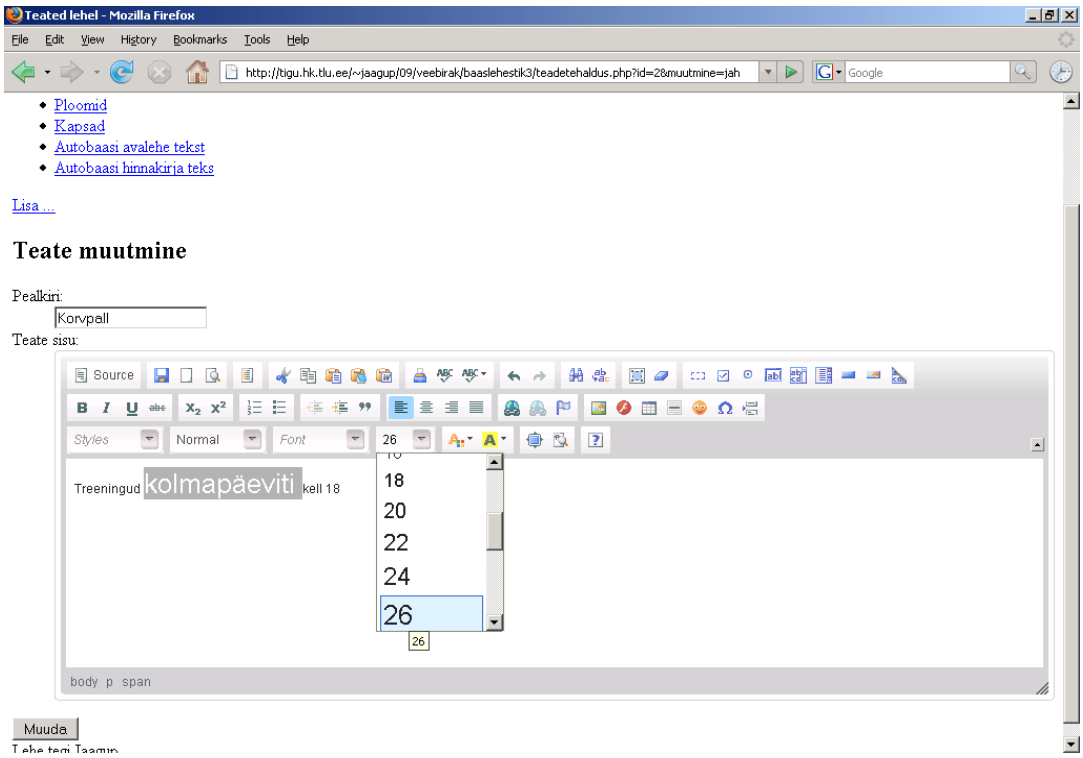
Siin puhul sattusid redaktori failid eraldi alamkausta ckeditor - seega laeme sisse faili aadressilt ckeditor/ckeditor.js.

```
<script type="text/javascript" src="ckeditor/ckeditor.js"></script>
```

Uus versioon on nõnda targaks tehtud, et teksti ala graafiliseks redaktoriks muutmiseks on tarvis vaid talle külge panna atribuut class='ckeditor'

```
<textarea rows='20' cols='30' class='ckeditor' name='sisu'>$sisu</textarea>
```

Ning kui kõik andmed ilusti kätte saadakse, siis võibki rahulikult menüüst kujunduskäske valida ja teksti suurust muuta.



Sama asi ka lisamise juures: tuleb lihtsalt vaadata, kus asub tekstiala, talle külge panna klass ckeditor ning jällegi on elu tunduvalt värvilisem. Kui millegipärast ei ole, siis tasub kõigepealt uurida, et kas viidatud kohast ikkagi javaskripti fail kergesti kätte saadakse. Selle aadressi võib kasvõi brauserireale sisse lüüa ja kontrollida, et mis sealt välja antakse. Samuti tasub kontrollida, et ega veebilehitsejal pole Javaskript kinni keeratud - turvapõhjustel seda mõnikord tehakse.

```
if(isSet($_REQUEST["lisamine"])){
  ?>
    <form action='<?=$_SERVER["PHP_SELF"] ?>'>
     <input type="hidden" name="uusleht" value="jah" />
     <h2>Uue teate lisamine</h2>
     <dl>
       <dt>Pealkiri:</dt>
       <dd>
        <input type="text" name="pealkiri" />
       </dd>
       <dt>Teate sisu:</dt>
       <dd>
         <textarea rows="20" cols="30" class="ckeditor" name="sisu"></textarea>
       </dd>
      </dl>
      <input type="submit" value="sisesta" />
    </form>
  <?php
}
```

Kui niisama lihtsalt panna lehe tekstialadele redaktor külge, siis on küll ilus teksti värviliseks joonistada, kuid suure tõenäosusega pärast vaadates pole muutused korralikult näha. Raskemal juhul võib osa teksti sootuks kadunud olla.

Põhjuseks, et redaktorist tulnud HTMLi varjestab PHP vastavalt serveri konfiguratsioonile ära. MySQL Improved aga ei vaja SQL-lausesse paigutamiseks andmete langjoontega varjestamist ning nõnda jõuavad tavajuhul mitmele poole \-märgid, mis häirivad HTMLi vaatamist.

Liigsetest langjoontest aitab vabaneda funktsioon `stripslashes`, mis siis varjestatud sisu uuesti "tavaliseks" teeb.

```
  if(isSet($_REQUEST["muutmisid"])){
    $kask=$yhendus->prepare("UPDATE lehed SET pealkiri=?, sisu=? WHERE id=?");
    $kask->bind_param("ssi", $_REQUEST["pealkiri"], stripslashes($_REQUEST["sisu"]),
$_REQUEST["muutmisid"]);
    $kask->execute();
  }
```

Lehtede tabeli loomiseks oli lause

```
CREATE TABLE lehed(
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  pealkiri VARCHAR(50),
  sisu TEXT
);
```

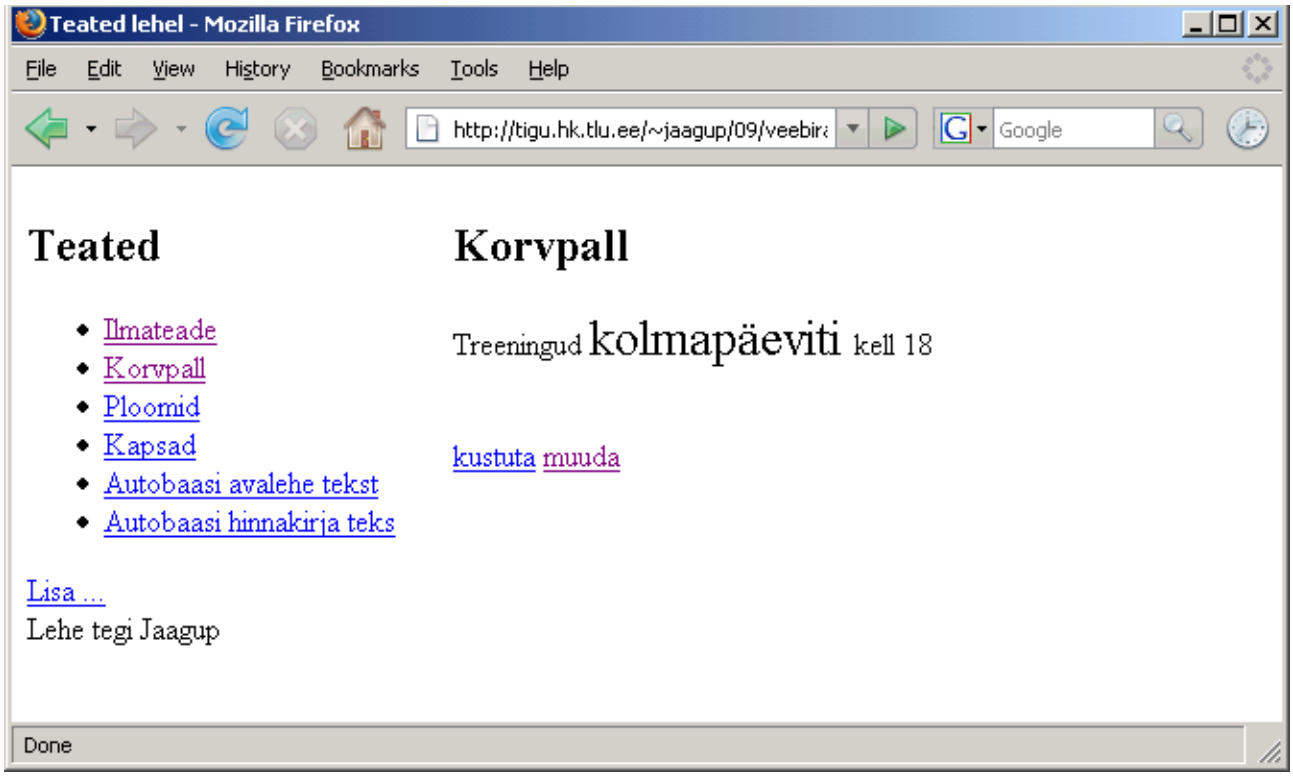
Redaktorit kasutava lehe kood tervikuna.

```
<?php
  $yhendus=new mysqli("localhost", "jaagup", "xxxxxx", "jaagup");
  if(isSet($_REQUEST["uusleht"])){
    $kask=$yhendus->prepare("INSERT INTO lehed (pealkiri, sisu) VALUES (?, ?)");
    $kask->bind_param("ss", $_REQUEST["pealkiri"], stripslashes($_REQUEST["sisu"]));
    $kask->execute();
    header("Location: $_SERVER[PHP_SELF]");
    $yhendus->close();
    exit();
  }
  if(isSet($_REQUEST["kustutusid"])){
    $kask=$yhendus->prepare("DELETE FROM lehed WHERE id=?");
    $kask->bind_param("i", $_REQUEST["kustutusid"]);
    $kask->execute();
  }
  if(isSet($_REQUEST["muutmisid"])){
    $kask=$yhendus->prepare("UPDATE lehed SET pealkiri=?, sisu=? WHERE id=?");
    $kask->bind_param("ssi", $_REQUEST["pealkiri"], stripslashes($_REQUEST["sisu"]),
$_REQUEST["muutmisid"]);
    $kask->execute();
  }
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Teated lehel</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
    <style type="text/css">
       #menyykiht{
         float: left;
         padding-right: 30px;
       }
       #sisukiht{
         float:left;
       }
       #jalusekiht{
         clear: left;
       }
    </style>
    <script type="text/javascript" src="ckeditor/ckeditor.js"></script>
  </head>
  <body>
    <div id="menyykiht">
        <h2>Teated</h2>
        <ul>
          <?php
             $kask=$yhendus->prepare("SELECT id, pealkiri FROM lehed");
             $kask->bind_result($id, $pealkiri);
```

```
          $kask->execute();
          while($kask->fetch()){
            echo "<li><a href='$_SERVER[PHP_SELF]?id=$id'>".
                 htmlspecialchars($pealkiri)."</a></li>";
          }
       ?>
     </ul>
     <a href='<?="$_SERVER[PHP_SELF]?lisamine=jah" ?>'>Lisa ...</a>
</div>
<div id="sisukiht">
   <?php
     if(isSet($_REQUEST["id"])){
        $kask=$yhendus->prepare("SELECT id, pealkiri, sisu FROM lehed WHERE id=?");
        $kask->bind_param("i", $_REQUEST["id"]);
        $kask->bind_result($id, $pealkiri, $sisu);
        $kask->execute();
        if($kask->fetch()){
         if(isSet($_REQUEST["muutmine"])){
            echo "
               <form action='$_SERVER[PHP_SELF]'>
                 <input type='hidden' name='muutmisid' value='$id'/>
                 <h2>Teate muutmine</h2>
                 <dl>
                   <dt>Pealkiri:</dt>
                   <dd>
                     <input type='text' name='pealkiri' value='".
                              htmlspecialchars($pealkiri)."'/>
                   </dd>
                   <dt>Teate sisu:</dt>
                   <dd>
                     <textarea rows='20' cols='30'
                         class='ckeditor' name='sisu'>$sisu</textarea>
                   </dd>
                 </dl>
                 <input type='submit' value='Muuda' />
               </form>
            ";
         } else {
          echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
          echo $sisu;
          echo "<br /><a href='$_SERVER[PHP_SELF]?kustutusid=$id'>kustuta</a> ";
          echo "<a href='$_SERVER[PHP_SELF]?id=$id&amp;muutmine=jah'>muuda</a>";
         }
        } else {
          echo "Vigased andmed.";
        }
     }
     if(isSet($_REQUEST["lisamine"])){
       ?>
         <form action='<?=$_SERVER["PHP_SELF"] ?>'>
          <input type="hidden" name="uusleht" value="jah" />
          <h2>Uue teate lisamine</h2>
          <dl>
            <dt>Pealkiri:</dt>
            <dd>
             <input type="text" name="pealkiri" />
            </dd>
            <dt>Teate sisu:</dt>
            <dd>
              <textarea rows="20" cols="30" class="ckeditor" name="sisu"></textarea>
            </dd>
           </dl>
           <input type="submit" value="sisesta" />
         </form>
       <?php
     }
   ?>
</div>
<div id="jalusekiht">
   Lehe tegi Jaagup
</div>
```

```
  </body>
</html>
<?php
  $yhendus->close();
?>
```

Ning näide CKEditori abil kujundatud teatest.



Ülesandeid

* Tutvu lehel http://ckeditor.com/ redaktori demoga

* Loo või otsi rakendus koerte andmete sisestamiseks ja vaatamiseks
* Võimalda koera kirjeldus kujundada graafilise redaktori abil.

* Otsi võimalusel ka teisi Javaskripti-põhiseid redaktoreid ning katseta neid

Veebilehestiku lõikude hoidmine andmebaasis

Siiani tehtud näidetes on muudetav sisu olnud lehestiku struktuuri mõttes samas kohas - teateid on kord rohkem kord vähem, kuid nad saab ikka ühest loetelust kätte.

Kui kellegi tarbeks veebilehestikku teha, siis on küllalt tavaline, et sisu soovitakse muuta mitmesugustes kohtades - täiesti erinevatel lehtedel ning ka lehe paigutuse suhtes mitmesugustes kohtades. Üheks võimaluseks sellist lehestikku veebi kaudu hallatavaks teha on hoida kõik

muudetavad kohad teadetena andmebaasis ning lehe näitamisel siis vastavad sisud sealt välja võtta. Nõnda on lehe haldus suhteliselt sarnane siinsele teadete haldusele. Lehte sisu poolest näidatakse just sellisena nagu ta tehtud on ning tekstide muutmise võimalus ei sea kuigivõrd piiranguid lehestiku ülesehitusele. Lihtsalt muudetavaid kohti peab olema piiratud hulk- et nende hilisem otsimine teadete menüüst ei osutuks üle jõu käivaks. Viisaka plokkide nimetamise abil saab ka sellest murest üle.

Funktsioonid eraldi failis

Väiksema lehestiku puhul kannatab enamiku vajaminevast ühte faili kokku kirjutada. Mõnda aega on niimoodi hea ülevaatlik. Ning uue rakenduse loomiseks piisab lihtsalt vastava lehe kopeerimisest. Suurema rakenduse puhul aga kipuks nõnda ühes failis olevate koodiridade hulk keerukalt suureks minema. Ligikaudu tuhatkond rida on suurus, mida kannatab hea süsteemi korral veel ühes failis hallata. Üle selle kipub igatmoodi kirjuks minema. Kui aga funktsioonid teemade kaupa eraldi failidesse - põhjalikuma struktueerituse puhul ka klassidesse - paigutada, siis on lootust veel tükk aega rahus elada enne, kui kood päriselt üle pea kasvama kipub.

Samuti on funktsioonide eraldamise eeliseks, et siis on kujundus võimalikult ühes failis ja programmeerimise pool teises. Nii ei teki liialt palju nii kujundajate kui ka arendajate poolt kirutud "spageti-koodi". Erinevalt mõnest muust keelest (ASP.NET, Zope lehemallid) ei ole PHPl kindlat ja üheselt tunnustatud koodi ja kujunduse eraldamise tava välja kujunenud, kuid kodeerimispraktikaid ja mallisüsteeme on loodud päris mitmesuguseid.

Siin näites loodi eraldi fail "sisufunktsioonid.php", kuhu esimeses järjekorras pannakse funktsioon teate sisu küsimiseks vastavalt ploki id-numbrile. Lehtede tabelist küsitakse otsitud teate sisu ning tagastatakse funktsioonist.

sisufunktsioonid.php

```
<?php
   $yhendus=new mysqli("localhost", "jaagup", "xxxxxx", "jaagup");
   function kysiSisu($id){
      global $yhendus;
      $kask=$yhendus->prepare("SELECT sisu FROM lehed WHERE id=?");
      $kask->bind_param("i", $id);
      $kask->bind_result($sisu);
      $kask->execute();
      if($kask->fetch()){
        return $sisu;
      }
      return "Andmed kadunud";
   }
?>
```

Avalehel - või ka mujal - teadete näitamiseks tuleb hoolitseda, et sisufunktsioonide fail oleks sisse loetud. Edasi saab teate andmed näidata andes funktsioonile ette vastava teatekoha identifikaatori. Kui kord teated valmis teha, siis edaspidi võib neile juba julgesti id järgi viidata. Kuni teadet vaid muudetakse, aga ei kustutata, senikauaks jääb id püsima. Nõnda siis piisabki hallatava sisuploki nägemiseks vaid kahest reast

```
require_once("sisufunktsioonid.php");
echo kysiSisu(16);
```

avaleht.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml”>
  <head>
       <title>Autobaasi avaleht</title>
       <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  </head>
  <body>
      <?php require("menyy.php") ?>
      <h1>Tore vahva asjalik autobaas</h1>
      <div>
         Meie uued uudised:
         <?php
           require_once("sisufunktsioonid.php");
           echo kysiSisu(16);
         ?>
      </div>
  </body>
</html>
```

Menüüfail lehtede vahel navigeerimiseks

menyy.php

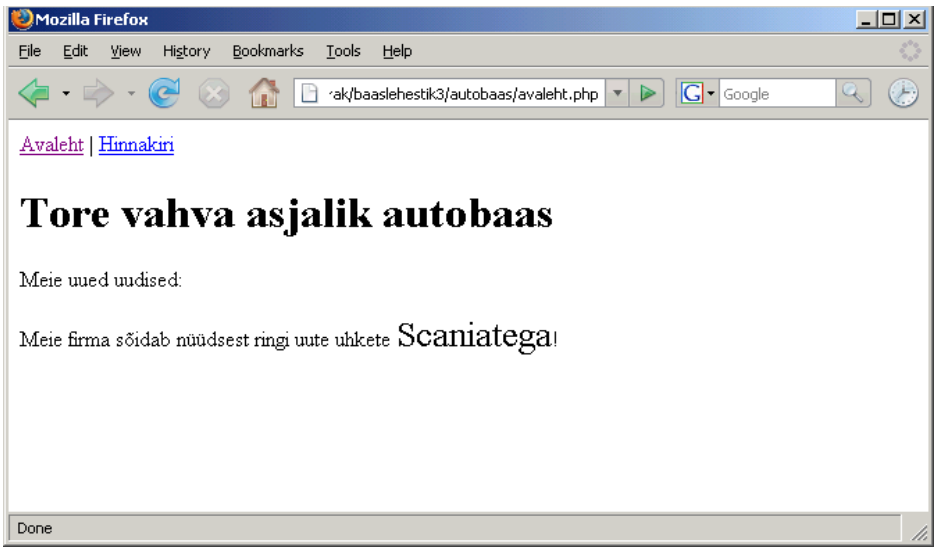
```
<div>
  <a href="avaleht.php">Avaleht</a> | <a href="hinnakirjaleht.php">Hinnakiri</a>
</div>
```

Hinnakirjalehel kasutatakse samuti loodud funktsiooni sisu küsimiseks. Lihtsalt teate kood on erinev.

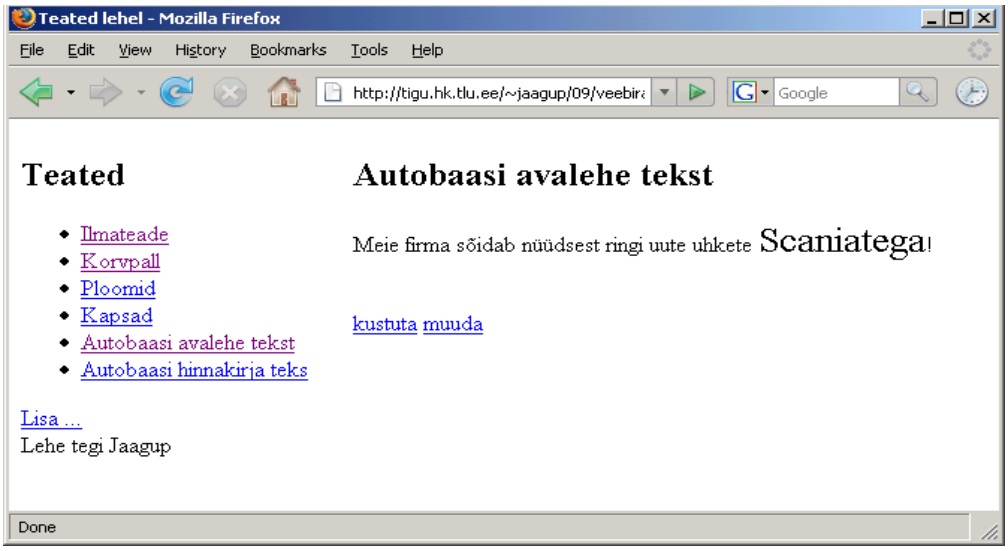
hinnakirjaleht.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml”>
  <head>
       <title>Autobaasi avaleht</title>
       <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  </head>
  <body>
      <?php require("menyy.php") ?>
      <h1>Hinnakirja leht</h1>
      <div>
         Meie kehtiv hinnakiri:
         <?php
           require_once("sisufunktsioonid.php");
           echo kysiSisu(19);
         ?>
      </div>
  </body>
</html>
```

Kui teated paigas, siis pole enam muud muret kui lehte vaatama asuda. Eks viisakas teemakohane kujundus ka külge - aga see on juba staatilise kujundamise asi millega mõnigi hakkama saab. Kui tegemist on peaasjalikult tutvustava/teatava lehega ning olulist infosüsteemi selle taga pole, siis teadete haldamise oskusest + nende lehel näitamise oskustest piisab peaaegu ükskõik kui ilusa ja keerulise lehestiku kokkupanekust. Autobaasi veebilehe redigeeritava avalehe demo.



Ning lehe muutmine käib praegusel juhul täiesti vabalt sama teadete halduse lehestikku kasutades.



Ülesandeid

* Tee autobaasi lehestiku näide läbi
* Kavanda ja koosta temaatiline kujundatud veebilehestik, kus mitmes vajalikus kohas saab sisu veebi kaudu muuta.
Andmeid hoia eraldi selle rakenduse jaoks loodud tabelis, et ei tekiks segadusi seostest muude lehtedega.
* Võimaluse korral võta kujunduse aluseks mõni juba valmis varem olev viimistletud kujundus (nt. lehelt http://www.freecsstemplates.org/ või midagi isetehtut). Lehtede muudetavad sisuosa plokid võta andmetabeli kirjetest eelnevalt näidatud moel.