Andmebaasitabeli veebiväljund

Kord valmis tehtud veebilehti saab mugavasti veebist pärast vaadata. Kui aga tahta lehtede sisu vahetevahel muuta, või siis vastavalt kasutaja soovidele mitmesuguses järjestuses või moel kuvada - sellisel puhul aitavad andmebaaside võimalused lehestiku loomisele märgatavalt kaasa.

Enamikes andmebaasides paiknevad andmed relatsiooniliselt ehk tabelite kujul. Lihtsamal juhul on veebirakenduse juures tegemist ühe andmetabeliga. PHPga koos kasutatakse sageli MySQLi nimelist andmebaasiprogrammi, sest nad on mõeldud suhteliselt sarnasele sihtgrupile. Kokku mõned gigabaidid andmeid salvestatuna ning mõned päringud sekundis on süsteemile üldiselt jõukohased. Mahtude kasvamisel kordades aga tasub juba põhjalikuma serveri optimeerimise või muude vahendite peale mõelda.

Harjutamiseks saab serveri mugavasti püsti WAMP või XAMPP-nimelise komplekti abil. Andmebaasi kasutajaliidese eest aitab sealjuures hoolitseda PhpMyAdmin. Samas enamasti võimalik suhtlus ka käsurealt.

SQL

Andmebaasiga suhtlemiseks kasutatakse SQL-keelt. Selles leiduvad käsud andmetabelite loomiseks, sinna andmete lisamiseks, andmete küsimiseks, muutmiseks ja kustutamiseks.

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 või PHPMyAdmini aknasse 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 |
+----+-----------+-------------------------+
```

Ülesandeid

- Tee näide läbi, lisa veel mõned read ja kustuta neid.
- Loo tabel kassid tulpadega id, kassinimi, toon
- Lisa paar kassi
- Väljasta kassid
- Väljasta kassid toonide järjekorras
- Kustuta üks kass

Tabeli sisu vaatamine

Andmebaasitabelis kannatab andmeid hoida ning SQL-käskude või mõne haldusliidese kaudu saab neid ka sinna lisada või sealt vaadata. Tavakasutaja aga eeldab, et ta näeb või sisestab veebilehitseja aknast just seda mis talle vaja ning ei taha ega jõua end koormata mitmesuguste tehniliste trikkidega. Veebirakenduse loojate üks tähtis ülesanne ongi andmed sobival kujul kasutajale ette näidata ning samuti veebilehtedel tehtud muutused pärast tabelites järgmiste kasutuskordade tarbeks talletada.

Sarnaselt kui ise SQL-käsklusi andmebaasi käsureale kirjutades õnnestub andmeid lisada ja küsida, saab ka PHP andmebaasiga SQL-käskude kaudu sidet pidada. Üheks võimaluseks käske veebilehe koodist andmebaasini vahendada on teek nimega MySQL Improved. Nii nagu käsitsi andmebaasiga suheldes peab teadma, kus masinas baas asub, millise kasutajanime ja parooliga sinna ligi pääseb ning millise nimega baasiga on tegemist - samad andmed vaja teada ka PHP poolt ühendust luues. Kui PHP ja MySQL asuvad samas masinas, siis sobib baasiserveri nimeks localhost. Siin näites pruugin kasutajanimeks ja parooliks juku ning kala. Ja baasi nimeks siin jukubaas2. Eks oma lahendust luues tule siis need väärtused sisse kirjutada, mis parajasti pruukida on või teenusepakkujalt antakse. XAMPP vaikimisi seadete korral sobib näiteks serveriks "localhost", kasutajaks "root", parooliks tühi tekst "" ning katsetada saab baasis nimega "test". Avalikuks väljapanekuks pole selline komplekt küll viisakas, aga oma arvutis toimetamiseks käib küll.

Edasi tuleb andmete kättesaamiseks mitu sammu ette võtta. Mõne vahendiga saab veidi lihtsamalt,

aga MySQL Improved teegi eeliseks on, et kui andmed viisakalt ette valmistada ning andmete SQL-käskudesse panekuks kasutada eelkompileeritud käsklusi (prepared statement), siis pole karta, et pahatahtlikke sisestuste abil veebilehtedelt saaks suuremat kurja teha. Muidu on aastaid olnud probleemiks, et kavalad veebilehtedel sisse kirjutatud laused võivad serveris käima minna ning pahandust tekitada. Lihtsamal juhul oma rakenduse andmeid kustutades või muutes, kuid keerukamatel juhtudel võivad löögi alla sattuda ka teiste rakenduste andmed või lausa välised serverid, kui kord sisse murtud masinat edasiste rünnakute alusena kasutatakse. Seetõttu siis siin matejalis andmete vahendajaks MySQL Improved tüüpi objekt, mida luuakse käsuga new mysqli ja antakse vajalikud ühendumisparameetrid kaasa.

Järgneva prepare-lausega palutakse $yhendus-nimelises muutuja kaudu kättesaadaval mysqli-objektil ette valmistada SQL-lause lehtede andmetabelist id, pealkirja ja sisu küsimiseks. Edasine bind_result määrab, kuhu muutujatesse saadud andmed pannakse. Andmebaasiga suhtlevad vahendid tehakse nõnda, et nad suudaksid toimida ka väga suurte andmekoguste korral ning ei loeks ilmaasjata suuremat kogust väärtusi mällu. Näiteks miljon rida on andmebaasis hoidmise jaoks täiesti kõlbulik kogus. Korraga mällu lugemisel võtab miljon kirjet aga hulga megabaite ning sealt hiljem midagi vajalikku kätte saada võib tülikas olla.

Edasi tulev execute() paneb käskluse baasis käima. Õnnetusena ei hoiatata, kui see lause unustatakse, aga lihtsalt andmeid ei saa kätte.

Vahepeal on mõningane osa HTML-i lehe kujunduse kuvamiseks. Baasis tulevate andmetega hakatakse tegelema siis, kui saabub tsükkel while($kask->fetch()). Iga fetch-käsklus tõstab päringu vastuste juurest ühe rea bind_param-käsuga määratud muutujatesse ning nendega võib tsüklikringi jooksul vajalikud toimetused ette võtta. Nõnda on korraga muutujate kaudu mälus vaid ühe andmerea ehk lehe andmed ning rakendus ei võta serveri mälu kuigivõrd. Praegu trükitakse pealkiri lihtsalt <h2> ja </h2> vahele ning näidatakse seetõttu suurema ja rasvasena välja. Sisu tuleb tavalise lõigu ehk div-ina. Käsk htmlspecialchars aitab hoolitseda, et kogemata andmete hulka sattunud erisümbolid (peamiselt < ja >) ei tekitaks lehe ülesehituse juures segadust.

Lehe väljastuse lõppemisel on viisakas andmebaasiühendus kinni panna.

Lehti väljastav kood tervikuna:

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas2");
  $kask=$yhendus->prepare("SELECT id, pealkiri, sisu FROM lehed");
  $kask->bind_result($id, $pealkiri, $sisu);
  $kask->execute();
?>
<!doctype html>
<html>
  <head>
     <title>Teated lehel</title>
  </head>
  <body>
     <h1>Teadete loetelu</h1>
     <?php
       while($kask->fetch()){
         echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
         echo "<div>".htmlspecialchars($sisu)."</div>";
       }
     ?>
  </body>
</html>
<?php
  $yhendus->close();
?>
```

Ning pilt valminud veebilehest:

Teadete loetelu

Ilmateade

Kuiv ilm

Korvpall

Treening reedel kell 18

Matemaatika

Homme tunnikontroll

Emakeel

Kontrolltöö

Ülesandeid

- Tee näide läbi
- Loo/otsi üles tabel kassid (id, kassinimi, toon). Näita kasside andmed veebilehele.
- Pane kasside toonideks inglisekeelsed värvinimetused
- Näita iga kass lehel vastavat värvi.

Teadete valik

Mõnekümne kassi andmed mahuvad ühele lehele ära. Aga seda vaid juhul, kui näidatakse vaid kassi nime ja värvi. Kui juba lisada omaniku andmed ning veidigi suurem kassi pilt, siis veidigi väiksema ekraani peal on mugav juba kasse ühekaupa vaadata. Järgnevalt uurimegi, kuidas selliseid lehti koostada, kus võimalik tabeli ühe rea andmeid eraldi välja tuua.

Üsna mugav on lehele andmeid saata aadressiriba kaudu. Kui kirjutan failinimele taha küsimärgi ning sinna taha id=2 ehk siis nt. teadetevalik.php?id=2 , siis selle väärtuse 2 saan programmis küsida muutujast $_REQUEST["id"]. Või kui tahan kontrollida, kas failinime järel saadeti parameeter nimega id, siis kontrollin `if(isSet($_REQUEST["id"]))`

Nõnda ka järgmises lõigus. Kui parameeter saadeti, siis järelikult soovitakse vaadata ühe konkreetse lehe andmeid, mida id näitab. Kui aga parameetrit pole, siis inimene järelikult ei tea veel lehte selle numbri järele küsida ning tal on põhjust pigem loetelust omale sobiv valida.

Kui id on olemas, siis saab selle järgi küsida lehe muud andmed - praeguses näites pealkirja ja sisu. Tavalise SQL-lause juures saab ühe rea küsimiseks panna WHERE-tingimuse juurde vastava piirangu. Nt `SELECT id, pealkiri, sisu FROM lehed WHERE id=2;`

Kuna siin veebirakenduses tahetakse vastavalt kasutaja valikult näha erinevaid lehti, siis peab saama seda arvu muuta. MySQL Improved teek lubab muutuva väärtuse kohale panna küsimärgi ning pärast selle väärtuse bind_param-käsu abil asendada. Hiljem tulev rida

```
$kask->bind_param("i", $_REQUEST["id"]);
```

teatab, et parameetri tüübiks on täisarv ehk integer ehk täht i. Ning parameeter saab oma väärtuse muutujast `$_REQUEST["id"]`. Edasi juba andmete kättesaamine bind_result kaudu määratud muutujatesse nagu ennegi. Eelnevas näites võis andmeid tulla palju ning seetõttu tuli nad while-tsükli kaudu välja kuvada. Ühe id järgi küsides saab kätte ainult ühe rea, seetõttu piisab selle kättesaamiseks ühest fetch-käsklusest. Kas küsimine õnnestus, seda annab teada if-lause. Andmeid ei saa küsides näiteks juhul, kui keegi on aadressirea kaudu sisestanud olematu lehe id-numbri. Muul juhul saab pealkirja ja sisu ilusti kätte ning neid võib lehel kuvada.

```
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.";
}
```

Tavakasutaja ei pea peast lehtede numbreid teadma. Tema pigem vaatab neid menüüst ning valib sobiva. Edasi juba saadetakse vastava teate id-number aadressiriba kaudu lehele, leht avaneb uuesti ning näitab küsitud teate sisu. Menüü kokku saamiseks sobib järgnev koodilõik. Kui viites (a href) jätta faili nime kohale küsimärk, siis avatakse sama fail ilma, et peaks selle faili nime teadma.

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

Edasi kirjete kaupa näitav kood tervikuna.

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas2");
?>
<!doctype html>
<html>
  <head>
    <title>Teated lehel</title>
    <style type="text/css">
```

```
      #menyykiht{
        float: left;
        padding-right: 30px;
      }
      #sisukiht{
        float:left;
      }
      #jalusekiht{
        clear: left;
      }
    </style>
      <meta charset="utf-8" />
  </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='?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();
?>
```

Veebilehel kõigepealt näha menüü ning soovitus teema valida

Tere tulemast avalehele! Vali menüüst sobiv teema.

Teated

- Ilmateade
- Korvpall
- Matemaatika
- Emakeel

Lehe tegi Jaagup

Edasi juba tuleb vastava teema id aadressiribale ning näeb valitud pealkirja all peituvat sisu.

Teated

Korvpall

- Ilmateade
- Korvpall
- Matemaatika
- Emakeel

Treening reedel kell 18

Lehe tegi Jaagup

Ülesandeid

- Tee näide läbi
- Loo tabel koerte kohta (id, koeranimi, kirjeldus, pildiaadress)
- Sisesta andmed mõnede koerte kohta, pildid otsi veebist
- Loo lehestik, kus vasakus servas näha koerte nimed. Nimele vajutades kuvatakse lehel suurelt see koer koos pildi ja kirjeldusega.

Andmete lisamine ja kustutamine

Veebi kaudu on andmeid ilus vaadata. Ainult, et sellisena saab lehed ka ilma serveripoolse programmeerimistoeta tööle panna. Vajadusel saab kindla arvu lehti kopeerida ning viited vastavalt sättida ning võibki andmeid soovitult lugeda. Kui aga tahta, et kasutajapoolsed andmed ka kuidagi serverisse talletuks ning teised neid lugeda saaks - seda juba naljalt ilma serveripoolse programmita teha ei õnnestu. Muidugi kaasnevad serveris talletamisega ka omad mured: keegi võib hooletusest või pahatahtlikkusest sinna hulgem andmeid saata ning sellega serveris oleva andmebaasi täis kirjutada või lihtsalt kahtlaste postitustega suure hulga segadust tekitada. Aga hea ja halb käivad

käsikäes ning mugavuse nimel tuleb vahel ka mõnevõrra riskida. Abilisteks hiljem varukoopiad, registreerimised, modereerimised ja muud täiendused.

Andmete lisamiseks tuleb need kõigepealt kasutajalt kätte saada. Selleks sobib sisestusvorm - olgu siis pidevalt lehel nähtaval või eraldi viite peale näidatav. Sisestusvormist tulevad andmed saadetakse salvestamiseks serverisse. Siinses näites toimib kõik sama faili kaudu, kuid iseenesest võib toimetuse jaoks ka eraldi teine väike fail loodud olla. Pärast andmete salvestamist on kasulik leht uuesti edasi suunata - kas või samale lehele, aga nõnda, et inimese sisestatud andmed uuesti kaasa ei tuleks - sellisel juhul pole karta, et värskendusnupu vajutamine andmeid korduvalt salvestama hakkab.

Lisamisvormi nähtavaks muutumiseks loodi viide parameetriga lisamine.

```
<a href='?lisamine=jah'>Lisa ...</a>
```

Kui selline parameeter jõuab serverisse, siis näidatakse kasutajale tühjad lahtrid, kuhu oma andmed kirja panna. Definition list (dl) koos nimetuse (definition term, dt) ning sisuga (dd, definition data) võimaldab mugavalt sisestuselemendid koos seletustega välja kuvada. Kaasas on ka varjatud element nimega `uusleht`, mille abil siis hiljem kontrollida, et kasutaja on uue lehe andmed saatnud. Andmete teele panekuks veebis nupp tüübist submit.

```
 if(isSet($_REQUEST["lisamine"])){
   ?>
     <form action='?'>
      <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
  }
?>
```

Nupule vajutades avatakse leht uuesti. Kaasa liiguvad eelnevalt väljadesse sisestatud andmed. Eelnevalt varjatult kaasa pandud parameeter nimega uusleht näitab, et nüüd on paras aeg saabuvad andmed tabelisse kirjutada. Andmete lisamiseks tabelisse on INSERT-lause. Lisatavate väärtuste kohta tulevad algul küsimärgid, bind_param-käsu abil paigutatakse nende asemele tegelikud väärtused. Tekst "ss" bind_param-käsu esimese parameetrina näitab, et mõlemad saabuvad väärtused on stringi ehk teksti tüüpi. Väärtusteks on siis pealkiri ja sisu, mis $_REQUEST-muutujast sisse loetakse. Vältimaks lehe korduslaadimisel uuesti salvestamist, tasub Location-päisekäsuga lehe avamine edasi suunata - kas või samale lehele ($_SERVER[PHP_SELF]). Viisakasti siis andmebaasiühendus ka sealjuures kinni ning lehe avamisele lõpp - 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();
  }
```

Näites lisati ka kustutamise moodus. Eraldi vaatamise lehel sai juurde kustutamise viide, kus aadressiga suunatakse samale lehele ning antakse kaasa `kustutusid`.

```
              echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
              echo htmlspecialchars($sisu);
              echo "<br /><a href='?kustutusid=$id'>kustuta</a>";
```

Lehe päises kustutusid saabumisel käivitatakse `DELETE`-lause koos etteantud kirje numbriga.

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

Lisamis- ja kustutusvõimeline kood tervikuna.

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas2");
  if(isSet($_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>
<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='?id=$id'>".
                          htmlspecialchars($pealkiri)."</a></li>";
            }
         ?>
       </ul>
       <a href='?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='?kustutusid=$id'>kustuta</a>";
            } else {
              echo "Vigased andmed.";
            }
         }
         if(isSet($_REQUEST["lisamine"])){
           ?>
             <form action='?'>
              <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();
?>
```

Lehel algul näha teadete loetelu.

Teated

- Ilmateade
- Korvpall
- Matemaatika
- Emakeel

Lisa ...
Lehe tegi Jaagup

Pealkirjale vajutades näeb vastava kirje andmeid. Kustutusviite kaudu saab kirjest lahti.

Teated

- Ilmateade
- Korvpall
- Matemaatika
- Emakeel

Matemaatika

Homme tunnikontroll
kustuta

Lisa ...
Lehe tegi Jaagup

Nii pole seda teadet ka enam menüüs näha. Paistab lisamisviide uute andmete sisestamiseks.

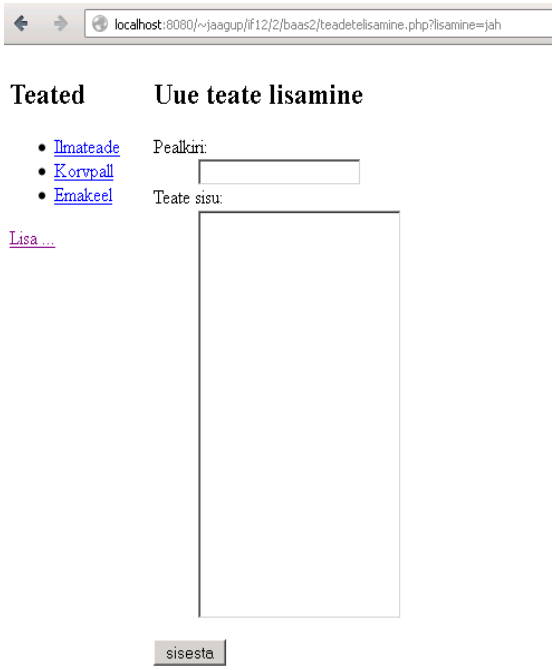
Teated

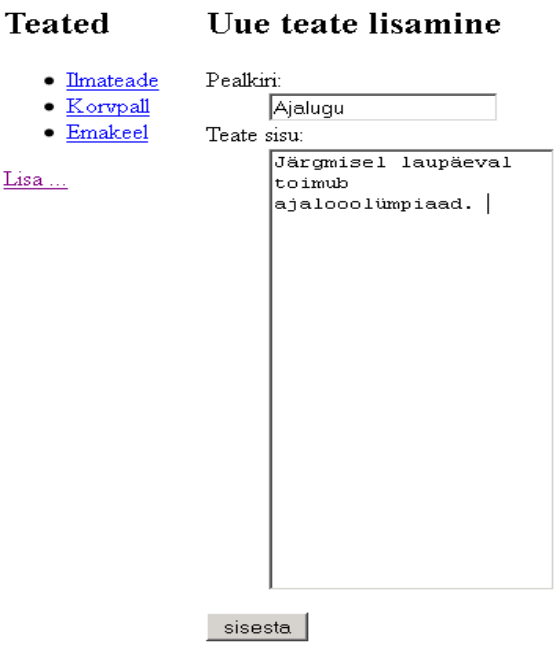
- Ilmateade
- Korvpall
- Emakeel

Lisa ...
Lehe tegi Jaagup

Lisamisviite kaudu kuvatakse lisamisvorm

Edasi tuleb need andmed sisse kirjutada.





Pärast sisestamist võibki menüüs uud teadet imetleda.

Teated

- Ilmateade
- Korvpall
- Ajalugu
- Emakeel

Lisa ...
Lehe tegi Jaagup

Ülesandeid

- Tee näide läbi
- Loo/otsi koerte tabel (id, koeranimi, kirjeldus, pildiaadress)
- Võimalda koeri veebi kaudu lisada, vaadata ja kustutada

Peoõhtu registreerimisvorm

- Koosta veebileht peokuulutusega
- Loo sinna juurde veebileht, kus kasutaja saab oma eesnime, perekonnanime ja elektronposti sisestada. Andmed talletatakse tabelisse (ei näidata veebilehel).
- Loo eraldi administraatorileht, kus saab sisestusi näha (sisselogimist pole vaja)
- Administraator saab vigaseid sisestusi ka kustutada
- Loo teine andmetabel, kus kirjas peo etteasted ja sündmused koos arvatava kellaajaga.

Väljasta andmed kellaaegade järjekorras eraldi veebilehele. Kujunda veebileht koos eelmistega ühtseks lehekstikuks.

- Loo eraldi administraatorileht peo sündmuste lisamiseks ja kustutamiseks.