VEEBIRAKENDUSTE LOOMINE

MTAT.03.230 (6 EAP)
14. Loeng

Helle Hein

AJAX – II osa
Asynchronous JavaScript + XML

Teemad

- GET andmete saatmine
- Tekstivälja sisu lugemine
- POST andmete saatmine
- Ajax vahendid: tööriistad ja teegid
- Ajax raamatud

Servleti näide: disaini puudused

- HTML lehelt ei saadetud andmeid servletile
  - Lahendus: lisada andmed URL lõppu (GET andmed)
    - Kasutada GET formaati:

```
address?var1=val1&var2=val2
```

- Märkused
  - Järgmistes näidetes me võtame andmed sisendväljadelt, kuid selles paneme otse URLi
  - Serveris andmete saamiseks servletis kasutame `request.getParameter`

Etapid

- JavaScript
  - Defineerida HTTP päringute saatmiseks objekt
  - Initsialiseerida päring
    - Määrata anonüümne funktsioon vastuse käsitlemiseks
      - `onreadystatechange` atribuut
    - GET või POST päring servletile
    - URL ile GET andmed lõppu
  - Vastuse käsitlemine
    - Oodata `readyState` 4 ja HTTP staatuskood 200
    - Eraldada `responseText` või `responseXML`
    - Kasutada `innerHTML` et lisada tulemus
- HTML
  - Laadida JavaScript kesksest kaustast. Kasutada stiili.
  - Määrata element, mis algatab päringu, lisada id
  - Anda `id` väljundi kohahoidjale

Päringuobjekti defineerimine

```
function getRequestObject() {
  if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else {
    return(null);
  }
}
```

Pole muutusi

Päringu saatmine

```
function ajaxResult(address, resultRegion) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseText(request,
                                  resultRegion); };
  request.open("GET", address, true);
  request.send(null);
}
```

HTML lehelt aadress

Vastuse töötlemine

```
function showResponseText(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    htmlInsert(resultRegion, request.responseText);
  }
}

function htmlInsert(id, htmlData) {
  document.getElementById(id).innerHTML = htmlData;
}
```

Lisaks abifunktsioon

HTML kood

```
...
<fieldset>
  <legend>
    Saadame GET andmeid, tulemust näidatakse HTML
  </legend>
  <form action="#">
    <input type="button" value= "Näita Chicago aega"
           onclick='ajaxResult
                    ("show-time-in-city?city=Chicago",
                     "chicago-time")'/>
  </form>
  <div id="chicago-time"></div>
</fieldset>
…
```

Servleti kood

- Servlett loeb parameetri "`city`"
  - Kasutab seda aja määramiseks
  - Tundmatu nimi tekitab vea
- Servlett genereerib HTML
  - Tagastab teksti HTML märgistega
    - Kontrollib, kas tulemus on legaalne `xhtml`

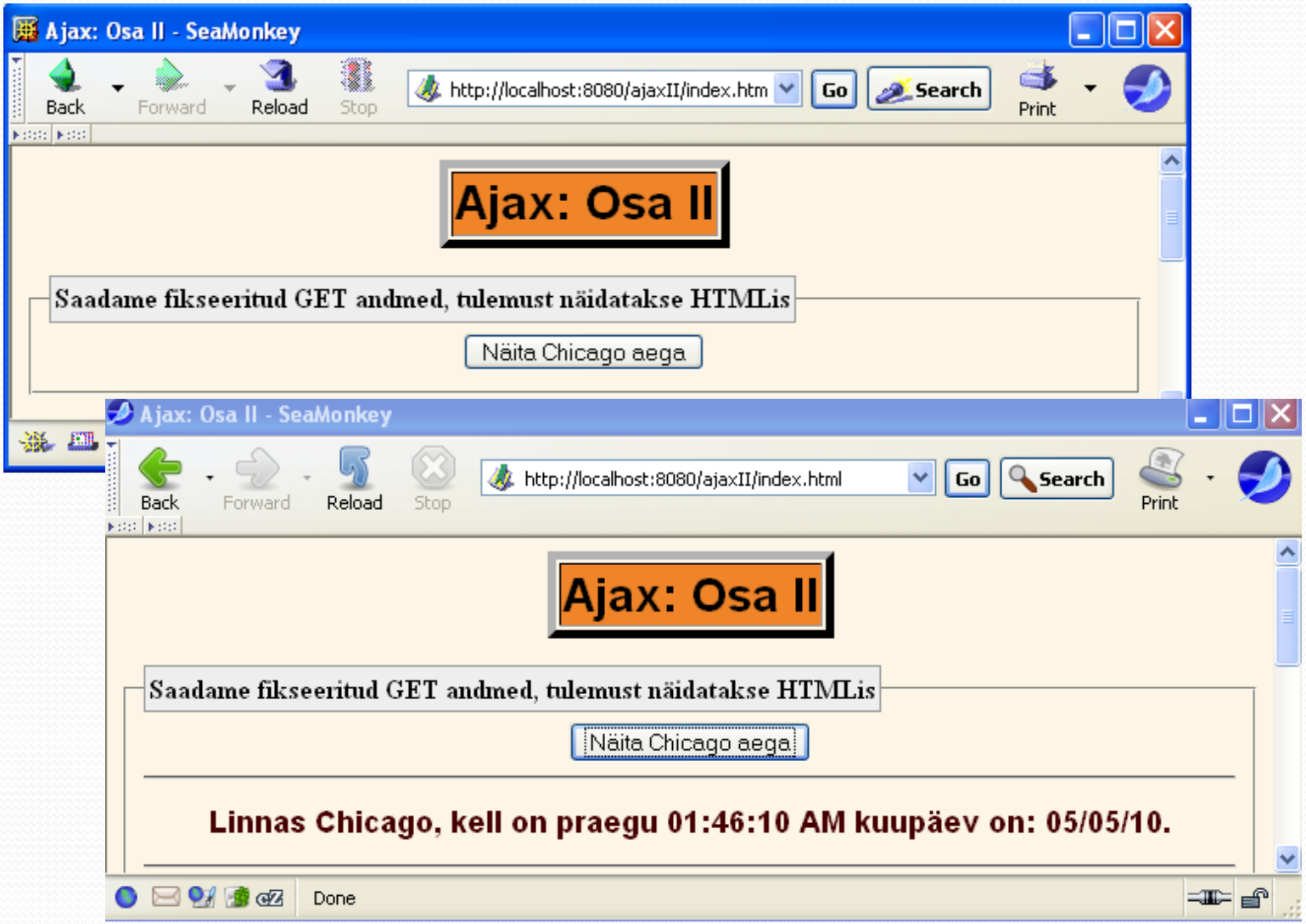
Servleti kood

```
public class ShowTimeInCity extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    PrintWriter out = response.getWriter();
    String cityName = request.getParameter("city");
    String message = CityUtils.getTime(cityName);
    if (message.startsWith("Li")) { // Found city
      message =
        String.format("<hr/><h2>%s</h2><hr/>", message);
    } else { // No city or unknown city
      message =
        String.format("<h2 class='error'>%s</h2>", message);
    }
    out.print(message);
  }
}
```

Klass CityUtils

- Hoiab tabelit linnade ja vastavate ajavöönditega (ja elanike arvuga)
  - Antud linna nime järgi leiab erinevuse tundides kohaliku ja serveri aja vahel
- Arvutab serveri aja
  - Kasutab standardklassi `GregorianCalendar`
- Teisendab aja antud linna jaoks kutsudes välja meetodi "`getTime`".
- Formaadib aja ja kuupäeva kasutades `String.format` koos `%tr` ja `%tD`

Tulemus


Ajax: Osa II - SeaMonkey
Back
Forward
Reload
Stop
http://localhost:8080/ajaxII/index.htm
Go
Search
Print
Ajax: Osa II
Saadame fikseeritud GET andmed, tulemust näidatakse HTMLis
Näita Chicago aega
Ajax: Osa II - SeaMonkey
Back
Forward
Reload
Stop
http://localhost:8080/ajaxII/index.html
Go
Search
Print
Ajax: Osa II
Saadame fikseeritud GET andmed, tulemust näidatakse HTMLis
Näita Chicago aega
Linnas Chicago, kell on praegu 01:46:10 AM kuupäev on: 05/05/10.
Done

Kasutaja sisendi lugemine
GET näide: disaini puudused

- Linna nimi alati Chicago
  - Lahendus: saata kasutaja andmed
    - Määrata `id` tekstiväljale
      - `<input type="text" id="someID"/>`
    - Lugeda andmed tekstiväljalt
      - `document.getElementById(someID).value`
- Tekstiväljad võivad sisaldada tühikuid ja teisi märke, mis on keelatud URLis
  - Lahendus: filtreerida (escape) tekstivälja andmed
    - Kasutada "`escape`" tühikute ja teiste märkide konverteerimiseks
    - Panna tulemus GET andmetesse
    - Lisada URLi GET andmed

Uued elemendid - kokkuvõte

- Lugeda tekstivälja väärtus

```
userInput = document.getElementById(someID).value;
```

- Kasutada `escape`, et teisendada erimärgid

```
userInput = escape(userInput);
```

- Panna andmed URLi

```
url = baseURL + "?someName=" + userInput;
```

- Saata päring ja töödelda tulemust
- Abifunktsioon

```
function getValue(id) {
  return(escape(document.getElementById(id).value));
}
```

Etapid

- JavaScript
  - Defineerida HTTP päringute saatmiseks objekt
  - Initsialiseerida päring
    - Määrata anonüümne funktsioon vastuse käsitlemiseks
      - `onreadystatechange` atribuut
    - GET või POST päring servletile
      - URLis on GET lõpus päringusõne
      - Andmed on saadud **`document.getElementById(id).value`** abil
  - Vastuse käsitlemine
    - Oodata `readyState` 4 ja HTTP staatuskood 200
    - Eraldada `responseText` või `responseXML`
    - Kasutada `innerHTML` , et lisada tulemus
- HTML
  - Laadida JavaScript kesksest kaustast. Kasutada stiili.
  - Määrata element, mis algatab päringu
  - Määrata **`id`** sisendelemendile
  - Defineerida tühi kohahoidja element koos `id`-ga

Päringuobjekti defineerimine

```
function getRequestObject() {
  if (window.ActiveXObject) {
    return(new
  ActiveXObject("Microsoft.XMLHTTP"));
  } else if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else {
    return(null);
  }
}
```

Pole muutusi

Päringu saatmine

```
function showTimeInCity(inputField, resultRegion) {
  var baseAddress = "show-time-in-city";
  var data = "city=" + getValue(inputField);
  var address = baseAddress + "?" + data;
  ajaxResult(address, resultRegion);
}

function ajaxResult(address, resultRegion) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseText(request,
                                  resultRegion); };
  request.open("GET", address, true);
  request.send(null);
}
```

Funktsioon `ajaxResult` on sama

Vastuse töötlemine

```
function showResponseText(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    htmlInsert(resultRegion, request.responseText);
  }
}
```

Pole muutusi

HTML kood

```
...
<fieldset>
  <legend>
    Saadame dünaamiliselt GET andmed, tulemust
  näidatakse HTMLis
  </legend>
  <form action="#">
    <label>City: <input type="text" id="city-1"/>
    </label><br/>
    <input type="button" value= "Näita linna aega"
           onclick='showTimeInCity("city-1",
                                   "city-1-time")'/>
  </form>
  <div id="city-1-time"></div>
</fieldset>
...
```

Stiilileht

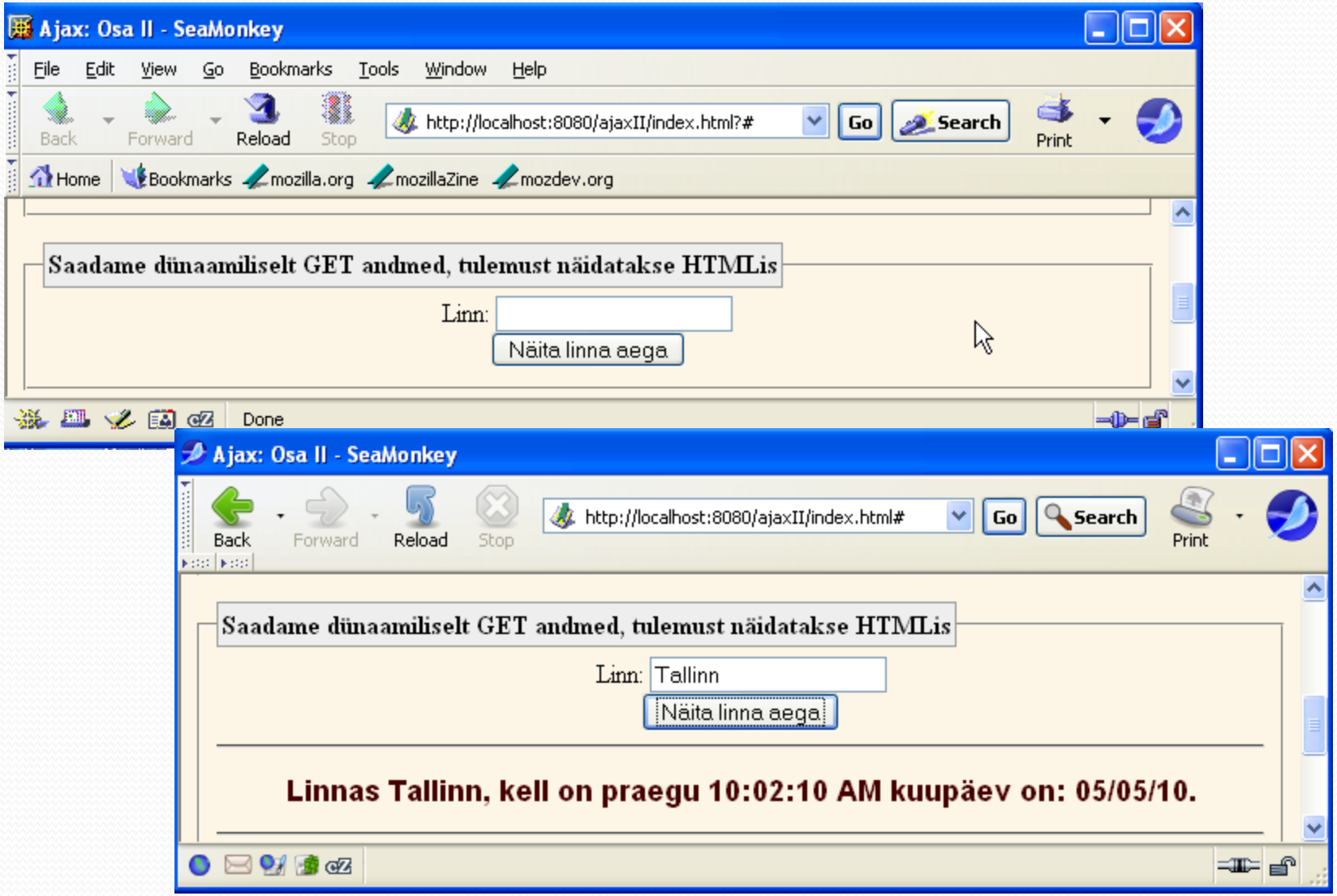
```
h2 { color: #440000;
     font-weight: bold;
     font-size: 18px;
     font-family: Arial, Helvetica, sans-serif;
}
.error { background-color: yellow;
         color: red;
         font-weight: bold;
         font-size: 20px;
         font-family: Arial, Helvetica, sans-serif;
         border-style: inset;
}
```

Servleti kood

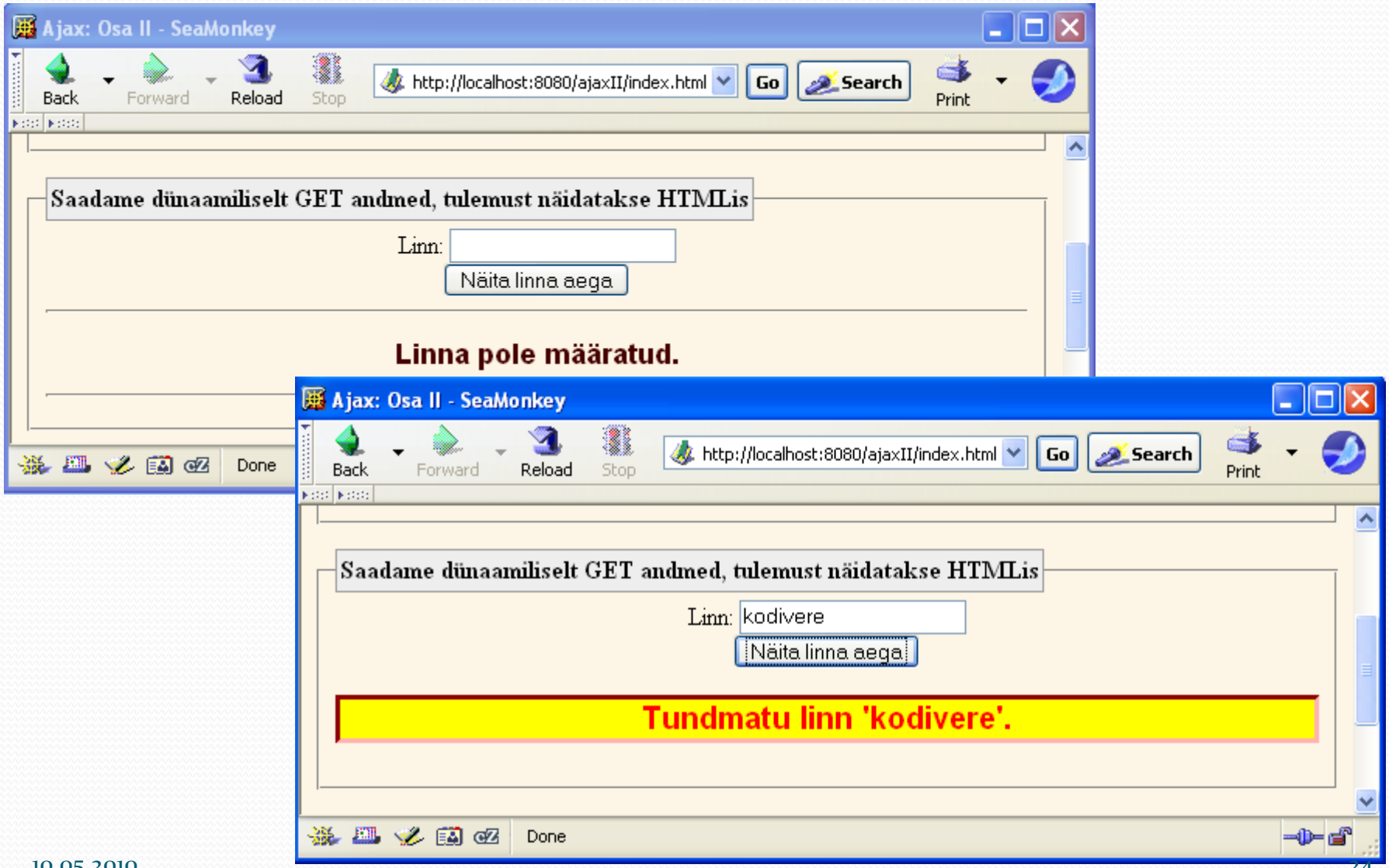
```
public class ShowTimeInCity extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    PrintWriter out = response.getWriter();
    String cityName = request.getParameter("city");
    String message = CityUtils.getTime(cityName);
    if (message.startsWith("Li")) { // Found city
      message =
        String.format("<hr/><h2>%s</h2><hr/>", message);
    } else { // No city or unknown city
      message =
        String.format("<h2 class='error'>%s</h2>", message);
    }
    out.print(message);
  }
}
```

Pole muutusi

Tulemused


Ajax: Osa II - SeaMonkey
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:8080/ajaxII/index.html?#
Go Search Print
Home Bookmarks mozilla.org mozillaZine mozdev.org
Saadame dünaamiliselt GET andmed, tulemust näidatakse HTMLis
Linn:
Näita linna aega
Done
Ajax: Osa II - SeaMonkey
Back Forward Reload Stop
http://localhost:8080/ajaxII/index.html#
Go Search Print
Saadame dünaamiliselt GET andmed, tulemust näidatakse HTMLis
Linn: Tallinn
Näita linna aega
Linnas Tallinn, kell on praegu 10:02:10 AM kuupäev on: 05/05/10.

Tulemused


Ajax: Osa II - SeaMonkey
Back
Forward
Reload
Stop
http://localhost:8080/ajaxII/index.html
Go
Search
Print
Saadame dünaamiliselt GET andmed, tulemust näidatakse HTMLis
Linn:
Näita linna aega
Linna pole määratud.
Done
Ajax: Osa II - SeaMonkey
Back
Forward
Reload
Stop
http://localhost:8080/ajaxII/index.html
Go
Search
Print
Saadame dünaamiliselt GET andmed, tulemust näidatakse HTMLis
Linn:
kodivere
Näita linna aega
Tundmatu linn 'kodivere'.
Done

Andmete saatmine POST abil

GET vs. POST

- POST või GET
  - POST
    - URL on lihtsam
    - Andmed on kaitstud nende eest, kes vaatavad üle su õla
    - Võib saata suurema koguse andmeid
    - Võib saata erimärke (nt üles laadida faile)
  - GET
    - Lehe võib järjehoidjasse panna
- Ajax korral lõppkasutajad ei näe URL, seega valik on suhteliselt vaba
  - Suure hulga andmete korral on eelistatud POST

POST abil andmete saatmine JavaScriptis

- Koguda andmed vormist
- Määrata `id` to input elements

```
<input type="text" id="some-id"/>
```

  - Lugeda andmed

```
var value1 = document.getElementById("some-id").value;
```

  - URL-kodeerida andmed ja panna päringusõnesse

```
var data = "var1=" + escape(value1);
```

- Määrata POST käsu GET asemel

```
request.open("POST", address, true);
```

- Määrata vormi sisutüüp

```
request.setRequestHeader("Content-Type“,
        "application/x-www-form-urlencoded");
```

- Lisada andmed käsku "send"

```
request.send(data);
```

Etapid

- JavaScript
  - Defineerida HTTP päringute saatmiseks objekt
  - Initsialiseerida päring
    - Määrata anonüümne funktsioon vastuse käsitlemiseks
      - `onreadystatechange` atribuut
    - POST päring servletile
    - Panna POST andmed meetodisse send
      - Andmed on saadud `document.getElementById(id).value` abil
  - Vastuse käsitlemine
    - Oodata `readyState` 4 ja HTTP staatuskood 200
    - Eraldada `responseText` või `responseXML`
    - Kasutada `innerHTML` , et lisada tulemus
- HTML
  - Laadida JavaScript kesksest kaustast. Kasutada stiili.
  - Määrata element, mis algatab päringu
  - Määrata `id` sisendelemendile
  - Defineerida tühi kohahoidja element koos `id`-ga

Päringuobjekti defineerimine

```
function getRequestObject() {
  if (window.ActiveXObject) {
    return(new
  ActiveXObject("Microsoft.XMLHTTP"));
  } else if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else {
    return(null);
  }
}
```

Pole muutusi

Päringu initsialiseerimine

```
function showTimeInCityPost(inputField, resultRegion) {
  var address = "show-time-in-city";
  var data = "city=" + getValue(inputField);
  ajaxResultPost(address, data, resultRegion);
}

function ajaxResultPost(address, data, resultRegion) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseText(request,
                                  resultRegion); };
  request.open("POST", address, true);
  request.setRequestHeader
                ("Content-Type",
                 "application/x-www-form-urlencoded");
  request.send(data);
}
```

Vastuse töötlemine

```
function showResponseText(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    htmlInsert(resultRegion, request.responseText);
  }
}
```

Pole muutusi

HTML kood

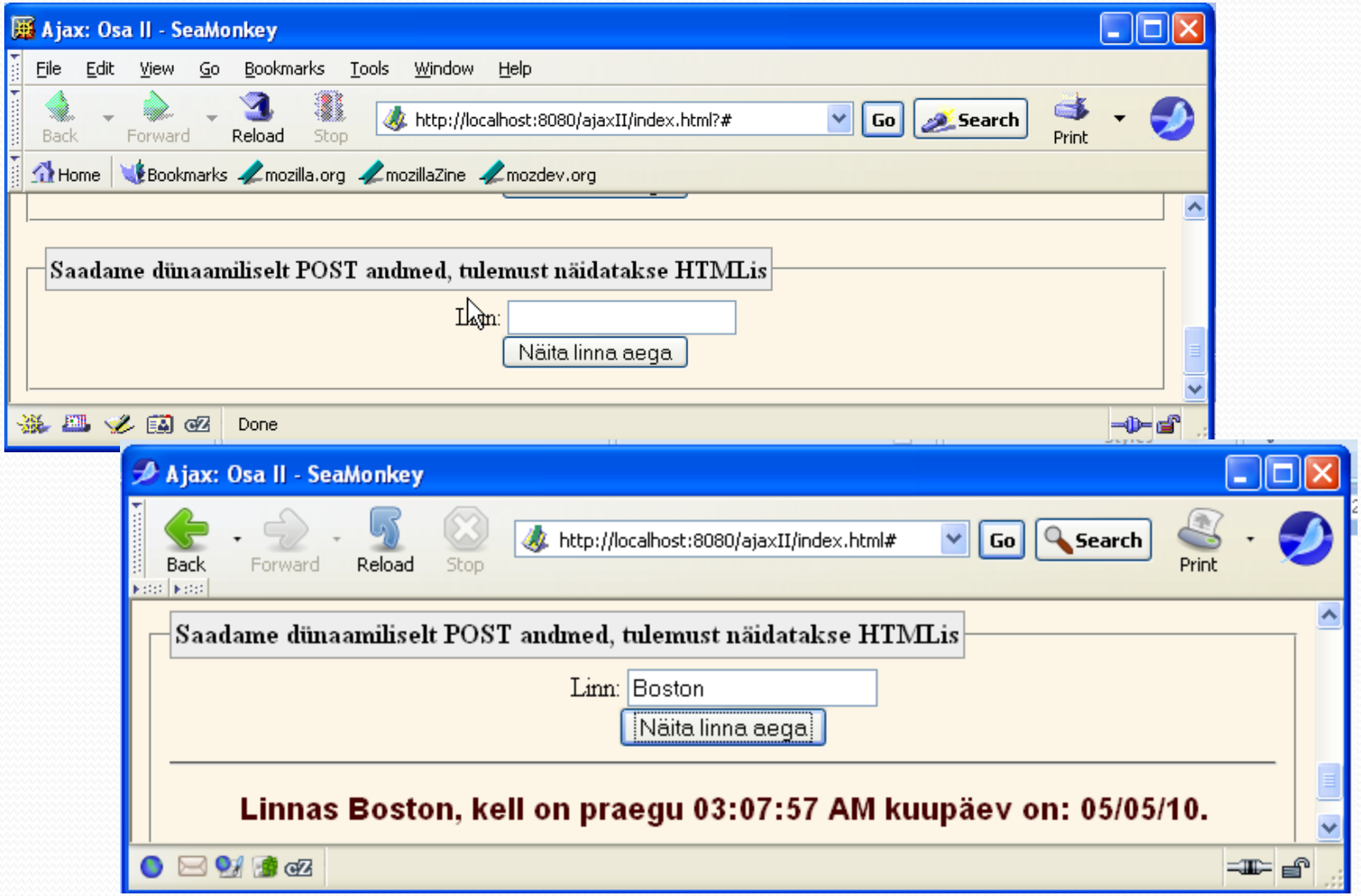
```
...
<fieldset>
  <legend>
    Saadame dünaamiliselt POST andmed, tulemust
  näidatakse HTMLis
  </legend>
  <form action="#">
    <label>City: <input type="text" id="city-2"/>
    </label><br/>
    <input type="button" value= "Näita linna aega"
           onclick='showTimeInCityPost("city-2",
                                       "city-2-time")'/>
  </form>
  <div id="city-2-time"></div>
</fieldset>
...
```

Servleti kood

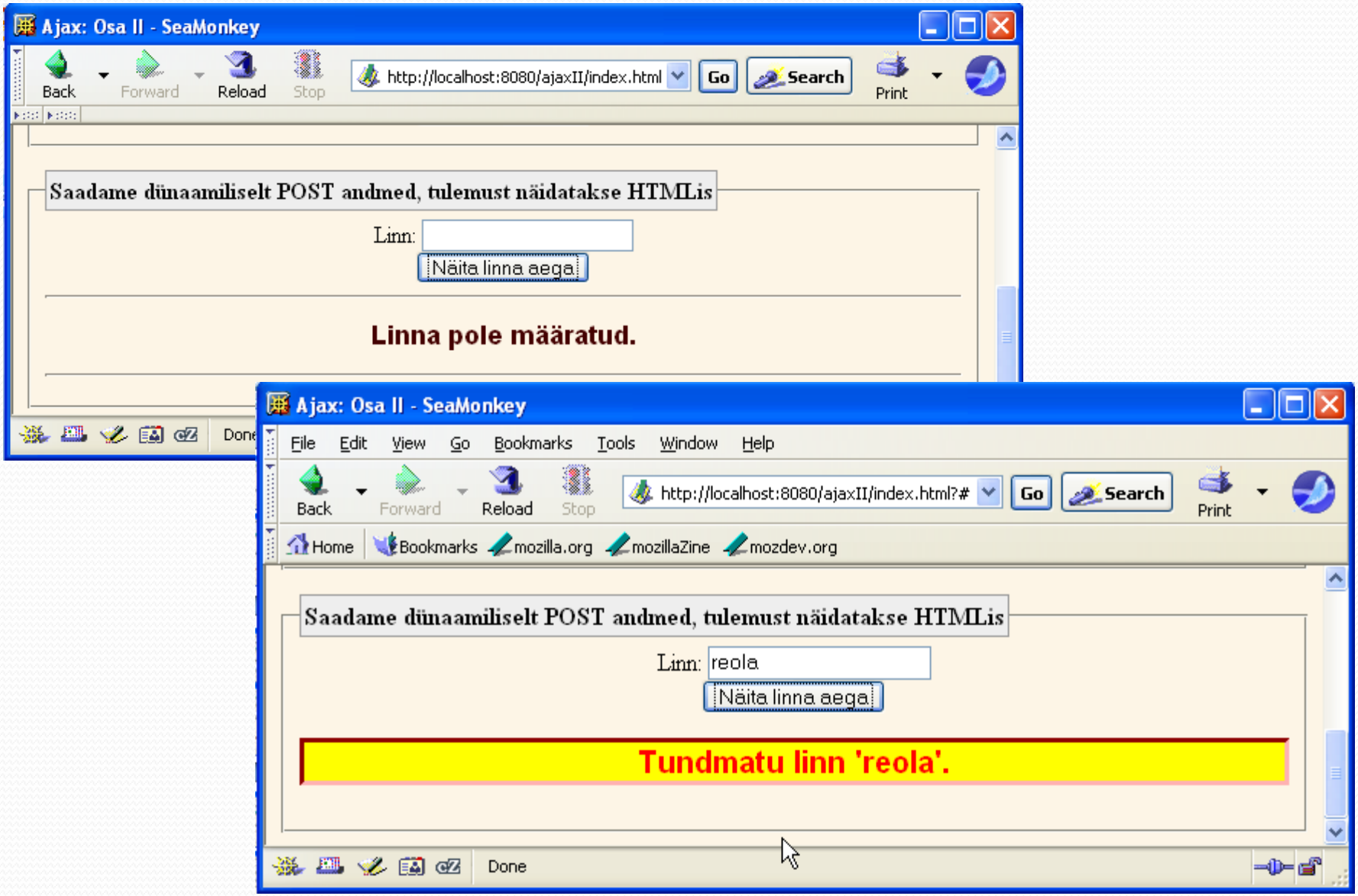
```
public class ShowTimeInCity extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    // sama, mis eelmistes näidetes
  }

  public void doPost(HttpServletRequest request,
                     HttpServletResponse response)
      throws ServletException, IOException {
    doGet(request, response);
  }
}
```

Tulemused


Ajax: Osa II - SeaMonkey
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:8080/ajaxII/index.html?#
Go Search Print
Home Bookmarks mozilla.org mozillaZine mozdev.org
Saadame dünaamiliselt POST andmed, tulemust näidatakse HTMLis
Linn:
Näita linna aega
Done
Ajax: Osa II - SeaMonkey
Back Forward Reload Stop
http://localhost:8080/ajaxII/index.html#
Go Search Print
Saadame dünaamiliselt POST andmed, tulemust näidatakse HTMLis
Linn: Boston
Näita linna aega
Linnas Boston, kell on praegu 03:07:57 AM kuupäev on: 05/05/10.

Tulemused


Ajax: Osa II - SeaMonkey
Back
Forward
Reload
Stop
http://localhost:8080/ajaxII/index.html
Go
Search
Print
Saadame dünaamiliselt POST andmed, tulemust näidatakse HTMLis
Linn:
Näita linna aega
Linna pole määratud.
Ajax: Osa II - SeaMonkey
File Edit View Go Bookmarks Tools Window Help
Back
Forward
Reload
Stop
http://localhost:8080/ajaxII/index.html?#
Go
Search
Print
Home
Bookmarks
mozilla.org
mozillaZine
mozdev.org
Saadame dünaamiliselt POST andmed, tulemust näidatakse HTMLis
Linn:
reola
Näita linna aega
Tundmatu linn 'reola'.
Done

Ajax: Erinevate andmeformaatide käsitlemine

- HTML tabelite ehitamine JavaScriptis
- XML andmed
  - Tulemuste parsimine
  - XML andmete koostamine serveril MVC abil
- JSON andmed
  - Tulemuste parsimine
  - JSON andmete koostamine serveril MVC abil
- Andmed sõne kujul (String)
  - Tulemuste parsimine
  - Sõne kujul andmete koostamine serveril MVC abil
- Kombineeritud andmed
  - Sobiva andmeesituse valimine

HTML tabelite ehitamine

- Paljudel juhtudel on serveri andmed seotud konkreetse HTML vormiga kliendi juures
  - Sellisel juhul on mõistlik serverist saata HTML märgised ja klient paneb need kohale
- Teistel juhtudel server kasutab andmeid, mis paiknevad paljudes vormides või erinevatel lehtedel
  - Andmeid võidakse kasutada erineval viisil eri rakendustes
  - Sellisel juhul on mõistlik saata andmed standardses formaadis
    - Klient peab eraldama info sellest andmeformaadist
    - Klient peab ehitama andmete põhjal HTML

HTML tabeli ehitamine - päised

```
function getTable(headings, columns) {
  var table = "<table border='1'>\n" +
              getTableHeadings(headings) +
              getTableBody(columns) +
              "</table>";
  return(table);
}
function getTableHeadings(headings) {
  var firstRow = "  <tr>";
  for(var i=0; i<headings.length; i++) {
    firstRow += "<th>" + headings[i] + "</th>";
  }
  firstRow += "</tr>\n";
  return(firstRow);
}
```

HTML tabeli ehitamine - veerud

```
function getTableBody(columns) {
  var numRows = columns[0].length;
  var numCols = columns.length;
  var body = "";
  for(var row=0; row<numRows; row++) {
    body += "  <tr>";
    for(var col=0; col<numCols; col++) {
      body += "<td>" + columns[col][row] + "</td>";
    }
    body += "</tr>\n";
  }
  return(body);
}
```

Elemendi lisamine ja eraldamine

```
// Insert the html data into the element
// that has the specified id.

function htmlInsert(id, htmlData) {
  document.getElementById(id).innerHTML = htmlData;
}

// Return escaped value of textfield that has given id.
// The built in "escape" function converts < to &lt;, etc.

function getValue(id) {
  return(escape(document.getElementById(id).value));
}
```

Näide (JavaScript)

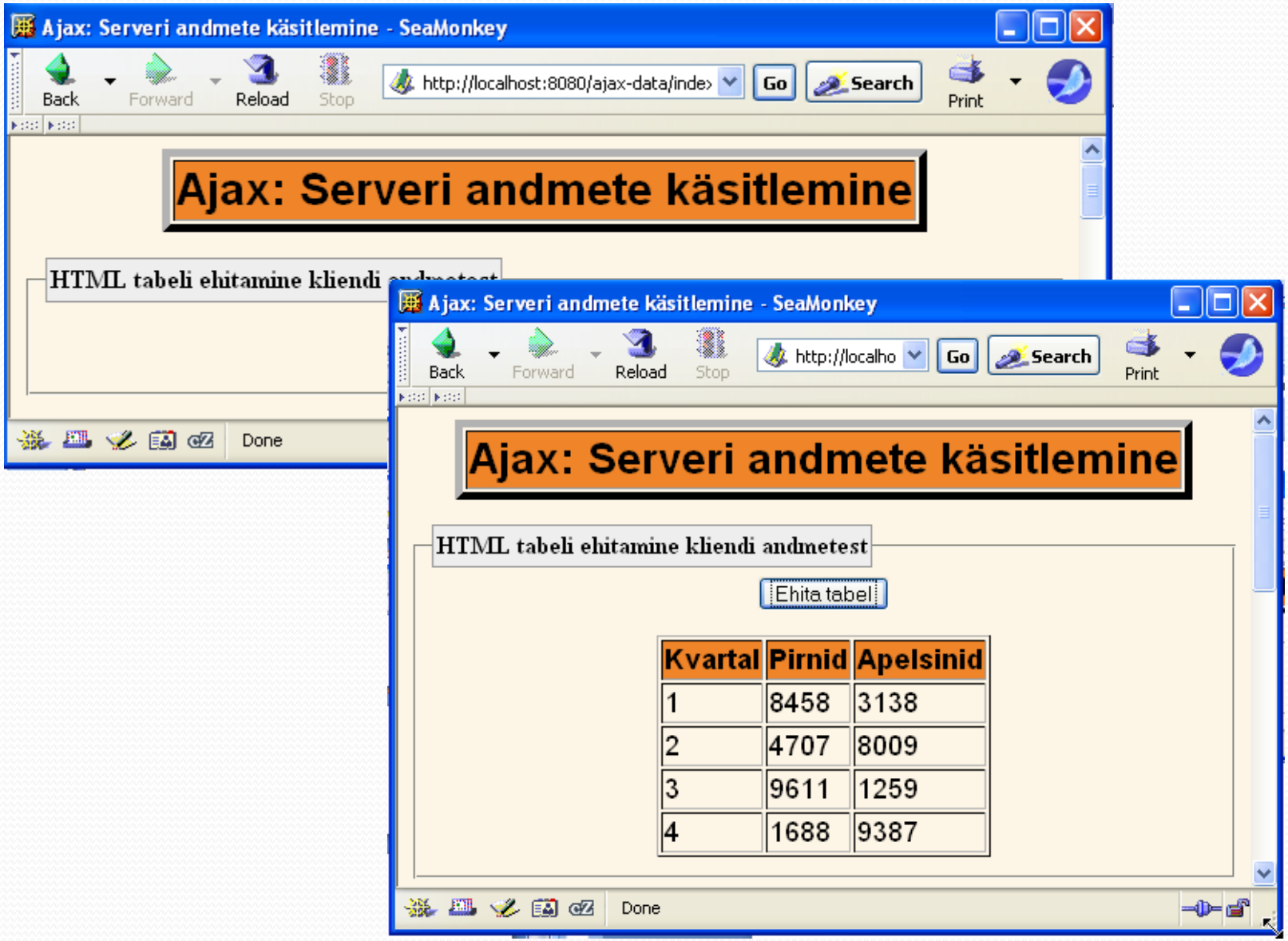
```
function clientTable(displayRegion) {
  var headings = ["Kvartal", "Pirnid", "Apelsinid"];
  var rows = [["1", randomSales(), randomSales()],
              ["2", randomSales(), randomSales()],
              ["3", randomSales(), randomSales()],
              ["4", randomSales(), randomSales()]];
  var table = getTable(headings, rows);
  htmlInsert(displayRegion, table);
}

function randomSales() {
  var sales = 1000 + (Math.round(Math.random() * 9000));
  return("$" + sales);
}
```

HTML kood

```
...
<fieldset>
  <legend>HTML tabeli ehitamine kliendi andmetest
  </legend>
  <form action="#">
   <input type="button" value="Ehita tabel"
          onclick='clientTable("client-table")'/>
  </form>
  <p/>
  <div id="client-table"></div>
</fieldset>
...
```

Tulemus


Ajax: Serveri andmete käsitlemine - SeaMonkey
Back
Forward
Reload
Stop
http://localhost:8080/ajax-data/inde
Go
Search
Print
Ajax: Serveri andmete käsitlemine
HTML tabeli ehitamine kliendi andmetest
Done
Ajax: Serveri andmete käsitlemine - SeaMonkey
Back
Forward
Reload
Stop
http://localho
Go
Search
Print
Ajax: Serveri andmete käsitlemine
HTML tabeli ehitamine kliendi andmetest
Ehita tabel
Kvartal
Pirnid
Apelsinid
1
8458
3138
2
4707
8009
3
9611
1259
4
1688
9387
Done

XML andmete käsitlemine

- XML andmedokumendi võtmine

```
var xmlDocument = request.responseXML;
```

- XML elementide massiivi leidmine

```
xmlDocument.getElementsByTagName(xmlElementName);
```

- Nt kui XML on järgmine

```
<a><b>ford</b>
   <c>bar</c>
   <b>vaz</b>
</a>
```

Siis

```
getElementsByTagName("b")
```

tagastab kaheelemendilise massiivi koos märgistega `b` ja nende sisuga

XML andmete käsitlemine

- Tekst algus- ja lõpumärgiste vahelt

```
someElement.childNodes[0].nodeValue
```

Nt kui XML on

```
<a><b>ford</b>
  <c>bar</c>
  <b>vaz</b>
</a>
```

Järgneva koodi tulemuseks `"ford"`:

```
var elementArray =
     xmlDocument.getElementsByTagName("b");
var value = elementArray[0].childNodes[0].nodeValue;
```

XML abifunktsioon

```
function getXmlValues(xmlDocument, xmlElementName) {
  var elementArray =
    xmlDocument.getElementsByTagName(xmlElementName);
  var valueArray = new Array();
  for(var i=0; i<elementArray.length; i++) {
    valueArray[i] =
      elementArray[i].childNodes[0].nodeValue;
  }
  return(valueArray);
}
```

Näiteks `<foo><a>one</a><q>two</q><a>three</a></foo>`

`getXmlValues(doc, "a")` tagastab `["one", "three"]`.

Päringu saatmine

// Siin edastame vastust käsitleva funktsiooni

```
function ajaxPost(address, data, responseHandler) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { responseHandler(request); };
  request.open("POST", address, true);
  request.setRequestHeader("Content-Type",
      "application/x-www-form-urlencoded");
  request.send(data);
}
```

Etapid

- JavaScript
  - Defineerida HTTP päringute saatmiseks objekt
  - Initsialiseerida päring
    - Määrata anonüümne funktsioon vastuse käsitlemiseks
      - `onreadystatechange` atribuut
    - POST päring servletile
    - Panna POST andmed meetodisse `send`
      - Andmed on saadud `document.getElementById(id).value abil`
  - Vastuse käsitlemine
    - Oodata `readyState` 4 ja HTTP staatuskood 200
    - Eraldada andmed **`responseXML`** abil
    - Eraldada tekst XML ist kasutades

      `getElementsByTagName` ja

      `childNodes[0].nodeValue`

    - Ehitada HTML tabel või muu HTML sisu
- HTML
  - Laadida JavaScript kesksest kaustast. Kasutada stiili.
  - Määrata element, mis algatab päringu
  - Määrata `id` sisendelemendile
  - Defineerida tühi kohahoidja element koos `id`-ga

Päringu initsialiseerimine

```
function xmlCityTable(inputField, resultRegion) {
  var address = "show-cities";
  var data = "cityType=" + getValue(inputField) +
             "&format=xml";
  ajaxPost(address, data,
           function(request) {
             showXmlCityInfo(request, resultRegion);
           });
}
```

Vastuse käsitlemine

```
function showXmlCityInfo(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    var xmlDocument = request.responseXML;
    var headings = ["Linn", "Aeg", "Elanikke"];
    var columns = [getXmlValues(xmlDocument, "name"),
                   getXmlValues(xmlDocument, "time"),
                   getXmlValues(xmlDocument, "population")];
    var table = getTable(headings, columns);
    htmlInsert(resultRegion, table);
  }
}
```

HTML kood

```
...
<fieldset>
  <legend>XML andmed serverist, HTML tabeli ehitamine</legend>
  <form action="#">
   <label for="city-type-1">Linna tüüp:</label>
   <select id="city-type-1">
     <option value="top-5-cities">Eesti viis suurimat
  linna</option>
     <option value="second-5-cities">Järgmised viis Eesti
  suurimat linna</option>
     <option value="cities-starting-with-P">
       Eesti linnad, mis algavad P tähega</option>
    </select>
   <br/>
   <input type="button" value="Show Cities"
          onclick='xmlCityTable("city-type-1", "xml-city-
  table")'/>
  </form>
  <p/>
  <div id="xml-city-table"></div>
</fieldset>
...
```

Serveri disain: MVC

- Loogika
  - Seada päised, lugeda päringu parameetrid, arvutada tulemus (servletis)
- Esitlus
  - Ehitada XML fail (JSPs – kasutada EL tulemuste kättesaamiseks)
- Väikesed kõrvalekalded tavalisest MVC
  - Võib seada `Content-Type` servletis, kasutada `RequestDispatcher.include` mitte `RequestDispatcher.forward`

Servleti kood

```
public class ShowCities extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    String cityType = request.getParameter("cityType");
    List<City> cities = CityUtils.findCities(cityType);
    request.setAttribute("cities", cities);
    String format = request.getParameter("format");
    String outputPage;
    if ("xml".equals(format)) {
      response.setContentType("text/xml");
      outputPage = "/WEB-INF/results/cities-xml.jsp";
    } ...
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(outputPage);
    dispatcher.include(request, response);
  }
```

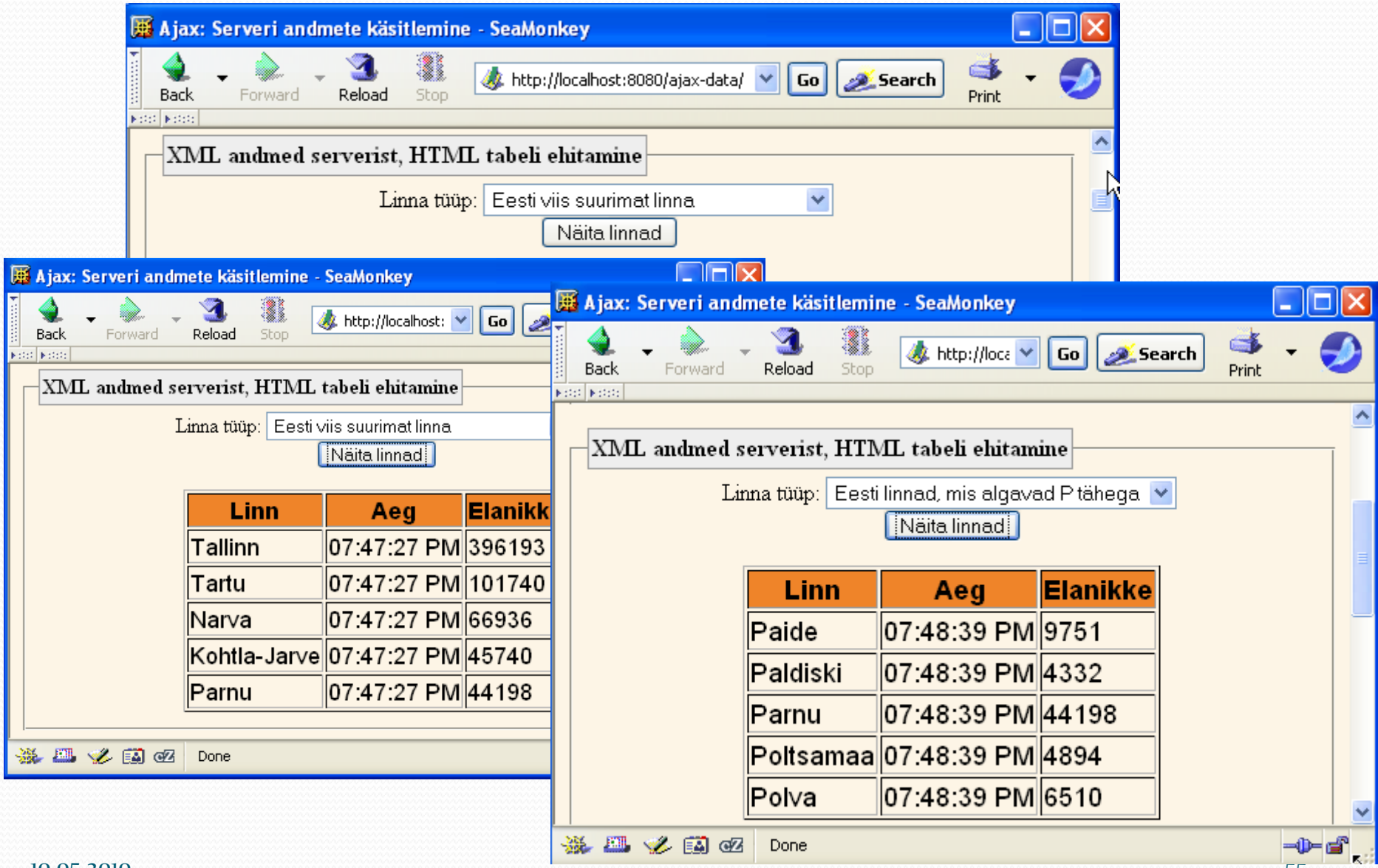
Servleti kood

```
public void doPost(HttpServletRequest request,
                   HttpServletResponse response)
    throws ServletException, IOException {
  doGet(request, response);
}
```

JSP kood(/WEB-INF/results/cities-xml.jsp)

```
<?xml version="1.0" encoding="UTF-8"?>
<cities>
  <city>
    <name>${cities[0].name}</name>
    <time>${cities[0].shortTime}</time>
    <population>${cities[0].population}</population>
  </city>
  <city>
    <name>${cities[1].name}</name>
    <time>${cities[1].shortTime}</time>
    <population>${cities[1].population}</population>
  </city>
  ...
  <city>
    <name>${cities[4].name}</name>
    <time>${cities[4].shortTime}</time>
    <population>${cities[4].population}</population>
  </city>
</cities>
```

XML andmed


Ajax: Serveri andmete käsitlemine - SeaMonkey
Back
Forward
Reload
Stop
http://localhost:8080/ajax-data/
Go
Search
Print
XML andmed serverist, HTML tabeli ehitamine
Linna tüüp:
Eesti viis suurimat linna
Näita linnad
Linn
Aeg
Elanikk
Tallinn 07:47:27 PM 396193
Tartu 07:47:27 PM 101740
Narva 07:47:27 PM 66936
Kohtla-Jarve 07:47:27 PM 45740
Parnu 07:47:27 PM 44198
Done
Eesti linnad, mis algavad P tähega
Linn
Aeg
Elanikke
Paide 07:48:39 PM 9751
Paldiski 07:48:39 PM 4332
Parnu 07:48:39 PM 44198
Poltsamaa 07:48:39 PM 4894
Polva 07:48:39 PM 6510

JSON andmete käsitlemine

JSON - JavaScript Object Notation

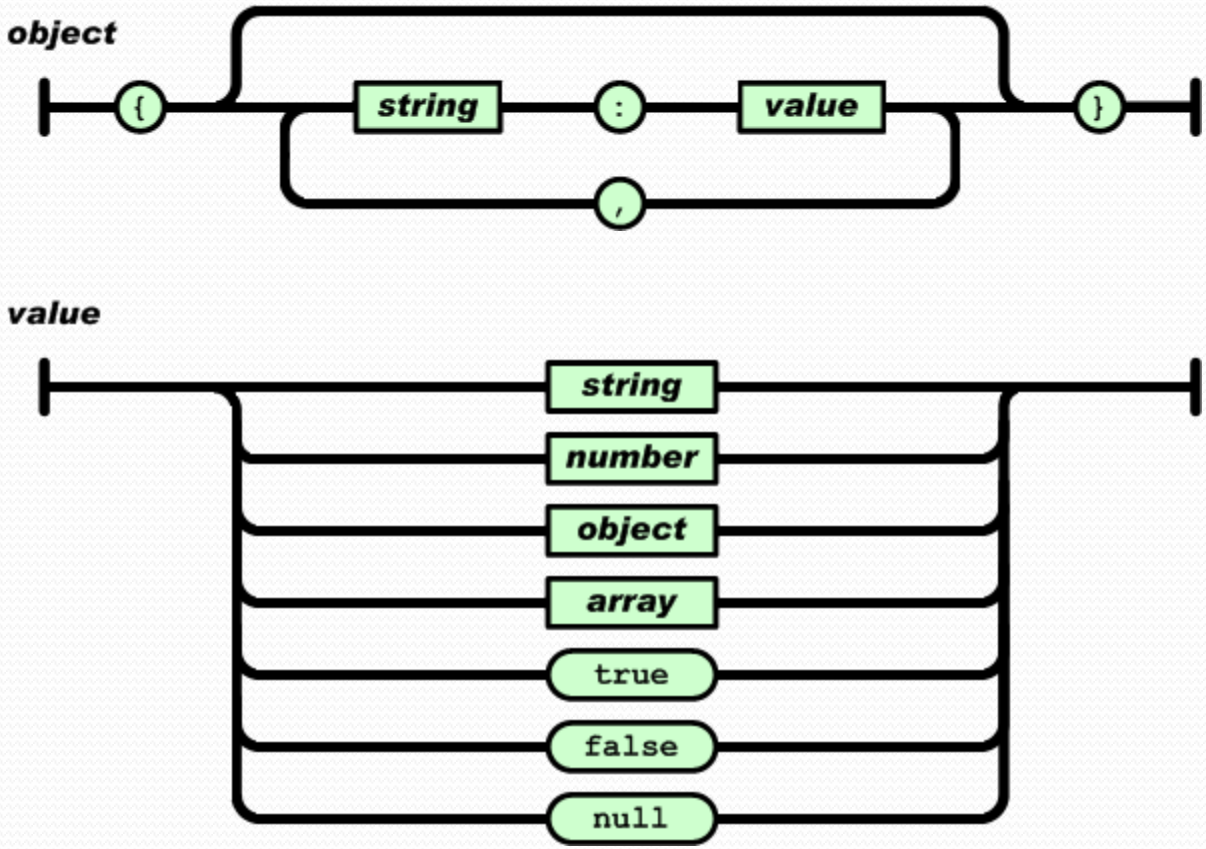
- Formaat on spetsifitseeritud RFC 4627 Douglas Crockfordi poolt
- JSON meedia tüüp on `application/json`
- Faili laiendiks `.json`
- JSON formaati kasutatakse struktureeritud andmete transportimiseks üle võrgu.
- Põhiliseks rakenduseks on AJAX veebiprogrammeerimine, kus ta on alternatiiviks formaadile XML.

- JSON koosneb kahest põhistruktuurist:

- Kollektsioon `nimi/väärtus` paaridest. Eri keeltes on see realiseeritud kas `object`, `record`, `struct`, `dictionary`, `hash table`, `list` või `array` (assotsiatiivne).

- Järjestatud väärtuste hulk. Eri keeltes on see realiseeritud kas `array`, `vector`, `list` või `sequence`.

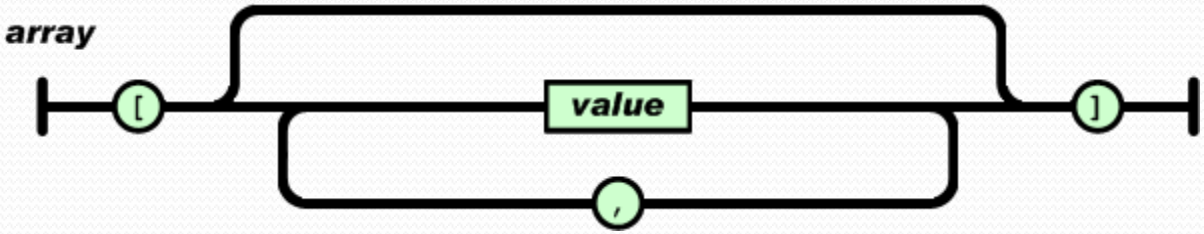
JSON süntaks

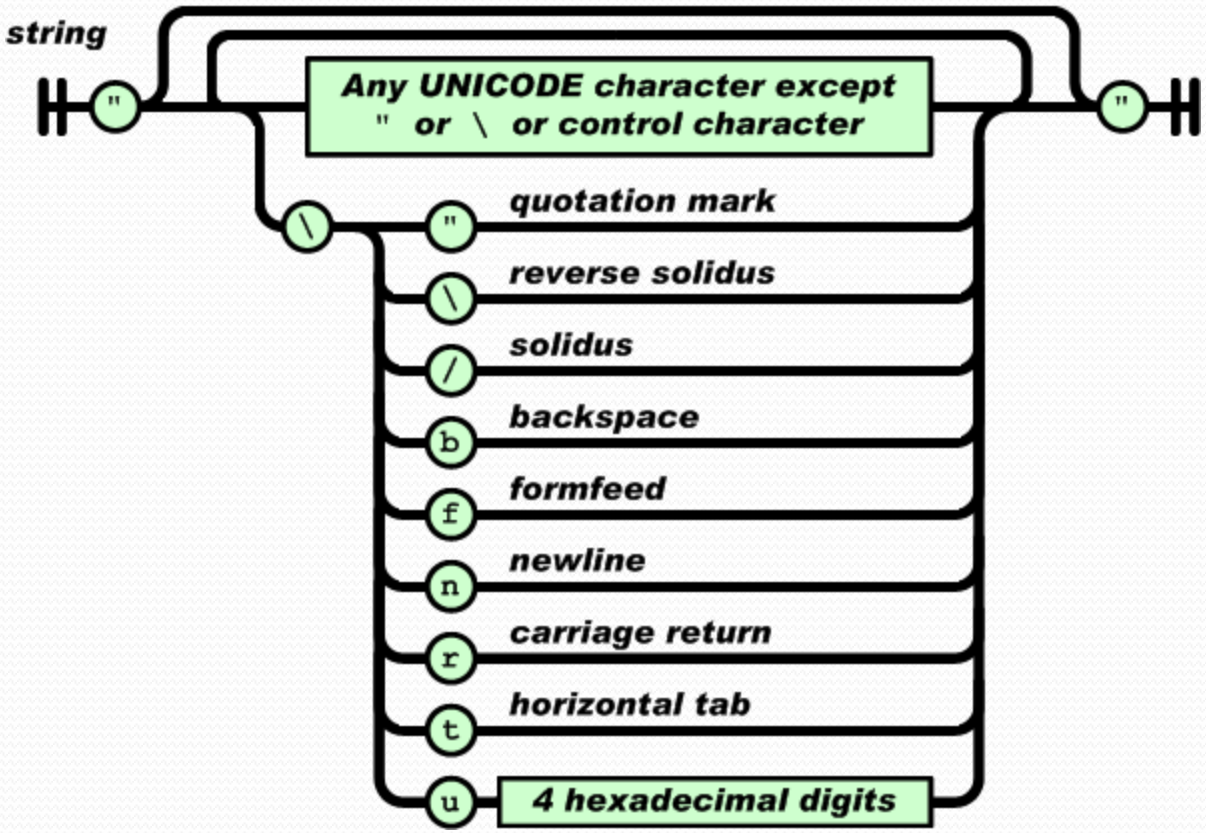


http://json.org/

```
var myFirstJSON = { "firstName" : "John",
                    "lastName"  : "Doe",
                    "age"       : 23
                    };

document.writeln(myFirstJSON.firstName);  // väljund John
document.writeln(myFirstJSON.lastName);   // väljund Doe
document.writeln(myFirstJSON.age);        // väljund 23
```





http://json.org/

```
var myJSONObject = {"bindings": [

{"ircEvent": "PRIVMSG", "method": "newURI",
                        "regex": "^http://.*"},

{"ircEvent": "PRIVMSG", "method": "deleteURI",
                        "regex": "^delete.*"},

{"ircEvent": "PRIVMSG", "method": "randomURI",
                        "regex": "^random.*"}
]
};

  myJSONObject.bindings[0].method // "newURI"
```

Objektiks teisendamine:

```
var myObject = eval('(' + myJSONtext + ')');
```

```
var employees = { "accounting" : [
                                  { "firstName" : "John",
                                    "lastName"  : "Doe",
                                    "age"       : 23
                                  },

                                  { "firstName" : "Mary",
                                    "lastName"  : "Smith",
                                    "age"       : 32 }
                                  ],

                  "sales"       : [
                                   { "firstName" : "Sally",
                                     "lastName"  : "Green",
                                     "age"       : 27
                                  },
                                   { "firstName" : "Jim",
                                     "lastName"  : "Galley",
                                     "age"       : 41
                                  }
                                  ]

                }
```

JSON andmete käsitlemine

- Põhiobjekt
  - Ümbritseda loogiliste sulgudega
  - Väljade nimed jutumärkidesse
  - Koolon välja nime ja väärtuse vahel
  - Komad peale `nimi: väärtus` paari.
- Välja väärtus
  - Sõned: kasuta ‘ või “
  - Arvud: pole vaja
  - Massiivid: komaga eraldatud väärtused kandilistes sulgudes [ ]
- JSON panek sõnesse
  - Ümbritseda sulgude ja ‘ “
  - Edastada tulemus "`eval`" et saada tagasi objekti

Näide

```
var firstObject =
  { field1: "string-value1",
    field2: 3,
    field3: ["a", "b", "c"]
   };
var someString =
  '({ f1: "val1", f2: "val2" })';
var secondObject = eval(someString);
```

- Tulemus
  - `firstObject.field1` → `"string-value1"`
  - `firstObject.field2` → `3`
  - `firstObject.field3[1]` → `"b"`
  - `secondObject.f1` → `"val1"`
  - `secondObject.f2` → `"val2"`

JSON näide-etapid

- JavaScript
  - Defineerida HTTP päringute saatmiseks objekt
  - Initsialiseerida päring
    - Määrata anonüümne funktsioon vastuse käsitlemiseks
      - `onreadystatechange` atribuut
    - POST päring servletile
    - Panna POST andmed meetodisse `send`
  - Vastuse käsitlemine
    - Oodata `readyState` 4 ja HTTP staatuskood 200
    - Eraldada andmed **`responseXML`** abil
      - Edastada sõne "eval“ JavaScript objekti saamiseks
      - Väljadele, massiivi elementidele viitamiseks kasutada JavaScript süntaksit
    - Kasutada `innerHTML` et sisestada elemendi väärtus
- HTML
  - Laadida JavaScript kesksest kaustast. Kasutada stiili.
  - Määrata element, mis algatab päringu
  - Määrata `id` sisendelemendile
  - Defineerida tühi kohahoidja element koos `id`-ga

Päringu initsialiseerimine

```
function jsonCityTable(inputField, resultRegion) {
  var address = "show-cities";
  var data = "cityType=" + getValue(inputField) +
             "&format=json";
  ajaxPost(address, data,
           function(request) {
             showJsonCityInfo(request,
  resultRegion);
           });
}
```

Vastuse käsitlemine

```
function showJsonCityInfo(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    var rawData = request.responseText;
    var data = eval(rawData);
    var headings = ["Linn", "Aeg", "Elanikke"];
    var columns = [data.names,
                   data.times,
                   data.populations];
    var table = getTable(headings, columns);
    htmlInsert(resultRegion, table);
  }
}
```

HTML kood

```
...
<fieldset>
  <legend>JSON andmed serverilt, HTML tabeli ehitamine
  </legend>
  <form action="#">
   <label for="city-type-2">Linna tüüp:</label>
   <select id="city-type-2">
     <option value="top-5-cities">Eesti viis suurimat
        linna</option>
     <option value="second-5-cities">Järgmised viis Eesti
        suurimat linna</option>
     <option value="cities-starting-with-P">
       Eesti linnad, mis algavad P tähega</option>
   </select>
   <br/>
   <input type="button" value="Näita linnad"
          onclick='jsonCityTable("city-type-2",
                                 "json-city-table")'/>
  </form>
  <p/>
  <div id="json-city-table"></div>
</fieldset>...
```

Serveri disaign: MVC

- Loogika
  - Ainus muudatus selles, milline leht valida
- Esitlus
  - Ehitada tekstikujul leht XML lehe asemel
  - Andmed JSON formaati
    - Panna sulud andmete ümber

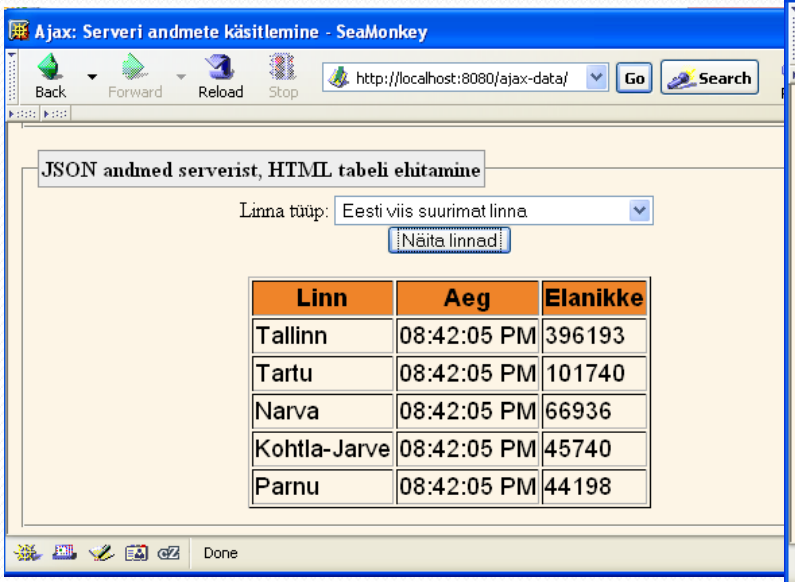
Servleti kood

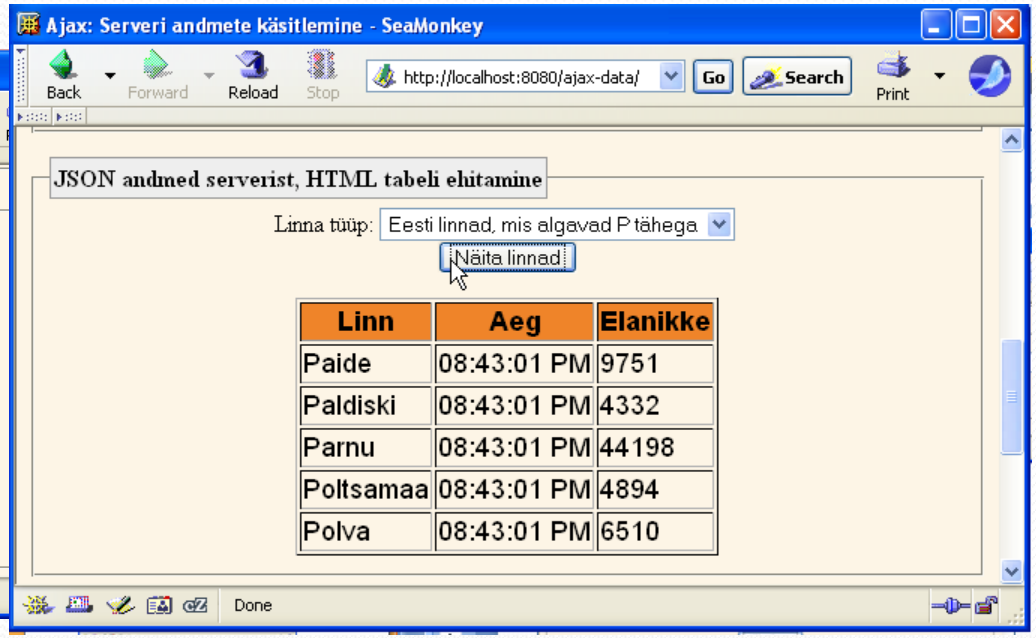
```
public class ShowCities extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    ...
    request.setAttribute("cities", cities);
    String format = request.getParameter("format");
    String outputPage;
    if ("xml".equals(format)) {
      response.setContentType("text/xml");
      outputPage = "/WEB-INF/results/cities-xml.jsp";
    } else if ("json".equals(format)) {
      response.setContentType("text/javascript");
      outputPage = "/WEB-INF/results/cities-json.jsp";
    } ...
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(outputPage);
    dispatcher.include(request, response);
  }
```

JSP kood (/WEB-INF/results/cities-json.jsp)

```
({ names: ["${cities[0].name}",
           "${cities[1].name}",
           "${cities[2].name}",
           "${cities[3].name}",
           "${cities[4].name}"],
   times: ["${cities[0].shortTime}",
           "${cities[1].shortTime}",
           "${cities[2].shortTime}",
           "${cities[3].shortTime}",
            "${cities[4].shortTime}"],
  populations: ["${cities[0].population}",
                "${cities[1].population}",
                "${cities[2].population}",
                "${cities[3].population}",
                "${cities[4].population}"]
})
```

JSON tulemused

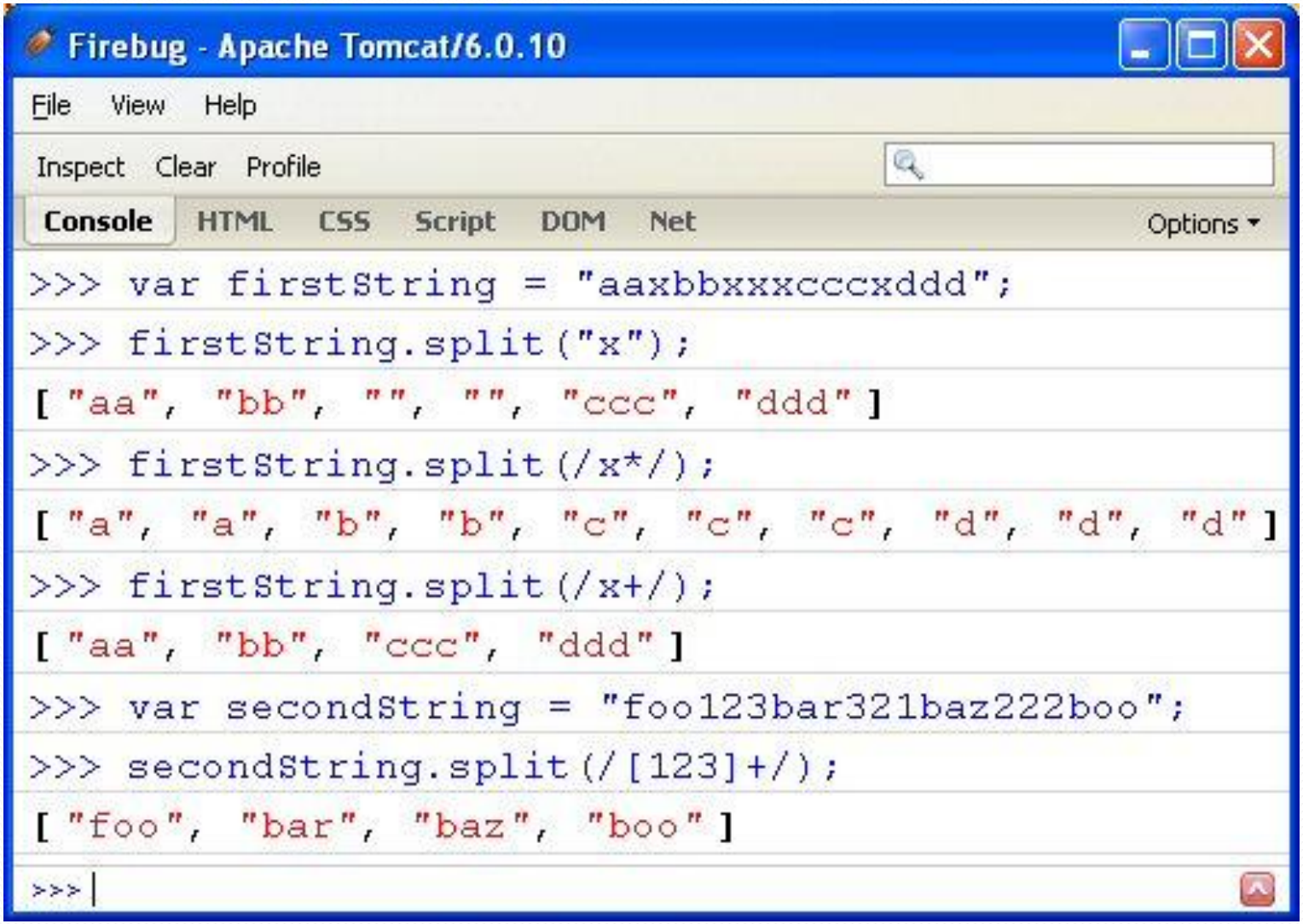




Sõnekujul andmete töötlemine

- Üldine lähenemine
  - Serveripoolne kood tekitab andmed mingis formaadis
  - Kliendipoolne kood töötleb seda
- Spetsiifiline lähenemine
  - Serveripoolne kood saadab sõned, milles on eraldajad
  - Kliendipoolne kood kasutab **`String.split,`** et eraldada osad massiivi
- `String.split` JavaScriptis
  - Sarnane `String.split` Java
  - Ühemärgilise eraldaja korral kasutada ‘ või “
  - Regulaaravaldiste korral kasutada /
- Viited
  - http://www.evolt.org/article/Regular_Expressions_in_JavaScript/17/36435/
  - http://www.javascriptkit.com/javatutors/re.shtml

`String.split`: Näide



Sõnekujul andmete töötlemine: Näide

Steps

- JavaScript
  - Defineerida objekt HTTP päringu saatmiseks
  - Initsialiseerida päring
  - Töödelda vastust
    - Eraldada vastuse tekst
      - Moodustada massiiv `String.split`
      - Töödelda massiivi elemente
    - Panna tulemus soovitud kohale
- HTML
  - Laadida JavaScript
  - Määrata element, mis algatab päringu
  - Anda id elemendile
  - Määrata tühi kohahoidja ja selle id

Päringu initsialiseerimine

```
function stringCityTable(inputField, resultRegion) {
  var address = "show-cities";
  var data = "cityType=" + getValue(inputField) +
             "&format=string";
  ajaxPost(address, data,
           function(request) {
             showStringCityInfo(request, resultRegion);
           });
}
```

Vastuse töötlemine

```
function showStringCityInfo(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    var rawData = request.responseText;
    var columnStrings = rawData.split(/\n/);
    var names = columnStrings[0].split("#");
    var times = columnStrings[1].split("#");
    var populations = columnStrings[2].split("#");
    var headings = ["City", "Time", "Population"];
    var columns = [names, times, populations];
    var table = getTable(headings, columns);
    htmlInsert(resultRegion, table);
  }
}
```

HTML kood

```
...
<fieldset>
  <legend>Getting String Data from Server, Building HTML Table
  </legend>
  <form action="#">
   <label for="city-type-3">Linna tüüp:</label>
   <select id="city-type-3">
     <option value="top-5-cities">Eesti viis suurimat linna</option>
     <option value="second-5-cities">Järgmised viis Eesti suurimat
       linna</option>
     <option value="cities-starting-with-s">
       Eesti linnad, mis algavad P tähega</option>
   </select>
   <br/>
   <input type="button" value="Näita linnad"
          onclick='stringCityTable("city-type-3",
                                   "string-city-table")'/>
  </form>
  <p/>
  <div id="string-city-table"></div>
</fieldset>...
```

Serveri disain: MVC

- Loogika
  - Muutusi ei ole
  - Otsustamine, milline JSP leht
- Esitlus
  - Ehitada plain/text leht
  - Võtta andmed eraldajate vahelt

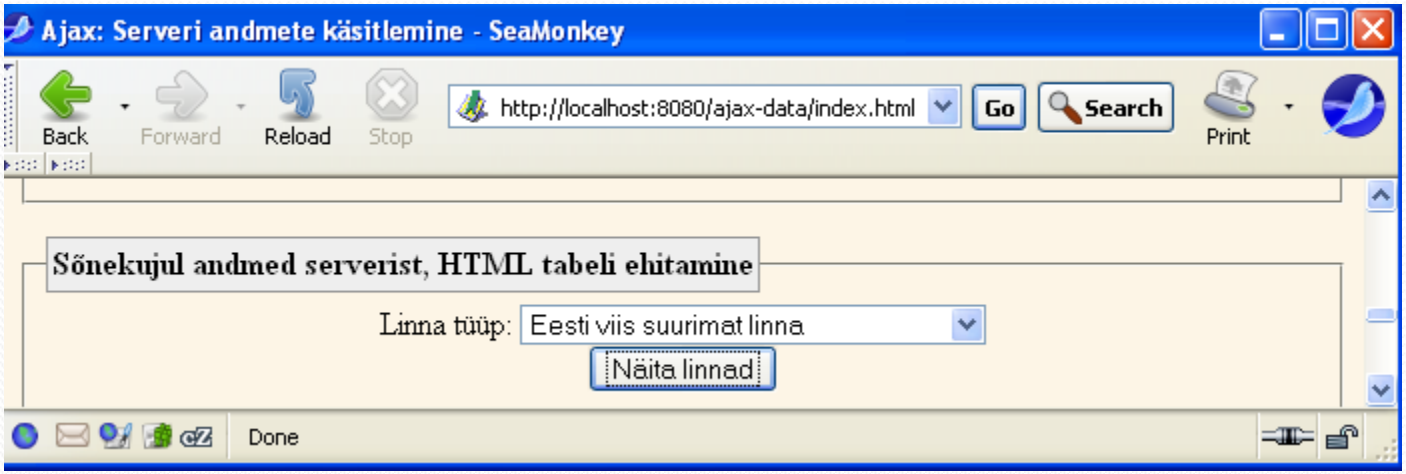
Servleti kood

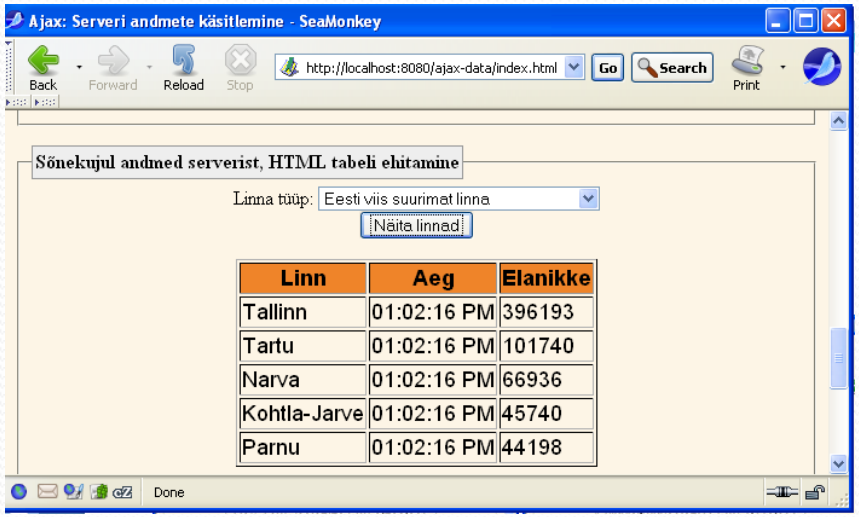
```
public class ShowCities extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    ...
    if ("xml".equals(format)) {
      response.setContentType("text/xml");
      outputPage = "/WEB-INF/results/cities-xml.jsp";
    } else if ("json".equals(format)) {
      response.setContentType("text/javascript");
      outputPage = "/WEB-INF/results/cities-json.jsp";
    } else {
      response.setContentType("text/plain");
      outputPage = "/WEB-INF/results/cities-string.jsp";
    }
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(outputPage);
    dispatcher.include(request, response);
  }
```

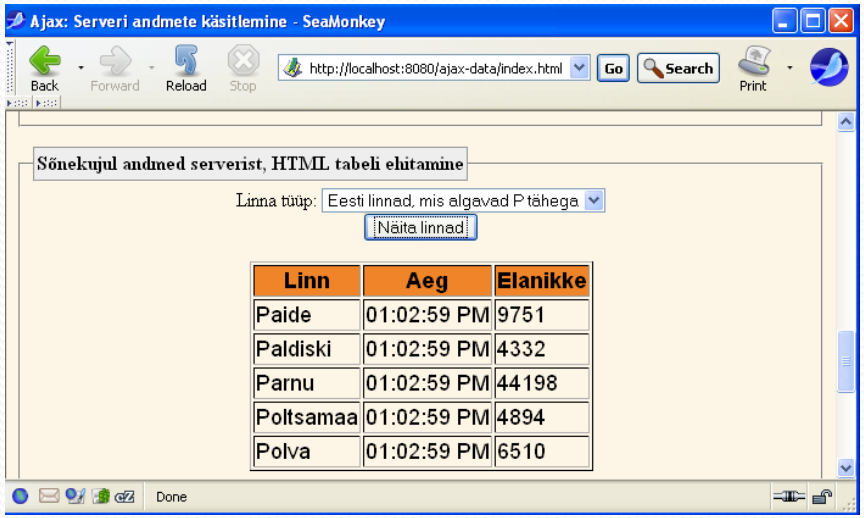
JSP kood (/WEB-INF/results/cities-string.jsp)

```
${cities[0].name}#${cities[1].name}#...#${cities[4].name}
${cities[0].shortTime}#...#${cities[4].shortTime}
${cities[0].population}#...#${cities[4].population}
```

Sõne kujul andmed: tulemused







Kombineeritud andmed

Idee

- Varem
  - Server
    - Otsustas, millisel kujul andmeid tagastada
  - Klient
    - Hardcoded "format"
    - Hardcoded vastuse töötleja funktsioon
- Nüüd
  - Server
    - Muutusi pole. Kasutab parameetrit "format" samamoodi.
  - Klient
    - Saab "format" väärtuse
    - Otsustab vastuse töötleja funktsiooni tekstivälja põhjal

JavaScript

```
function cityTable(cityTypeField, formatField,
                   resultRegion) {
  var address = "show-cities";
  var cityType = getValue(cityTypeField);
  var format = getValue(formatField);
  var data = "cityType=" + cityType +
             "&format=" + format;
  var responseHandler = findHandler(format);
  ajaxPost(address, data,
           function(request) {
             responseHandler(request, resultRegion);
           });
}

function findHandler(format) {
  if (format == "xml") {              // == is ok for strings!
    return(showXmlCityInfo);
  } else if (format == "json") {
    return(showJsonCityInfo);
  } else {
    return(showStringCityInfo);
  }
}
```

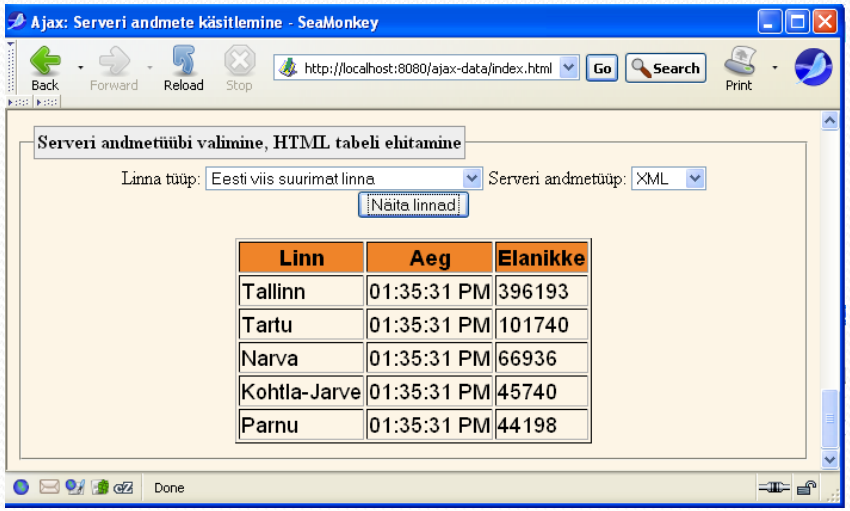
HTML

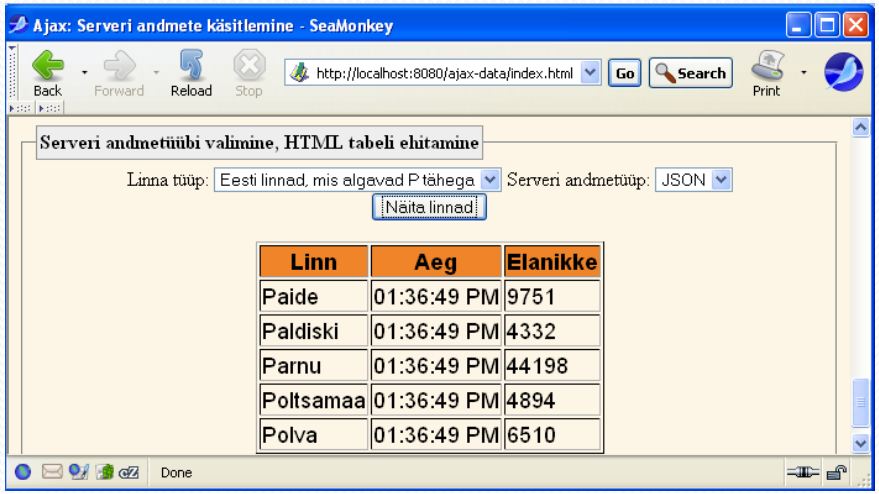
```
<fieldset>
  <legend>Choosing Server Datatype...</legend>
  <form action="#">
   <label for="city-type-4">Linna tüüp:</label>
   <select id="city-type-4">
     <option value="top-5-cities">Eesti viis ...</option>
     ...
    </select>
  <label for="data-type">Server Data Type:</label>
   <select id="data-type">
     <option value="xml" selected="selected">XML</option>
     <option value="json">JSON</option>
     <option value="string">sõne</option>
    </select>
   <br/>
   <input type="button" value="Näita linnad"
          onclick='cityTable("city-type-4", "data-type",
                             "city-table")'/>
  </form>
  <p/>
  <div id="city-table"></div>
</fieldset>
```

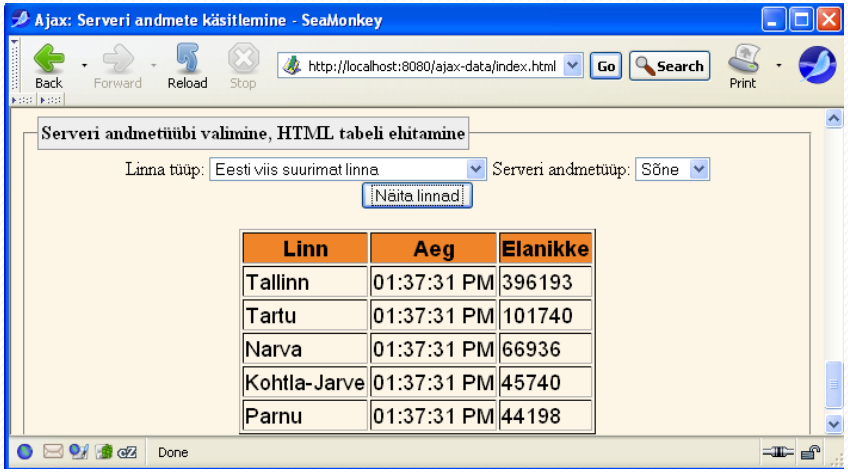
Serveri-poolne kood

- Pole muutusi

Andmete kombineerimine: tulemused







Kokkuvõte

- XML andmete töötlemine
  - `request.responseXML`
  - `getElementsByTagName`
  - `someElement.childNodes[0]`.nodeValue
- JSON andmete töötlemine
  - Edastada `eval` -ile
  - Käsitleda nagu normaalset JavaScript objekti
- String andmete töötlemine
  - `String.split` koos regulaaravaldistega
- Server
  - Kasutada MVC

Ajax Toolkits and Libraries

Client-Side Tools (JavaScript Libraries with Ajax Support)

- Prototype
  - http://www.prototypejs.org/
- Dojo
  - http://www.dojotoolkit.org/
- script.aculo.us
  - http://script.aculo.us/
- ExtJS
  - http://extjs.com/
- Yahoo User Interface Library (YUI)
  - http://developer.yahoo.com/yui/

Server-Side Tools

- Direct Web Remoting
  - Lets you call Java methods semi-directly from JavaScript
  - http://getahead.ltd.uk/dwr/
- JSON/JSON-RPC
  - For sending data to/from JavaScript with less parsing
  - http://www.json.org/
  - http://json-rpc.org/

Hybrid Client/Server Tools

- JSP custom tag libraries
  - Create tags that generate HTML and JavaScript
  - http://courses.coreservlets.com/Course-Materials/msajsp.html
- AjaxTags (built on top of script.aculo.us)
  - JSP custom tags that generate Ajax functionality
    - Supports many powerful Ajax capabilities with very simple syntax
    - http://ajaxtags.sourceforge.net
- Google Web Toolkit
  - Write code in Java, translate it to JavaScript
    - http://code.google.com/webtoolkit/
  - Also see https://ajax4jsf.dev.java.net/
    - GWT/JSF Integration Toolkit

JSF-Based Tools

- Trinidad (formerly Oracle ADF)
  - http://www.oracle.com/technology/products/jdev/htdocs/partners/addins/exchange/jsf/ (also myfaces.apache.org)
- Tomahawk
  - http://myfaces.apache.org/tomahawk/
- Ajax4JSF (and RichFaces)
  - http://labs.jboss.com/jbossajax4jsf/
- IceFaces
  - http://www.icefaces.org/
- Build your own
  - http://courses.coreservlets.com/Course-Materials/jsf.html

Books
(In Rough Order of Preference)

- Ajax in Practice
  - Crane et al. Manning.
- JavaScript: The Definitive Guide
  - Flanagan. O'Reilly.
- Foundations of Ajax
  - Asleson and Schutta. APress.
- Ajax in Action
  - Crane, Pascarello, and James. Manning.
- GWT in Action
  - Hanson and Tacy. Manning.
- Pro JSF and Ajax
  - Jacobi and Fallows. APress.
- Prototype and Scriptaculous in Action
  - Crane et al. Manning.
- Pro Ajax and Java Frameworks
  - Schutta and Asleson. APress.
- Professional Ajax
  - Zakas, et al. Wrox. *Wait for 2nd edition.*

Summary

- JavaScript
  - Define request object
    - Check for both Microsoft and non-MS objects. Identical in all apps.
  - Initiate request
    - Get request object
    - Designate an anonymous response handler function
    - Read user data with getElementById(id).value, then escape it
    - Initiate a GET or POST request
      - If POST, set Content-Type and put data in send method
      - If GET, append data on end of address after "?"
  - Handle response
    - Wait for readyState of 4 and HTTP status of 200
    - Extract return text with responseText
    - Do something with result
      - Use innerHTML to insert result into designated element
- HTML
  - Give ids to input elements and to result region. Initiate process.
- Java
  - Use JSP, servlet, or combination (MVC) as appropriate.
  - Prevent browser caching.