Sissejuhatus

Veeb tänapäevasel kujul asus jõudsalt levima 1990ndate aastate keskel. Suurelt jaolt kasutati seda küll staatiliste tekstide ja piltide mugavaks välja näitamiseks, kuid juba algusest peale olid kasutusel juures lisad, mis võimaldasid lehe sisu vastavalt kasutajale või tema tegevusele eraldi näidata. Levinumate näitena selle kohta kehtiva kuupäeva näitamine või otsing soovitud andmete alusel. Sellisel juhul ei piisa paljast valmistehtud lehe kopeerimisest kasutajale, vaid tuleb lehe sisu tekitada või ühendada mõne programmi võimaluste abil.

Vahendeid selleks on aegade jooksul olnud mitmesuguseid. Siiani kasutusel on võimalus käivitada suvalises meeldivas keeles koostatud programm veebiserveris ning lasta sel suhelda brauserist tulnud päringuga vastava liidese (CGI – Common Gateway Interface) abil. Keskkonnamuutujate kaudu saadakse kätte edastatavad andmed (näiteks otsisõna) ning programmi trükitav väljund suunatakse brauseri poole teele – tehniliselt nõnda lihtne see ongi. Iga programmeerija võis endiselt kirjutada omale sobivas keeles ning kõik toimis.

Veebisisendi ja -väljundiga programmidel on aga mõned eripärad. Olenevalt rakendusest, kuid küllalt sageli tuleb väljastada suures koguses muutumatut HTML-teksti ning sinna vahele vaid üksikud kohad, mis programmiga muuta vaja. Teiseks probleemiks veebirakenduste juures on, et kunagi ei saa usaldada sisendit kasutajalt – üle veebi võib tulla ligi suvaline häkker ning otsisõna asemele panna teele näiteks mõne videofilmi sisu binaarkujul. Sekelduste vältimiseks on seetõttu kasulik lisada sisendile piiranguid ja kontrolle.

Veebi leviku laienedes lisandus ka raamistikke, keeli ja keeletäiendusi, mille abil peaks veebiprogramme olema mugavam kokku panna kui ”tavaliste” programmeerimiskeelte abil. Suure veebileviku osaliseks sai keel nimega PERL, millel võrrelduna näiteks tol ajal muidu väga levinud C-ga olid tunduvalt mugavamad ja mitmekülgsemad vahendid tekstidega ümber käimiseks. Tuntumad eraldi veebi jaoks loodud vahendid ehk ASP (nüüdseks põhjalikult muudetuna ASP.NET), Zope, Java Servletid ja JSPd. Lihtsamate ja ka keskmiselt keerukate veebilehestike juures sai valitsevaks keeleks PHP.

PHP tutvustus

Algselt ühe koolipoisi katsetustest levima hakanud kodulehe koostamise abivahend (Personal Home Page) sai oma lihtsuse ja vabade kasutusõiguste tõttu üllatavalt populaarseks. Eks lihtsusega käivad koos ka mõned ohud, mida on uuemates versioonides püütud lappida. Kuid võlu, et kõik kohe arusaadav ja kasutatav on, paneb paljudki programmeerijad heal meelel PHPst alustama.

Päris algus

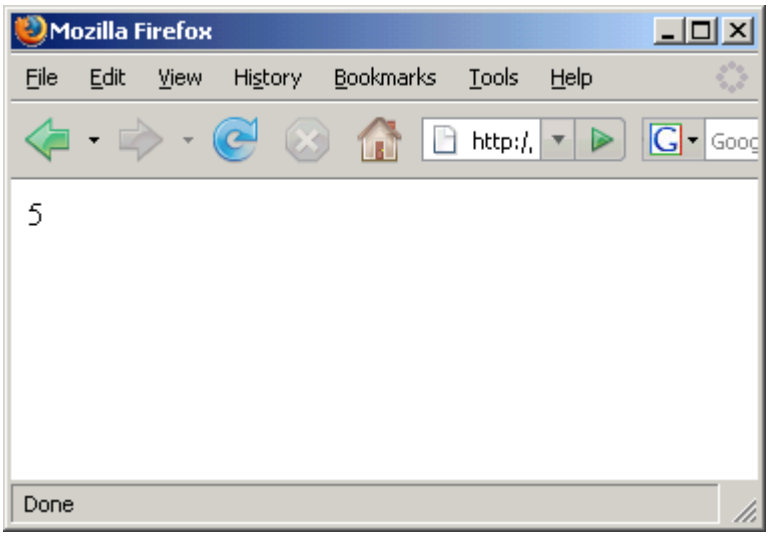
PHP-leht on tavaline teksti (või siis üldjuhul HTML-leht), kus sobivate märkide vahel saab programmikoodi käima lükata – nõnda nagu enamiku muudegi veebiprogrammeerimissüsteemide puhul. Lihtsaim demo näeb välja järgmine:

PHP-võimelise veebiserveri kaudu välja kuvatavasse kataloogi tuleb paigutada järgneva sisuga fail.

```
<?= 3+2 ?>
```

Laiendiks php – näiteks nimega algus.php

Kui nüüd leht veebilehitsejas avada, võiks seal ilutseda üks ilus suur number 5. Kolm ja kaks liideti kokku.



Väike seletus ka juurde: <?= näitab, et nüüd järgneb avaldis, mille väärtus enne kasutajale saatmist kokku arvutatakse. Avaldise lõppu tähistab ?>. Ning vahepealne 3+2 lihtsalt arvutatigi kokku ja trükiti välja. Kui lehel alguses või lõpus oleks veel muudki teksti, trükitaks ka see välja.

Ülesandeid

* Hangi või tee selgeks enesele võimalus PHP-võimelises veebiserveris veebilehtede loomiseks.
* Koosta tervitav leht ja vaata seda veebiserveri kaudu
* Muuda lehe sisu ning uuendusnupu vajutuse järel veendu muutuse kajastumises ka veebilehitsejas.
* Käivita konspektis olnud näide kahe arvu liitmise kohta.

* Muuda arve ja tehet, kontrolli tulemusi.

Muutuja, valik ja kordus

Järgnevalt juba veidi pikem koodilõik. Kui ei piirduta vaid ühe arvu või lause väljastusega, siis tuleb ploki alguseks märkida <?= asemel <?php. Siis võib kuni ?> -ga tähistatud ploki lõpuni rahulikult programmikoodi kirjutada -.see pannakse käima ning tulemus saadetakse veebilehistsejasse.

Programmeerimiskeeltes on levinud võimalus andmeid muutuja ehk märksõna alla meelde jätta. PHPs algavad muutujate nimed dollarimärgiga. See võimaldab neid hiljem vabamalt teksti sisse panna. Lõik

```
$eesnimi="Juku";
echo "Tere, $eesnimi!";
```

trükib aimatavalt välja ”Tere, Juku”.

Valiku jaoks on käsklus if. Tingimus pannakse ümarsulgude sisse. Kui tingimus vastab tõele (praegusel juhul vanus on väiksem kui seitse), siis täidetakse järgnevate looksulgude vahele paigutatud plokk.

```
$vanus=5;
if($vanus<7){
   echo "Oled noor!";
}
```

Korduse ehk silmuse ehk tsükli puhul võidakse looksulgude vahele kirjutatud toimetus ette võtta korduvalt. Järgnev näide teatab viis korda Kuku!

```
for($i=0; $i<5; $i++){
  echo "Kuku!";
}
```

Seletus ka juurde. Muutuja `$i` (PHPs algavad kõik muutujad dollarimärgiga) aitab meeles pidada, mitmenda korra juures ollakse. Käsu for ümarsulgude sees on kolm semikoolonitega eraldatud tsooni. Esimeses neist on algväärtustus, täidetakse üks kord, kohe `for`-ini jõudmise juures. Tüüpiliselt antakse siin loendurile algväärtus, praegusel juhul i-le 0.

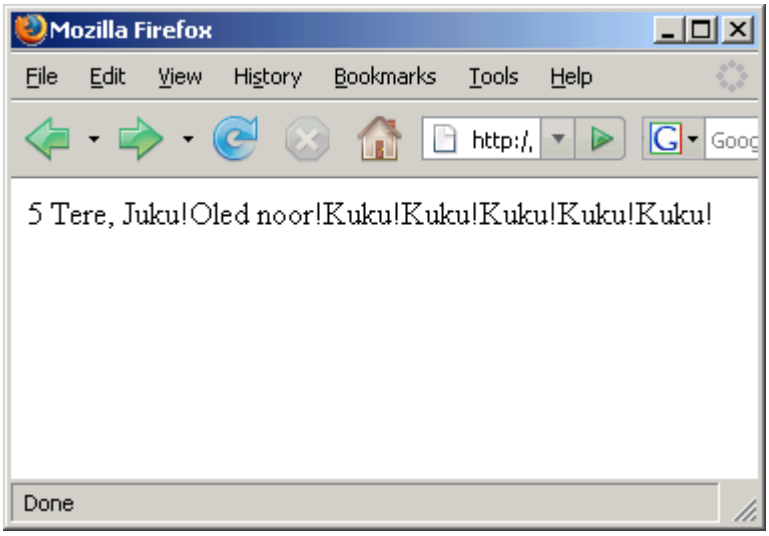
Keskmises tsoonis kontrollitakse tingimuse kaudu, et kas on põhjust `for`-ile järgnevate looksulgude vahel olevaid käske täitma hakata. Kui tingimus on tõene siis jah, muidu mitte. Võib ka juhtuda, et tingimus on juba esimesel korral väär - sellisel juhul ei täideta tsükli keha ühtegi korda.

Viimasesse ehk kolmandasse tsooni paigutatakse tsüklis edasiliikumise tegevus(ed) - siin juhul suurendatakse `$i` väärtust, et järgmisel korral oleks juba suurem arv loenduriga võrrelda. Lugemisega alustatakse tüüpiliselt nullist - siis hiljem massivide puhul kergem toimetada.

```
<?= 3+2 ?>

<?php
  $eesnimi="Juku";
  echo "Tere, $eesnimi!";

  $vanus=5;
  if($vanus<7){
```

```
      echo "Oled noor!";
  }

  for($i=0; $i<5; $i++){
    echo "Kuku!";
  }
?>
```

Koodilõigu loodud väljund järgmine:



Ülesandeid

* Käivita nähtud näide, muuda andmeid, jälgi tulemusi
* Lisa tervitatavale inimesele perekonnanimi eraldi muutujasse, trüki ka selle väärtus.
* Lisa tingimus üle saja-aastaste jaoks teatega "oled hirmus vana".
* Lisa for-tsükli sisse iga Kuku järjekorranumber.

PHP HTMLi sees

Nagu näha, võib PHP ka lihtsat teksti väljastada. Kui aga tahta kokku panna viisaka kujundusega veebilehte kus ka kasutaja midagi sisestada saab, siis tuleb HTMLi reeglitega arvestama hakata. Viisakasti tasub `<!DOCTYPE` abil ära määrata HTMLi versioon ning edasi juba kõik elemendid omadel kohtadel: väline element <html>, tema sees peamiste suurte plokkidena <head> ja <body>. Esimesse neist tulevad enamikus metaandmed ehk siis andmed dokumendi kohta, <body> sisse nähtav tekst ise.

Lihtsaim asjalik lõik PHPd HTML koodi sees võiks välja näha ehk järgmine:

```
 <?php
  echo "Kell on: ".date("H:i:s");
  ?>
```

Nagu aimata võite, võib selle tulemusena näha veebiserveri kella aega veebilehel kliendi masinas. Tekst "Kell on " tuleb välja jutumärkide vahelt. Järgnev punkt on operaator tekstide liitmiseks (sidurdamiseks). Ning käsklus date võimaldab kuupäeva ja kellaaja väljastada ettemääratud kujul. Siin näites kasutatud H tähendab tunde 24 tunni süsteemis, i minuteid (sest m ehk month on kuude

jaoks) ning s sekundeid. Koolonid trükitakse nende vahele niisama välja. Leht tervikuna siis:

```
<!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>PHP katsetused</title>
     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
 </head>
 <body>
   <h1>PHP katsetused</h1>
   <?php
      echo "Kell on: ".date("H:i:s");
   ?>
 </body>
</html>
```

Sisestusega veebileht

Lehele sisestuselementide lisamiseks on sinna kõigepealt vaja panna kujunduse mõttes nähtamatu plokk nimega form. Parameeter action näitab, millisele lehele saadetakse elementidesse kirjutatud andmed. Praeguses näites on fail ise nimega "teine.php" ning action saadab andmed ka faili "teine.php" ehk siis failile iseenesele. Kui andmed kirjutatakse väljadesse ja vajutatakse submit-nuppu, siis pannakse nad praegusel juhul kaasa avatava faili aadressiribale -

```
   <form action="teine.php">
      Eesnimi: <input type="text" name="eesnimi" />
      Vanus: <input type="text" name="vanus" />
      <input type="submit" value="Sisesta" />
   </form>
```

Järgmisel avamisel võib näha aadressiribal failinime näiteks kujul

```
teine.php?eesnimi=Juku&vanus=7
```

Järgmine samm on vormi kaudu aadressiribale jõudnud andmed kätte saada ning nendega midagi ette võtta. Andmete püüdmiseks sobib massiiv nimega $_REQUEST (teadjamatele: sinna jõuavad kokku andmed massiividest $_GET ja $_POST). Sisestatud tegelase lihtsaks tervitamiseks sobib rida

```
echo "Tere, $_REQUEST[eesnimi]";
```

Selle juures on aga probleemiks, et tervitada püütakse ka juhul, kui nime andmeid tegelikult ei ole. Sel juhul ilmuks ekraanile paljas tere koos komaga.

Kui me pole kindlad, kas andmed saabuvad või mitte, siis on viisakas nende olemasolu enne kasutamist kontrollida. Kusjuures on kaks täiesti erinevat andmete puudumise juhtu. Ühel puhul lihtsalt leht avati niisama, sisestades aadress aadressiribale. Teine puudumise puhk on juhul, kui lehele küll sisenetakse vormi kaudu andmeid sisestades ja submit-nupule vajutades, kuid tekstivälja väärtus jäeti tühjaks. Aadressiriba tekib siis kujul näiteks

```
teine.php?eesnimi=&vanus=7
```

Elemendi olemasolu kontrollib PHPs funktsioon nimega isSet, väljastades tõeväärtuse true/false. Näiteks

```
if(isSet($_REQUEST["eesnimi"])){...}
```

Sisestatud teksti olemasoluks on aga hea moodus kindlaks teha selle teksti pikkus käsuga strlen. Kui tekst juhtub tühi olema, siis lihtsalt on selle pikkus 0. Nõnda saabki kokku koodilõigu, mis esimesel avamisel ei ütle midagi. Nimelahtri tühjaksjätmisel aga teatab nime puudumisest.

```
if(isSet($_REQUEST["eesnimi"])){
   if(strlen($_REQUEST["eesnimi"])>0){
     echo "Tere, $_REQUEST[eesnimi]!";
   } else {
     echo "Nimi kirjutamata!";
   }
}
```

Eks arvude puhul saab olemasolu samamoodi kontrollida. Esialgu on ka vanus arvuti jaoks lihtsalt tekst ehk sümbolid. Seepärast ka sisestuslahter sai kirja

```
Vanus: <input type="text" name="vanus" />
```

Kui tahta kontrollida, et sinna lahtrisse saaks kirjutada vaid numbreid, siis tuleks selleks eraldi Javaskripti lõik kirjutada või mõne raamistiku abil sisse panna.

Kui serveris aga tahta saabunud arvuga viisakalt ümber käima hakata, siis on viisakas saabunud tekst mälus arvulisele kujule ümber muundada. Täisarvude puhul aitab seda teha funktsioon nimega intval. Edasi võib veebist tulnud arvuga käituda juba sarnaselt nagu iga muu arvuga. Siin siis tervitame nime sisestanut nõnda palju kordi, palju tal aastaid on.

```
if(strlen($_REQUEST["vanus"])>0){
  $v=intval($_REQUEST["vanus"]);
  for($i=0; $i<$v; $i++){
    echo "&Otilde;nne! ";
  }
}
```

Veebisisestuse üle ei saa aga kunagi kindel olla: kui juhtub, et mõni katsetaja kirjutab oma vanuseks miljoni, siis see koodijupp tervitab teda rõõmsalt miljon korda, kulutades vastavalt serveri aega. Selline tegevus pole küll serveri andmetele ohtlik, kuid kui häkkerid tahaksid hakata masinat kõvasti koormama, siis paarikümnest kohast pidevalt end miljoni kaupa tervitada laskmine on hea moodus serveri kiusamiseks. Kõike selliseid võimalusi ei jõua ega saagi kinni panna. Aga mõnigikord peab mõtlema, et kas, mida ja kui palju on põhjust turvata, ehk siis kiusajatele takistusi teha.

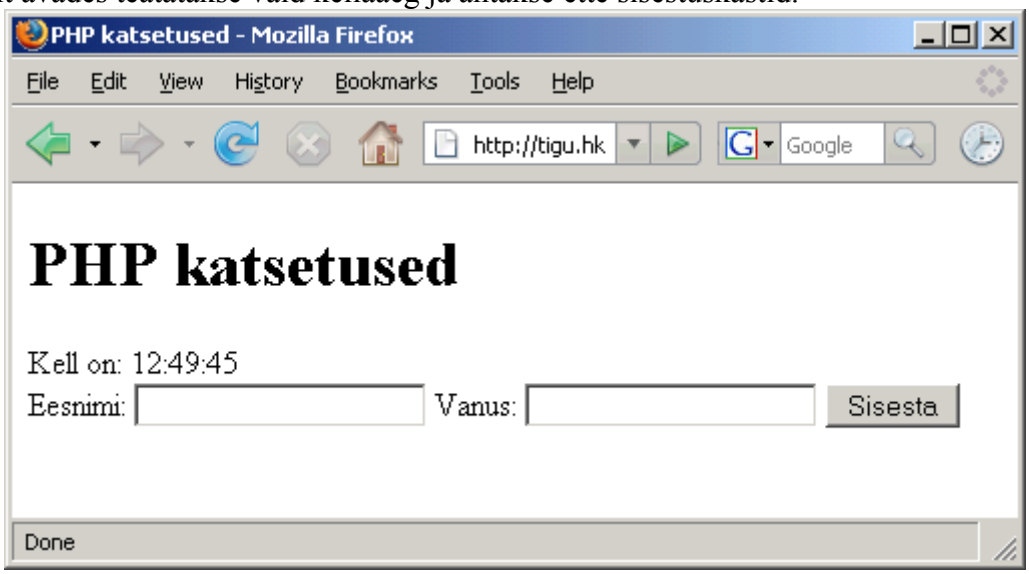
Nime ja vanuse järgi tervitav ja õnnitlev leht siis järgmine.

failinimi: teine.php

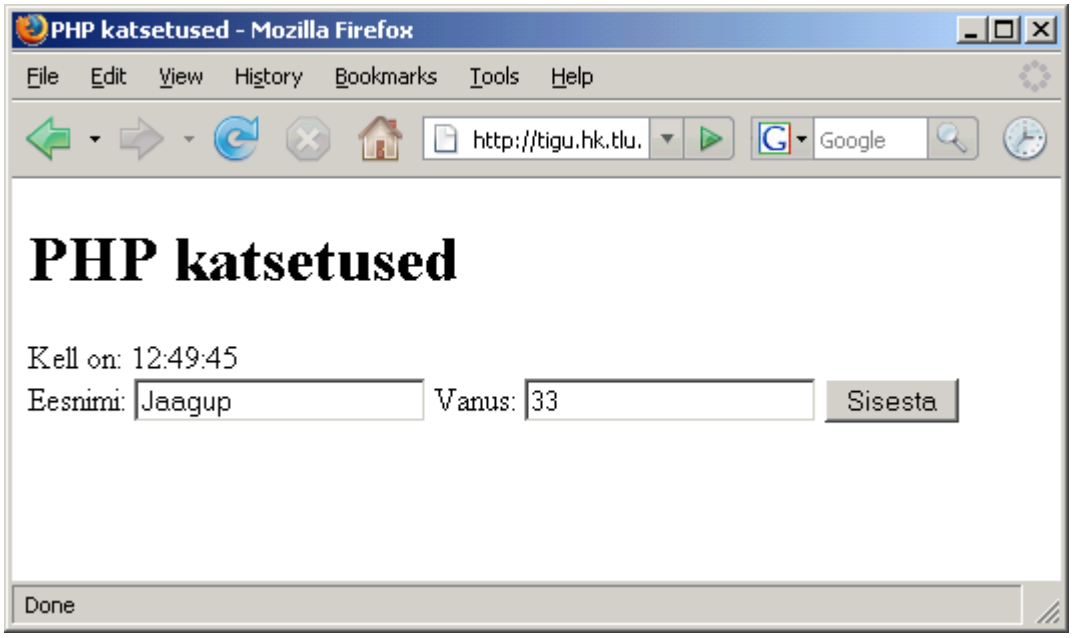
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
     <title>PHP katsetused</title>
     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
 </head>
 <body>
   <h1>PHP katsetused</h1>
   <?php
      if(isSet($_REQUEST["eesnimi"])){
         if(strlen($_REQUEST["eesnimi"])>0){
           echo "Tere, $_REQUEST[eesnimi]! ";
         } else {
           echo "Nimi kirjutamata! ";
         }
```

```
      }
      echo "Kell on: ".date("H:i:s");
   ?>
   <br />
   <?php
     if(strlen($_REQUEST["vanus"])>0){
       $v=intval($_REQUEST["vanus"]);
       for($i=0; $i<$v; $i++){
         echo "&Otilde;nne! ";
       }
     }
   ?>
   <form action="teine.php">
      Eesnimi: <input type="text" name="eesnimi" />
      Vanus: <input type="text" name="vanus" />
      <input type="submit" value="Sisesta" />
   </form>
 </body>
</html>
```

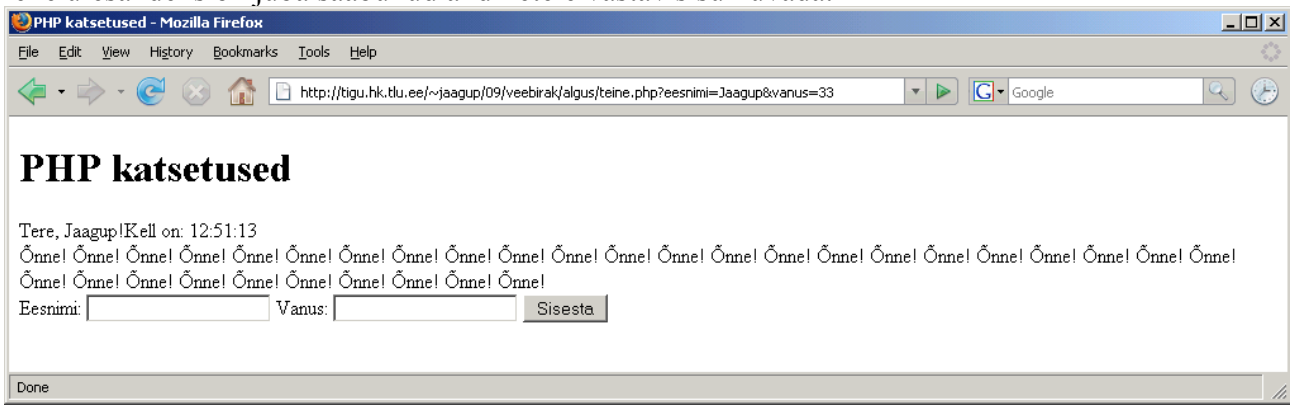
Kõigepealt avades teatatakse vaid kellaaeg ja antakse ette sisestuskastid.



Sinna kannatab andmed sisse kirjutada.



Edasi jõuavad need andmed aadressiriba kaudu faili juurde kuhu form-i action suunab. Ning selle lehe ülesandeks on juba saabunud andmetele vastav sisu kuvada.



Ülesandeid

* Lase kasutajal sisestada kaks arvu. Programm väljastab nende korrutise
* Trüki kasutaja määratud ridade ja veergude arvuga korrutustabel

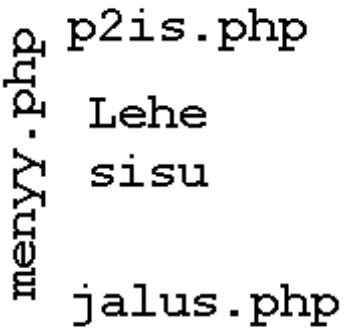
Lehe koostamine alamosadest

Aastate eest oli tähtsaks näitajaks serveri või teenusepakkuja juures, et kas too toetab SSI-d ehk Server Side Include't. Nüüd saab sama tulemuse PHP käskudega mugavasti kätte.

Paljude sarnase kujundusega lehtede juures on paratamatult hulk korduvat koodi - soovitakse samasse kohta paigutada logo, menüü ja ehk muudki. Üheks võimaluseks on algul blanketileht teha, seal tähtsamad kujunduseasjad ära määrata ning seejärel blanketist sobiv hulk koopiaid teha. Kui lehestik nõnda ühekordselt kokku pannakse, siis selline lähenemine täiesti sobib. Samuti aitavad mõned keskkonnad (nt. FrontPage) selliseid sarnaseid lehti genereerida.

Kui aga tehakse lehti lihtsa HTML-redaktoriga, samuti kui tahetakse lehe koodi ise kontrollida ning sinna hiljem programmilõikegi sisse panna, sellisel juhul on sisseloetavad alamosad igati head abilised.

Näitena koostame bussiliinide andmeid avaldava lehestiku, kus muutuvad ainult liini andmed. Lehe päis, jalus ja menüü jäävad ikka samadesse kohtadesse. Alamosad loetakse sisse järgneval joonisel näha olevatest failidest:



Päiseosa saab failist p2is.php. Sinna tuleb kogu HTMLi algusots: DOCTYPE, head ja title ning lehe keha algus.

p2is.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
 <head xmlns="http://www.w3.org/1999/xhtml">
      <title>Bussiliinid</title>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
      <style type="text/css">
        @import "kujundus.css";
      </style>
 </head>
 <body>
 <?php require("menyy.php"); ?>
```

Samuti loetakse sinna juurde sisse menüü failist menyy.php.

menyy.php

```
<div id="menyy">
  <h2>Men&uuml;&uuml;</h2>
  <ul>
     <li><a href="726.php">726</a></li>
     <li><a href="727.php">727</a></li>
  </ul>
</div>
```

Menüü on hea eraldi kohas hoida. Siis on sinna mugav lisada uusi ning eemaldada vanu lehekülgi. Samuti võib hiljem vajadusel paigutada menüükihi lehe koodis mõnda muusse, hetkel sobivamasse kohta.

Ka kujundus on paigutatud eraldi faili. Ehkki siinse sisselugemise teel saaks kujunduseosa suhteliselt mugavalt panna ka päisefaili, on vähegi pikema kujunduslõigu puhul see mõttekas panna omaette faili. Päisefaili kaudu lugedes arvatakse see kujunduslõik iga kord eraldi faili osaks olema ning sikutatakse kogupikkuses serverist kohale. Kui aga viidatakse eraldi css-failile, siis piisab selle ühe korra kohale tõmbamisest, ülejäänud osa ajast saab puhvris olevaid andmeid kasutada.

Kujundusefailis määratakse ära lehel asuvate plokkide asukohad. Käsklus float paneb kihi lehe peal "ujuma". Kuna nii menüükihil kui ka sisukihil on omadus float:left, siis nad ujuvad üksteisel sabas. Menüü on nõnda lai, nagu teksti laius teda lükkab, sisukihil on praegu määratud laiuseks 70% akna suurusest, et ta kasvaks ja kahaneks koos aknaga. Lõputeate kihi clear:left ütleb, et tuleb taas uuelt realt oma paiknemist alustada, nii jõuab ta ilusti eelmiste elementide alla.

kujundus.css

```
body{
  background-color: #ffeb90;
}

#menyy{
 float: left;
 padding-right: 30px;
}

#sisu{
 width: 70%;
 float: left;
}

#loputeade{
 clear: left;
}
```

Jalus praeguse seisuga lihtne fail, kus lõputeade ning HTML-lehe ots.

jalus.php

```
    <div id="loputeade">Lehe koostas Jaagup</div>
  </body>
</html>
```

Sisuleht

Kui abitükid olemas, siis saab asuda sisulehti koostama. Nemad loevad enesele require-käsuga sisse ette päise ja taha jaluse. Päis haarab enese külge veel kaasa menüü. Sisule ka omaette väike kiht paigutamiseks ümber ning võibki rahumeeli vajalikud andmed sinna sisse kirjutada.

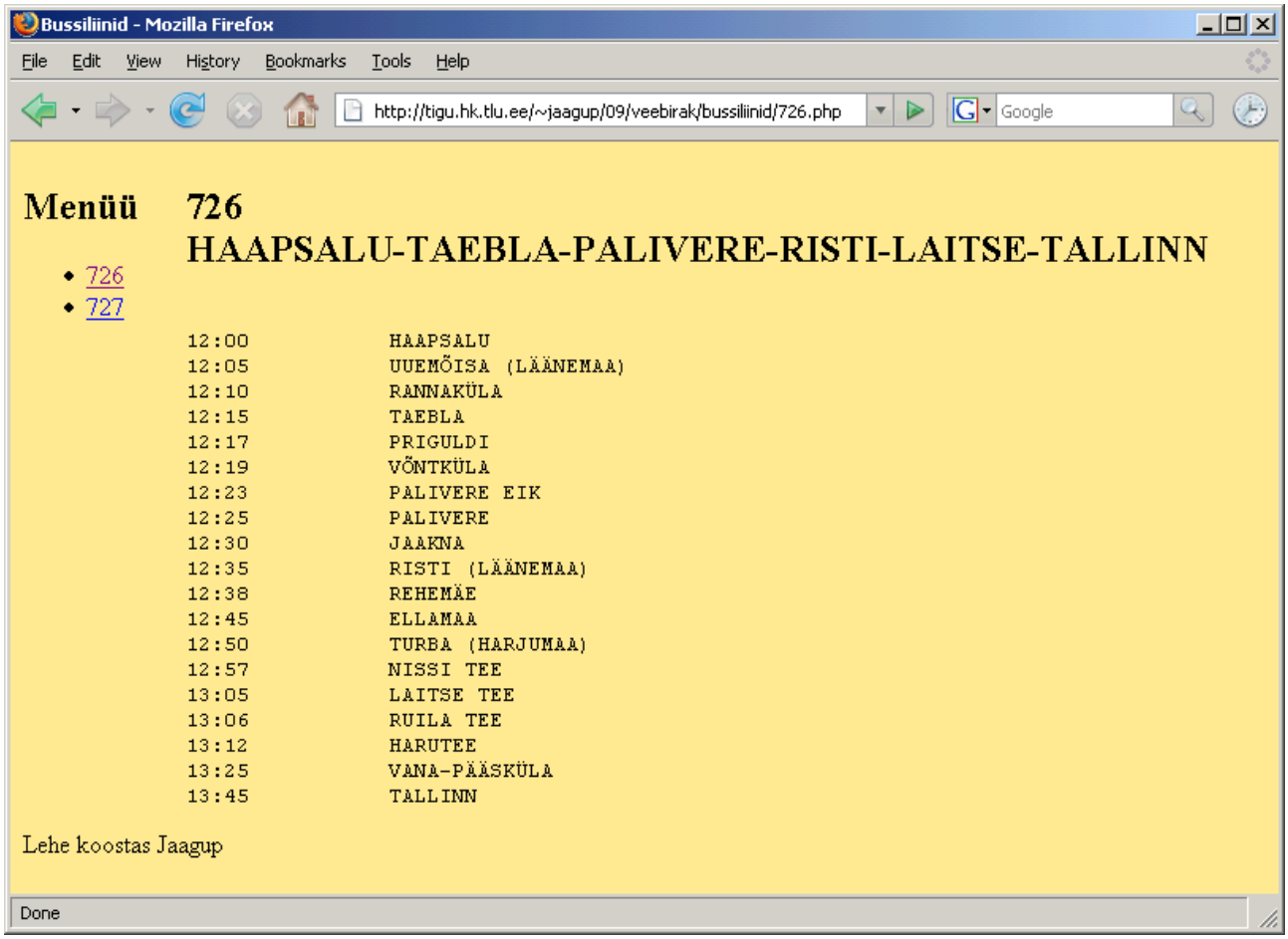
```
<?php require("p2is.php"); ?>
  <div id="sisu">
    <h2> 726 HAAPSALU-TAEBLA-PALIVERE-RISTI-LAITSE-TALLINN</h2>

<pre>
12:00              HAAPSALU
12:05        UUEMÕISA (LÄÄNEMAA)
12:10        RANNAKÜLA
12:15        TAEBLA
12:17        PRIGULDI
12:19        VÕNTKÜLA
12:23        PALIVERE EIK
12:25        PALIVERE
12:30        JAAKNA
12:35        RISTI (LÄÄNEMAA)
12:38        REHEMÄE
12:45        ELLAMAA
12:50        TURBA (HARJUMAA)
```

```
12:57        NISSI TEE
13:05        LAITSE TEE
13:06        RUILA TEE
13:12        HARUTEE
13:25        VANA-PÄÄSKÜLA
13:45        TALLINN
</pre>
  </div>
<?php require("jalus.php"); ?>
```

Kaks kõrvutist lehte erinevad vaid sisu poolest - siin siis teise bussiliini ajad ja peatused. Päis ning jalus võetakse külge samasugustena. Käsklus require erineb mõnel pool levinumast include-st selle poolest, et require annab otsitava faili puudumisel veateate, include jätab selle lihtsalt näitamata. Tegemise juures vigade vältimiseks on esimene variant kindlam.

```
<?php require("p2is.php"); ?>
  <div id="sisu">
    <h2>Liin 727, HAAPSALU-TAEBLA-PALIVERE-RISTI-TALLINN</h2>

<pre>
13:00              HAAPSALU
13:05        UUEMÕISA (LÄÄNEMAA)
13:10        RANNAKÜLA
13:15        TAEBLA
13:17        PRIGULDI
13:19        VÕNTKÜLA
13:23        PALIVERE EIK
13:25        PALIVERE
13:30        JAAKNA
13:35        RISTI (LÄÄNEMAA)
13:38        REHEMÄE
13:45        ELLAMAA
13:50        TURBA (HARJUMAA)
13:57        NISSI TEE
14:12        HARUTEE
14:25        VANA-PÄÄSKÜLA
14:45        TALLINN
</pre>
  </div>
<?php require("jalus.php"); ?>
```

Veebilehitsejas tuleb avada sisuleht. Viimane haarab omale ette ja taha vajalikud tükid külge ning võibki rahumeeli sõiduplaani vaadata.



Ülesandeid

* Tee näited läbi
* Lisa kolmanda bussiliini andmed

* Koosta sarnasel moel väike lehestik matka korralduse tarbeks: osalejad, varustus, marsruut.

Grupiga lehestiku koostamine:
Leppige kokku ettevõtmine, mis kõigile kõlbulik ja soovitavalt põnev tundub
Mõelge läbi juurdekuuluv, nt. tegevused, ajakava, varustus, osalejad.
Visandage paberile lehestiku üldplaan (logo, menüü, viited ...)
Koostage ja kujundage üks leht HTMLi ja CSSiga
Jagage korduvad lõigud omaette failidesse
Looge erineva sisuga lehed, haakige korduvad tükid külge.

Andmefailid eraldi kataloogis

Kuni kümnekonna harva muudetava faili puhul on eelmine lähenemine mugav küllalt. Andmete lisamist ja eemaldamist saab aga ka mugavamaks teha. Järgnevas näites korraldati nõnda, et liini andmete näitamiseks piisab vaid sobiva nimega faili paigutamisest selle jaoks ette nähtud kataloogi. Ülejäänuga saab PHP-leht juba ise hakkama.

Lehe sisu näitamiseks tuli juurde fail liinivaataja.php. Temale antakse aadressiriba kaudu ette näidatava liini number. Liinivaataja haarab ette päise, taha jaluse ning liiniandmete kataloogist keskele sisse etteantud numbriga algava tekstifaili sisu. Eeldatakse, et failinimed on kujul 726.txt, 727.txt jne. Samuti eeldatakse, et faili nimi enne laiendit on number - sissemurdmise vastaseks kaitseks töödeldakse saabunud parameetrit `$_REQUEST["liininr"]` käsuga intval, mis muudab iga talle antud teksti arvuks. Arvud jäävad ikka pärast .txt-ga liitmist samasuguseks tekstiks. Kui aga parameetriks antakse midagi muud, siis annab intval sellele tulemuseks 0. Nõnda pole karta, et keegi võiks failinime sisse paigutada kataloogide vahel liikumise või muid andmete õngitsemiseks tarvilikke käske.

Käsklus file_get_contents tagastab parameetriga ette antud faili sisu. @-märk käsu eel annab teada, et tekkivaid veateateid ei kuvataks ekraanile - need jällegi asjad, mis kipuvad häkkeritele masina ülesehituse kohta teavet andma ning sellest tasub turvalisuse huvides hoiduda.

liinivaataja.php

```
<?php require("p2is.php"); ?>
<div id="sisu">
 <pre><?php
     if(isSet($_REQUEST["liininr"])){
        $sisu=@file_get_contents("liiniandmed/".
                      intval($_REQUEST["liininr"]).".txt");
        if($sisu){
          echo $sisu;
        } else {
          echo "Andmed puuduvad";
        }
     } else {
       echo "Liininumber puudub";
     }
   ?>
 </pre>
</div>
<?php require("jalus.php"); ?>
```

Päis endiselt samasugune

p2is.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>Bussiliinid</title>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
      <style type="text/css">
        @import "kujundus.css";
      </style>
 </head>
 <body>
 <?php require("menyy.php"); ?>
```

Menüü skanneerib läbi kõik liiniandmete kaustas leiduvad failinimed. PHP 5st alates kättesaadav funktsioon `scandir` tagastab etteantud kataloogis leiduvatest failinimedest koosneva massiivi. foreach-tsükliga võetakse sealsed nimed ükshaaval ette, käsklus explode tükeldab failinime punkti koha pealt massiiviks nii, et ühe punkti korral tekib kaheelemendiline massiiv. Kohal 0 on nimi ise ning kohal 1 on laiend. Kontrollitakse üle, et kui faililaiend ikka on .txt, siis lisatakse menüüloetellu vastava faili nimi - pannes see ka ühtlasi viitega kaasa saadetavaks parameetriks.

menyy.php

```
<div id="menyy">
  <h2>Men&uuml;&uuml;</h2>
  <ul>
   <?php
    $failinimed=scandir("liiniandmed");
    foreach($failinimed as $failinimi){
      $m=explode(".", $failinimi);
      if($m[1]=="txt"){
        echo "<li><a href='liinivaataja.php?liininr=$m[0]'>$m[0]</a></li>";
      }
    }
   ?>
  </ul>
</div>
```

kujundus.css

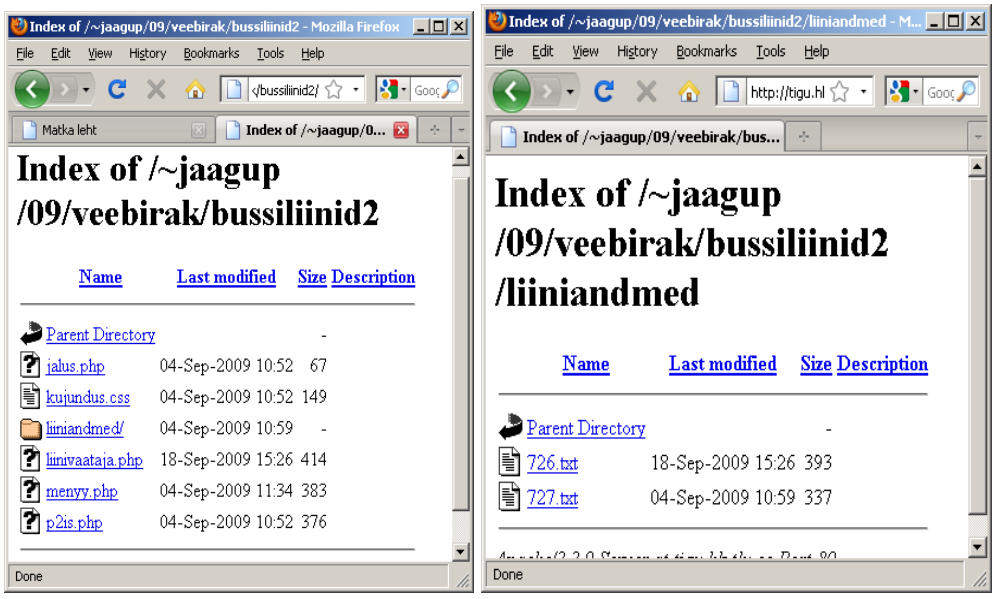
```
body{
  background-color: #ffeb90;
}

#menyy{
 float: left;
 padding-right: 30px;
}

#sisu{
 width: 70%;
 float: left;
}

#loputeade{
 clear: left;
}
```

Jalus endiselt lihtne.

jalus.php

```
    <div id="loputeade">Lehe koostas Jaagup</div>
  </body>
</html>
```

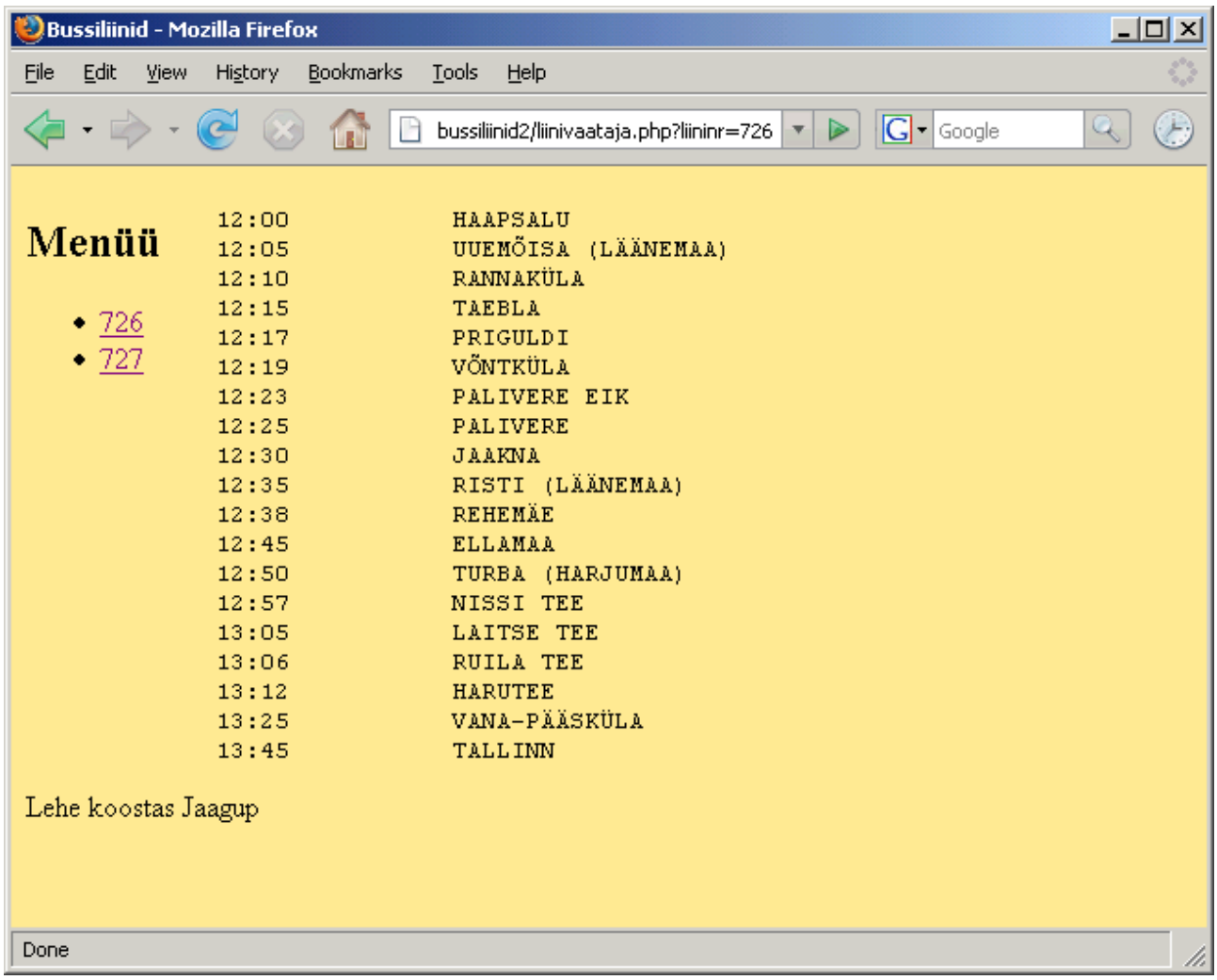
Nüüd piisab liiniandmete faili juures vaid andmetest endist, päist ja muid käsklusi pole lisada enam vaja, need tulevad liinivaataja.php juurest. Ning bussiliinide andmed kindlasti eraldi kausta nimega liiniandmed, siis leitakse need sealt üles:



liiniandmed/726.txt

```
12:00               HAAPSALU
12:05         UUEMÕISA (LÄÄNEMAA)
12:10         RANNAKÜLA
12:15         TAEBLA
12:17         PRIGULDI
12:19         VÕNTKÜLA
12:23         PALIVERE EIK
12:25         PALIVERE
12:30         JAAKNA
12:35         RISTI (LÄÄNEMAA)
12:38         REHEMÄE
12:45         ELLAMAA
12:50         TURBA (HARJUMAA)
12:57         NISSI TEE
13:05         LAITSE TEE
13:06         RUILA TEE
13:12         HARUTEE
13:25         VANA-PÄÄSKÜLA
13:45         TALLINN
```

Ja jällegi võib ilusti omale sobivad sõiduajad valida :-)



Ülesandeid

* Tee näited läbi
* Koosta sarnaselt laulusõnade lehestik: iga laul omaette lehel. Lihtsamal juhul on laulul pealkirjaks number. Keerukamal juhul on failinimi vabam, aga tasub kontrollida, et erisümboleid sisse ei jääks.
* Koosta eelmise näite abil pildigalerii. Kataloogis olevaid pilte saab veebi kaudu vaadata.
* Jaga galerii alateemadeks, ehk siis alamkataloogideks. Koosta navigeerimisvahendid lehtede vahel liikumiseks.