VEEBIRAKENDUSTE LOOMINE

MTAT.03.230 (6EAP)

8. Loeng

Helle Hein

Teema: JSP - JavaServer Pages
3. osa

Täna loengus:

- Servlettide ja JSP integreerimine:
- MVC arhitektuur
- Rakenduste paigaldamine; `web.xml`

JSP konstruktsioonide kasutamine

Lihtne rakendus

Keerukas rakendus

- Scriptielemendid, mis pöörduvad servleti koodi poole otse
- Scriptielemendid, mis pöörduvad servleti koodi kaudselt (Java klasse kasutades)
- Beans (oad)
- Tavamärgised (Custom Tags)
- Servlet/JSP - (MVC), koos ubade ja kohandatud märgistega

http://www.coreservlets.com/

Miks kasutada koos servlette ja JSP lehti?

- JSP kasutamine teeb HTML koostamise ja säilitamise lihtsamaks
  - Lihtsama funktsionaalsuse korral kasutada servleti koodi JSP skriptielementides
  - Keerukamate rakenduste korral kasutada klasse, mille poole pöörduda JSP skriptielementidest
  - Keerukamate rakenduste korral kasutada JavaBean'e ja tavamärgiseid
  - JSP eelduseks on, et üks leht annab ühe vaate

Võimalused päringu töötlemiseks

- Servlett (ainult)
  - Väljund on binaarne, nt. pilt
  - Väljundit pole, nt edasisuunamine
  - Väljundi esitus on väga muutuv
- JSP (ainult)
  - Väljund on tekst, nt HTML
  - Formaat ja paigutus on fikseeritud.
- Servleti ja JSP kombinatsioon
  - Päringule võib tulla vastuseks erinevaid vaateid
  - Keeruline andmetöötlus, kuid suhteliselt fikseeritud väljund.

- Ühendatud servlett/JSP protsess:
  - Esialgsele päringule vastab servlett
  - Servlett töötleb päringuandmeid, kasutab andmebaasi
  - Tulemused pannakse JavaBean'idesse
  - Päring edastatakse JSP lehele tulemuste esitamiseks
  - Erinevad JSP lehed vastavad erinevatele esitustele
- Seda nimetatakse "MVC" (Model View Controller) või "Model 2" JSP
- Nt. Apache Struts Framework
  http://jakarta.apache.org/struts/

Model-View-Controller (MVC) definitsioon

– Käsitlus, mille kohaselt päringu töötlemine jaotatakse kolmeks osaks

- Controller (juhtija)
- Osa, mis töötleb päringut,otsustab, millist loogikat rakendada ja millist JSP lehte kaasata

- Model:
- Klassid, mis esitavad andmeid

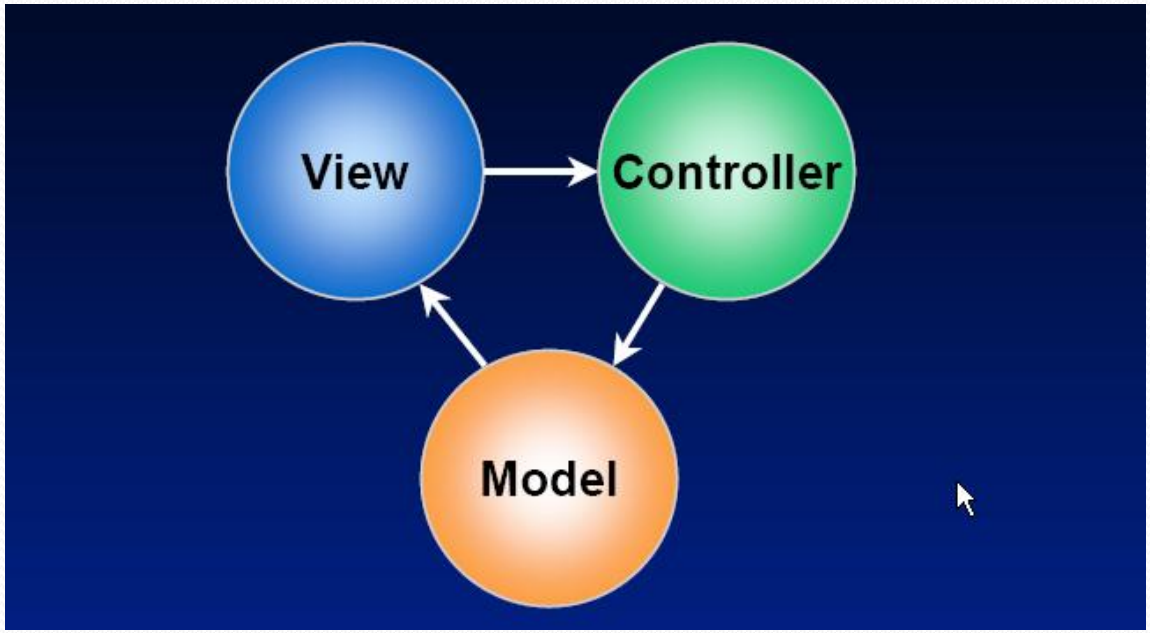
- View

JSP lehed, mis esitavad kliendile saadetavat väljundit

Näited

– MVC kasutades RequestDispatcher-it
– Struts
- JSF

MVC – Model – View – Controller arhitektuur



Model

Objekt, mis defineerib komponendi seisundi

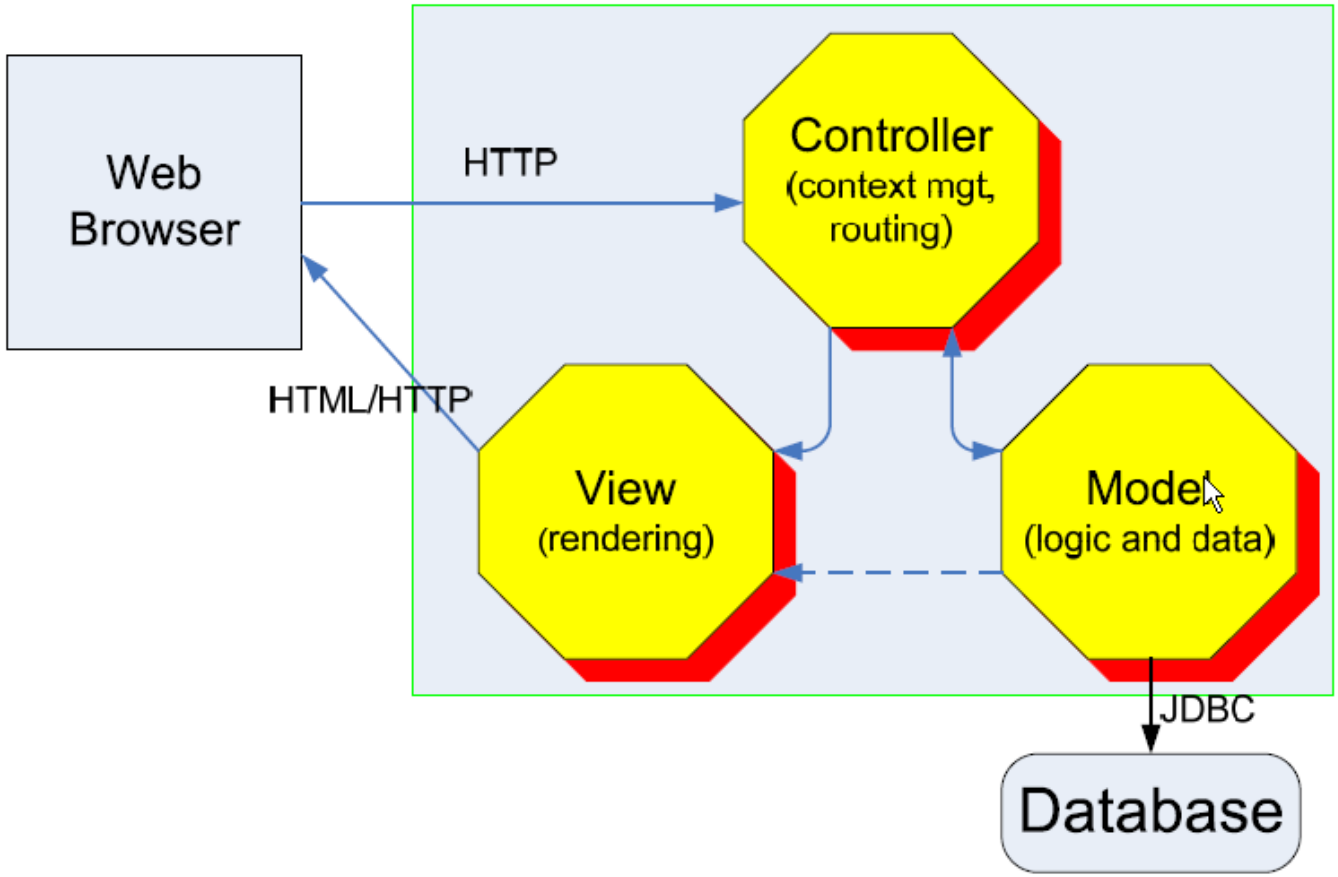
View

Komponendi visuaalne esitus ekraanil

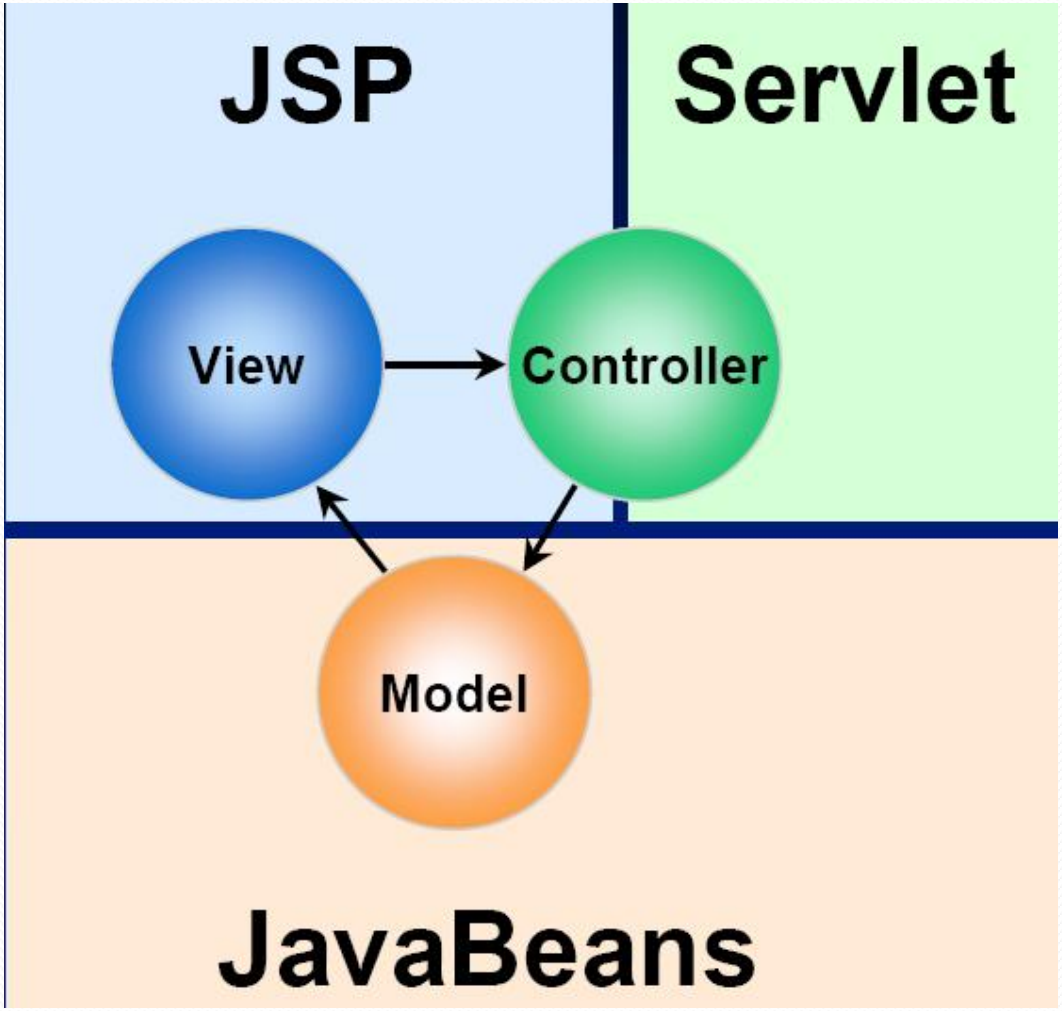
Controller

Objekt, mis vastab kasutaja päringule

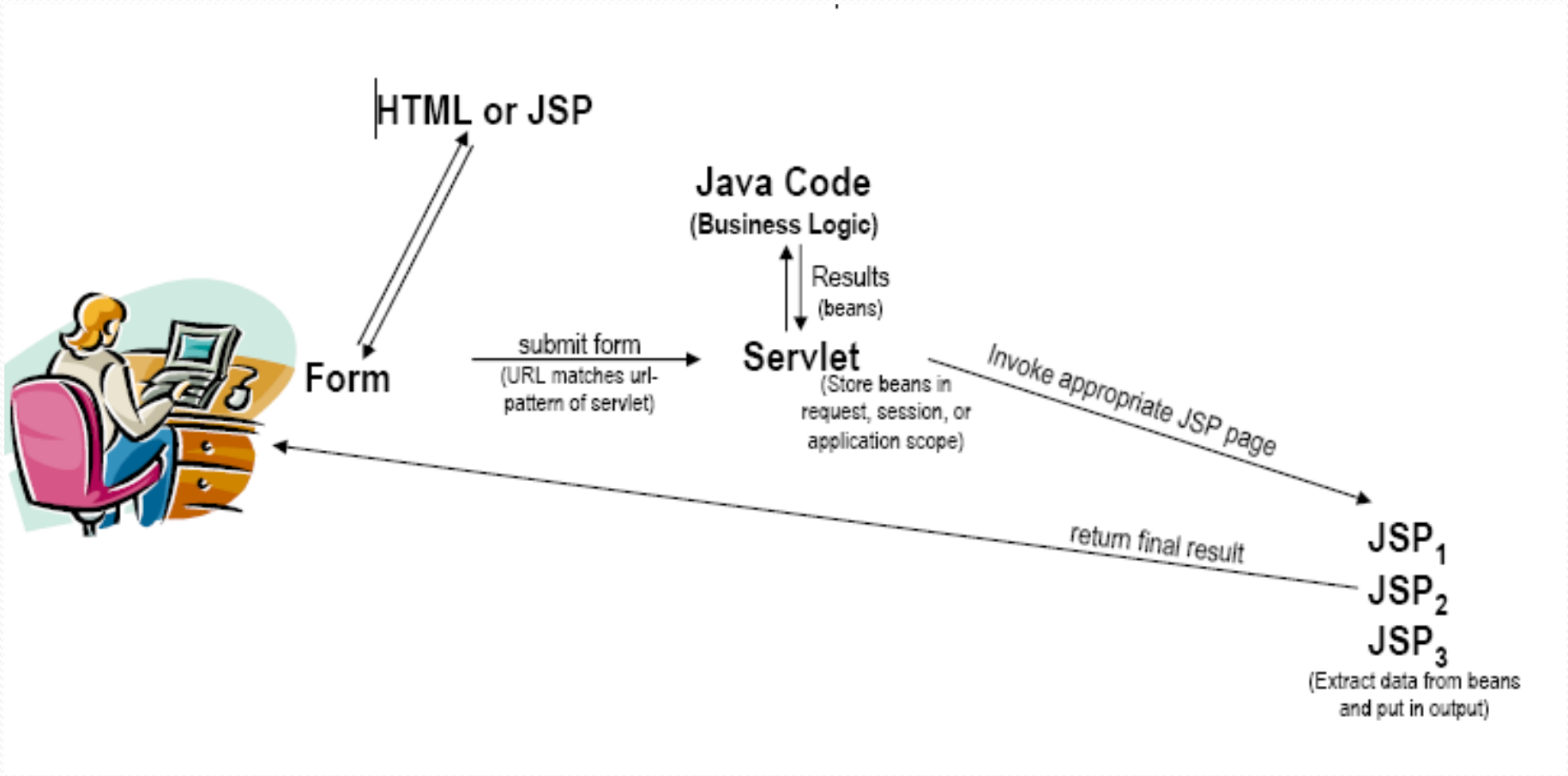
MVC for Web Applications



MVC – koos JSP, servlettide ja JavaBeanidega

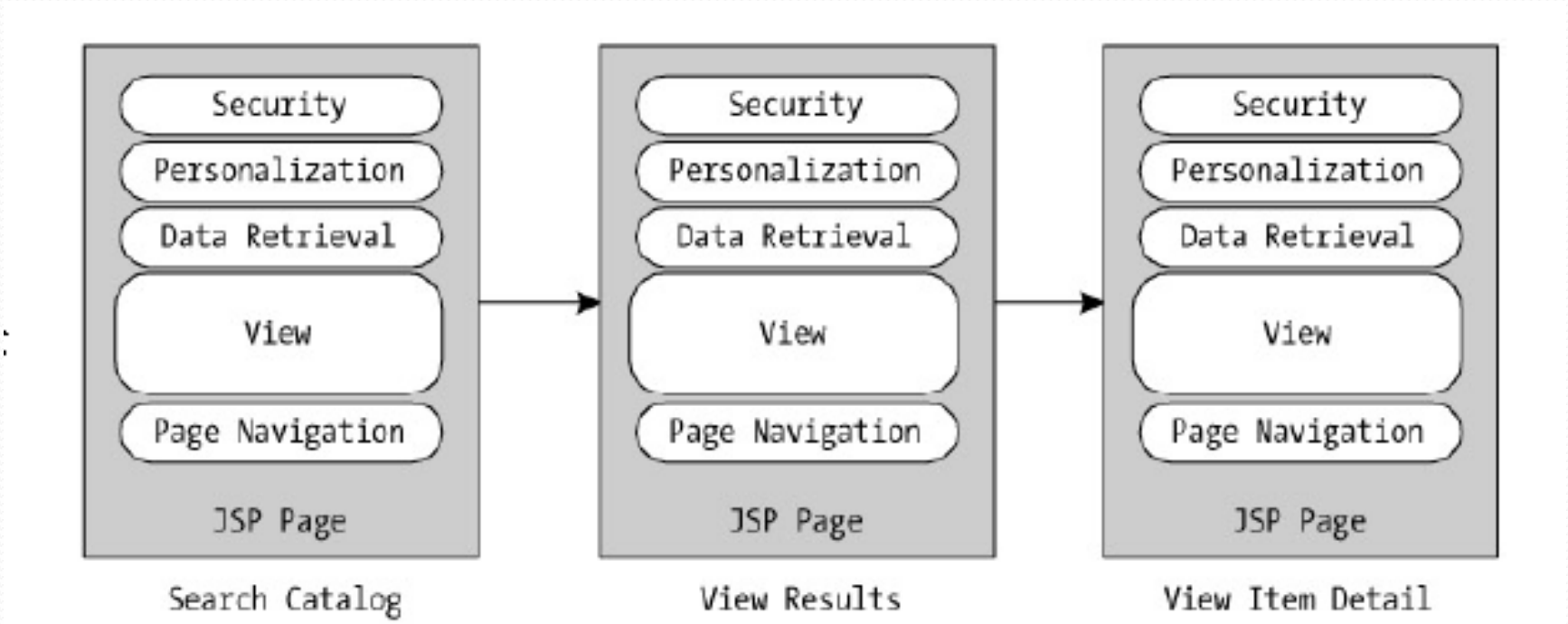


MVC juhtimisvood

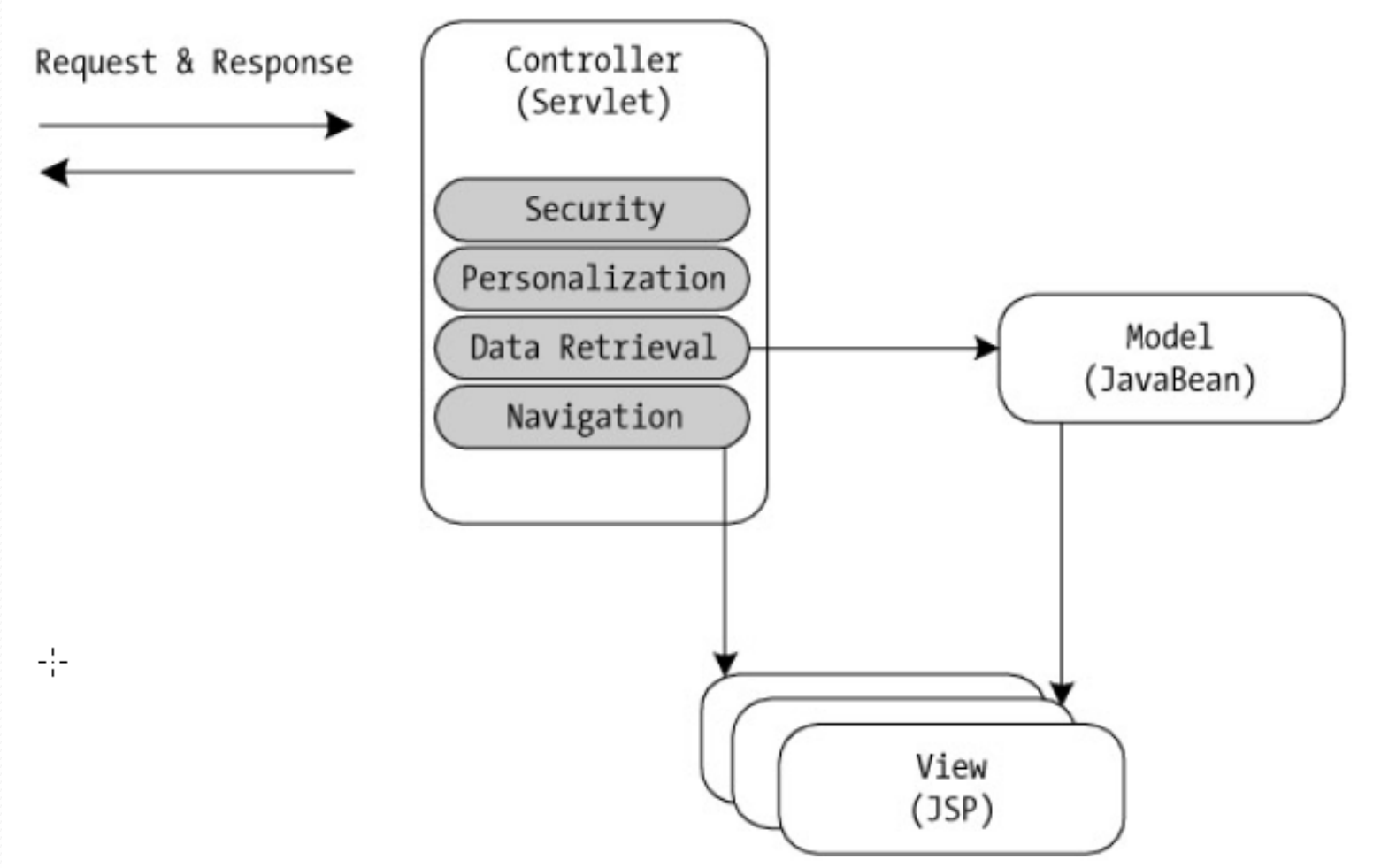


http://www.coreservlets.com/

Näide – Veebirakendus, mis näitab kataloogilehti (ilma MVC)



Näide – Veebirakendus, mis näitab kataloogilehti (MVC)



MVC veebiraamistikud

- Java raamistikud:

– Päringupõhised (request-driven): JSP Model 2 Architecture, Struts, Spring MVC

– Sündmuspõhised komponentidele orienteeritud (Event-driven, component-oriented), : JSF, Tapestry, Spring Web Flow, Wicket, Aranea…

- PHP MVC raamistikud:

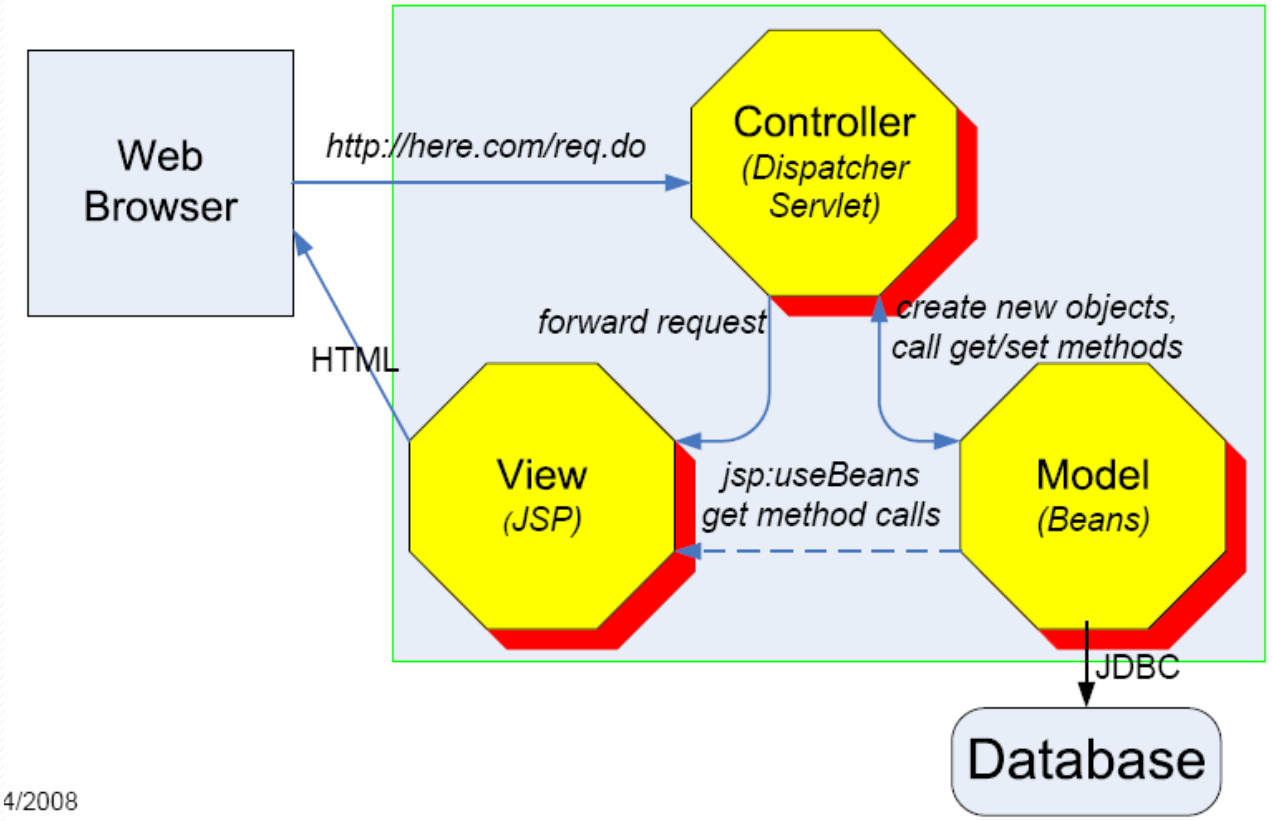
– Päringupõhised: Struts

– Sündmuspõhised, komponentidele orienteeritud: PRADO, Symfony, …

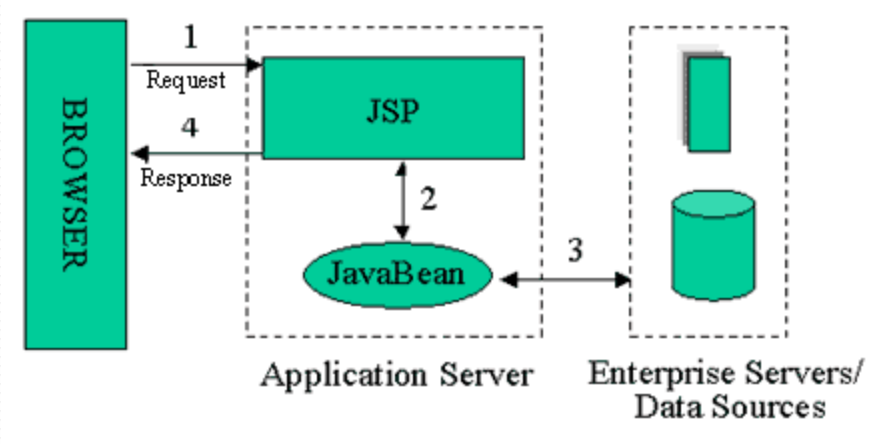
- Microsoft: ASP.Net MVC Framework

Marlon Dumas 2008

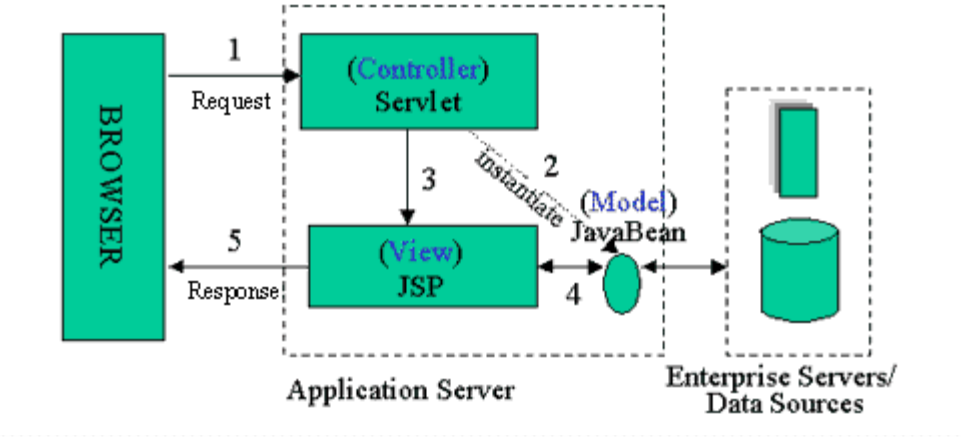
Request-driven MVC (JSP Model 2 Architecture)



Marlon Dumas 2008



JSP Model 1 arhitektuur



JSP Model 2 arhitektuur

Päringupõhine MVC

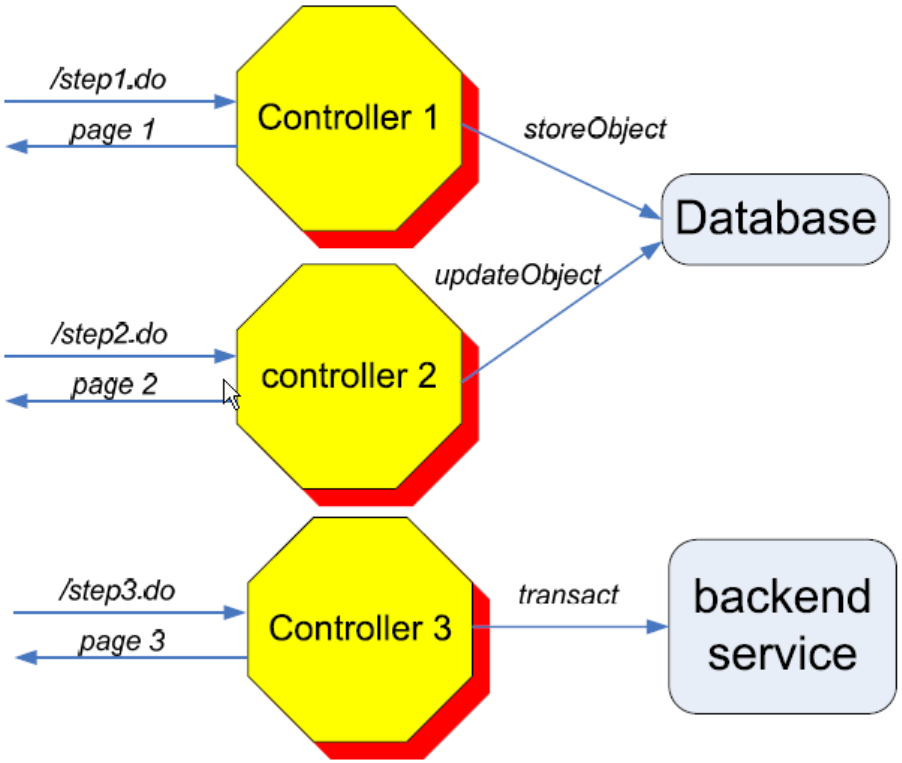
- Renderdamine ja äriloogika on eraldi
- Kuid:

– Mudel on “tasapinnaline” (hulk JavaBean-e)

– Ühendus vaate ja mudeli vahel tuleb teostada käsitsi

Marlon Dumas 2008

Page flows in Request-driven MVC



Marlon Dumas 2008

Sündmuspõhine, komponentidele orienteeritud MVC raamistikud

- Rakendus koosneb komponentidest
- Iga komponent sisaldab:

– View (renderdamise jaoks)

– Model

- Komponent (nt vorm) võib sisaldada teisi komponente (nt. tekstivälja ja sisestusvälja)
- Komponendid võivad reageerida sündmustele

Marlon Dumas 2008

Päringu edastamine servletist JSP lehele

- getRequestDispatcher tagastab päringute dispetšeri

```
String url = "/presentations/presentation1.jsp";
RequestDispatcher dispatcher =
    getServletContext().getRequestDispatcher(url);
```

  - Meetod forward annab juhtimise täielikult üle uuele lehele
    - See on MVC lähenemine
  - include väljakutse võtab uue lehe väljundi ja jätkamine samast kohast

Näidiskood – servleti doGet

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException {
  String operation = request.getParameter("operation");
  if (operation == null) {
    operation = "unknown";  }
  if (operation.equals("operation1")) {
    gotoPage("/operations/presentation1.jsp",
             request, response);
  }
  else if (operation.equals("operation2")) {
    gotoPage("/operations/presentation2.jsp",
             request, response);
  }
  else {
    gotoPage("/operations/unknownRequestHandler.jsp",
             request, response);  }}
```

Päringu edastamine servletist JSP lehele

```
private void gotoPage(String address,
                      HttpServletRequest request,
                      HttpServletResponse response)
    throws ServletException, IOException {
  RequestDispatcher dispatcher =
    getServletContext().getRequestDispatcher(address);
  dispatcher.forward(request, response);
}
```

JSP useBean skoobid

- request

```
<jsp:useBean id="..." class="..." scope="request" />
```

- session

```
<jsp:useBean id="..." class="..." scope="session" />
```

- application

```
<jsp:useBean id="..." class="..." scope="application" />
```

- page

```
<jsp:useBean id="..." class="..." scope="page" />
```

või

```
<jsp:useBean id="..." class="..." />
```

Seda skoopi ei kasutata MVC (Model 2) arhitektuuris

Andmete salvestamine servletis selliselt, et JSP leht kasutab andmeid ainult antud päringu korral

- Eesmärk
  - Salvestada andmed servletis, mida JSP leht saab hiljem kasutada **antud päringu** korral
- Servleti süntaks andmete salvetamiseks

```
SomeClass value = new SomeClass(…);
request.setAttribute("key", value);
// RequestDispatcher JSP lehele suunamiseks
```

- JSP süntaks andmete kättesaamiseks

```
<jsp:useBean
   id= "key"
   class= "SomeClass"
   scope="request" />
```

Andmete salvestamine servletis selliselt, et JSP leht kasutab andmeid ainult **antud kliendi** hilisemate päringute korral

- Eesmärk
  - Salvestada andmed servletis, mida JSP leht saab hiljem kasutada **antud kliendi hilisemate päringute** korral
- Servleti süntaks

```
SomeClass value = new SomeClass(…);
HttpSession session =
  request.getSession(true);
session.setAttribute("key", value);
```

  - // RequestDispatcher JSP lehele suunamiseks
- JSP süntaks

```
<jsp:useBean
   id="key"
   class="SomeClass"
   scope="session" />
```

Teine võimalus seansi jälgimiseks

- Kasutada `response.sendRedirect` käsku `RequestDispatcher.forward` asemel
- Erinevused: Kui kasutada sendRedirect:
  - Klient näeb JSP URL (Kui kasutada `RequestDispatcher.forward`, siis klient näeb servleti URL)
  - Kõigepealt suunatakse kliendile tagasi (two round trips to client)
- `sendRedirect` eelised:
  - Klient võib külastada JSP lehte eraldi
    - Klient võib JSP lehe aadressi salvestada (bookmark)
- `sendRedirect` puudused
  - Kuna kasutaja võib külastada JSP lehte ilma servletita, siis JSP lehele ei tarvitse andmed kättesaadavad olla
    - JSP leht peab selle olukorra tuvastama

Andmete salvestamine servletis selliselt, et JSP leht kasutab andmeid **suvalise kliendi** päringu korral

- Servleti süntaks

```
synchronized(this) {
  SomeClass value = new SomeClass(…);
  getServletContext().setAttribute("key",
                                   value);
  // RequestDispatcher JSP lehele suunamiseks
}
```

- JSP süntaks

```
<jsp:useBean
   id="key"
   class="SomeClass"
   scope="application" />
```

Päringu edasisuunamine JSP lehelt

```
<jsp:forward page="Relative URL" />
```

URL-i võib anda ka avaldisena:

```
<% String destination;
if (Math.random() > 0.5) {
  destination = "/examples/page1.jsp";
}
else {
  destination = "/examples/page2.jsp";
}
%>
<jsp:forward page="<%= destination %>" />
```

Lehtede kaasamine ilma neile suunamata

- Kui kasutada `RequestDispatcher` meetodit `forward`, siis

– Juhtimine läheb *täielikult üle uuele lehele*
– Esialgne leht *ei saa genereerida mingit väljundit*

- Kui kasutada `RequestDispatcher` meetodit `include`, siis :

– Juhtimine läheb *ajutiselt üle uuele lehele*
– Esialgne leht *saab genereerida väljundit enne ja pärast kaasatud lehte*
– Servlett ei näe kaasatud väljundit

Lehtede kaasamine ilma neile suunamata

```
response.setContentType("text/html");
String firstTable, secondTable, thirdTable;
if (someCondition) {
  firstTable = "Scores.jsp";
  secondTable = "Prices.jsp";
  thirdTable = "Weather.jsp";
} else if (...) { ... }
RequestDispatcher dispatcher =
request.getRequestDispatcher("Header.jsp");
dispatcher.include(request, response);
dispatcher = request.getRequestDispatcher(firstTable);
dispatcher.include(request, response);
dispatcher = request.getRequestDispatcher(secondTable);
dispatcher.include(request, response);
dispatcher = request.getRequestDispatcher(thirdTable);
dispatcher.include(request, response);
dispatcher = request.getRequestDispatcher("Footer.jsp");
dispatcher.include(request, response);
```

MVC rakendamine kasutades `RequestDispatcher`

1. **Defineerida Javabean-id andmete hoidmiseks**

2. **Kasutada servlette päringute töötlemiseks**

– Servlett loeb päringu parameetreid, analüüsib andmeid, jne

3. Täita JavaBean-id andmetega

– Servlett sisaldab tegevusloogikat (nn äriloogika)
Tulemused pannakse JavaBean-idesse.

4. Paigutada JavaBean kas `request, session` või servlet context-i**.**

– Servletis kasutatakse meetodit `setAttribute` objektile `request`, `session` või servletContext, et paigutada JavaBean objektide viidad.

MVC rakendamine kasutades `RequestDispatcher`

5. Edastada päring JSP lehele

– Servlett teeb kindlaks, milline JSP on sobiv antud päringu korral tulemusi esitama

6. Andmete kättesaamine JavaBean-idest

–JSP lehel kasutada jsp:useBean koos atribuudi scope sobiva väärtusega.
Siis `jsp:getProperty` abil väljastada JavaBean omadusi.
–JSP leht näitab JavaBean-ides sisalduvaid andmeid

MVC - Näide päringupõhisest andmete jagamisest

- **Eesmärk**– Näidata kasutajale juhuslikku arvu

- **Tüüp**

– Iga päring peab andma uue arvu, seega on sobiv päringupõhine lähenemine - `request`

NumberBean.java

```
package  arvud;
public class NumberBean {
 private double num = 0;

 public NumberBean(double number) {
  setNumber(number);
 }
 public double getNumber() {
  return(num);
 }
 public void setNumber(double number) {
  num = number;
 }
}
```

RandomNumberServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class RandomNumberServlet extends HttpServlet {
public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
  NumberBean bean = new NumberBean(Math.random());
  request.setAttribute("randomNum", bean);
  String address ="/WEB-INF/mvc/RandomNum.jsp";
  RequestDispatcher dispatcher =
        request.getRequestDispatcher(address);
        dispatcher.forward(request, response);
}
}
```

RandomNum.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
<HEAD>
<TITLE>Jagatud juurdep&auml;&auml;s: Leht1 1</TITLE>
</HEAD>
<BODY>
  <jsp:useBean id="randomNum" class="arvud.NumberBean"
       scope="request" />
  <H2>Juhuslik arv:
    <jsp:getProperty name="randomNum" property="number" />
  </H2>
</BODY></HTML>
```

JSP 2.0

```
…
<BODY>
  <H2>Random Number:
    ${randomNum.number}
  </H2>
</BODY></HTML>
```


```
RandomNumberServlet.class -> ..\examples\WEB-INF\classes

NumberBean.class -> ..\examples\WEB-INF\classes\arvud

RandomNum.jsp -> ..\examples\WEB-INF\mvc
```

Lisaks registreerimine `web.xml`-is

MVC - Näide seansipõhisest andmete jagamisest

- **Eesmärk**

– Näidata kliendi ees- ja perekonnanime
– Kui klient ei sisesta antud päringul oma nime, siis näidata nime, mis ta on sisestanud mõnel eelmistel korral
– Kui nime pole üldse sisestatud, siis näidata teade

- **Tüüp**

– Andmed säilitatakse iga kliendi jaoks, seega seansipõhine lähenemine - `session`

NameBean.java

```
package arvud;
public class NameBean {
private String firstName = "Puudub eesnimi";
private String lastName = "Puudub perekonnanimi";
  public NameBean() {}
  public NameBean(String firstName, String lastName) {
    setFirstName(firstName);
    setLastName(lastName);
  }
  public String getFirstName() {
    return(firstName);
  }
  public void setFirstName(String fname) {
    firstName = fname;
  }
…
}
```

RegistrationServlet.java

```
//import siia
public class RegistrationServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
       HttpServletResponse response)
       throws ServletException, IOException {
  HttpSession session = request.getSession();
  arvud.NameBean nameBean=(arvud.NameBean)
       session.getAttribute("nameBean");
  if (nameBean == null) {
    nameBean = new arvud.NameBean();
    session.setAttribute("nameBean", nameBean);
  }
```

RegistrationServlet.java

```
String firstName = request.getParameter("firstName");
if ((firstName != null) &&(!firstName.trim().equals(""))) {
  nameBean.setFirstName(firstName);
}
String lastName = request.getParameter("lastName");
if ((lastName != null) &&(!lastName.trim().equals(""))) {
  nameBean.setLastName(lastName);
}
String address ="/WEB-INF/mvc/ShowName.jsp";
RequestDispatcher dispatcher =
      request.getRequestDispatcher(address);
dispatcher.forward(request, response);
}
}
```

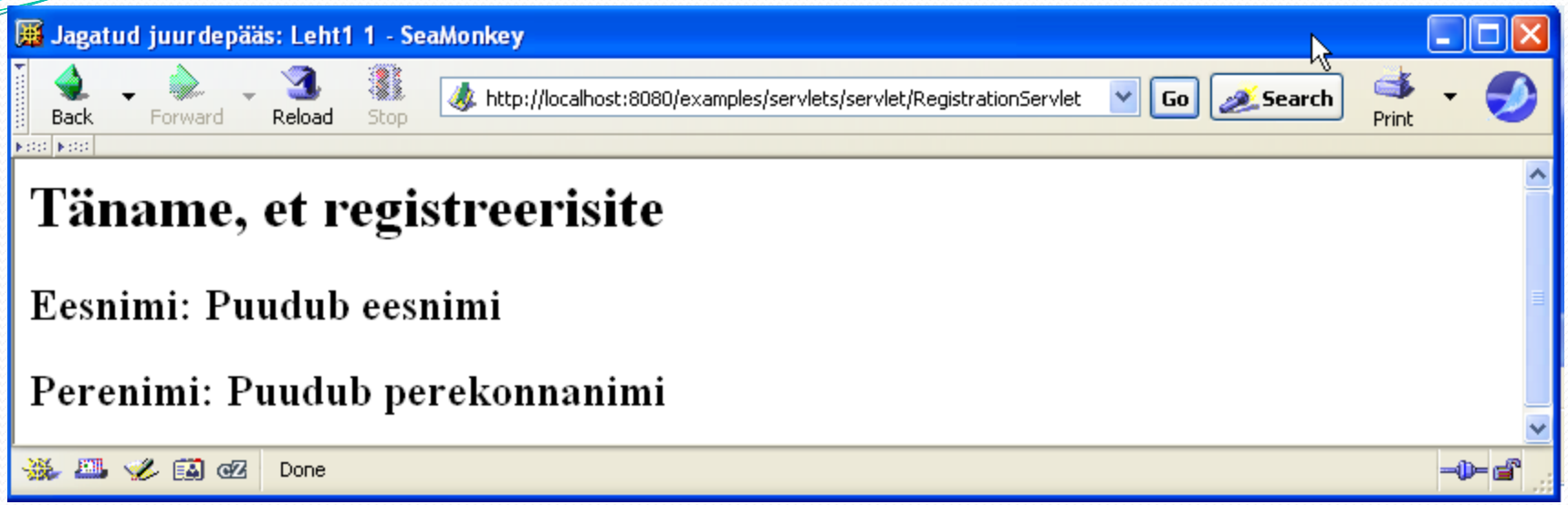
ShowName.jsp

```
…
<BODY>
<H1>T&auml;name, et registreerisite</H1>
  <jsp:useBean id="nameBean" class="arvud.NameBean"
       scope="session" />
<H2>Eesnimi:
  <jsp:getProperty name="nameBean" property="firstName" />
</H2>
<H2>Perenimi:
  <jsp:getProperty name="nameBean" property="lastName" />
</H2>
</BODY>
</HTML>
```

ShowName.jsp

JSP 2.0

```
…
<BODY>
  <H1>T&auml;name, et registreerisite</H1>
  <H2>Eesnimi:
    ${nameBean.firstName}
  </H2>
  <H2>Perenimi:
    ${nameBean.lastName}
  </H2>
</BODY>
</HTML>
```


Jagatud juurdepääs: Leht1 1 - SeaMonkey
Back
Forward
Reload
Stop
http://localhost:8080/examples/servlets/servlet/RegistrationServlet
Go
Search
Print
Täname, et registreerisite
Eesnimi: Puudub eesnimi
Perenimi: Puudub perekonnanimi
Done


Jagatud juurdepääs: Leht1 1 - SeaMonkey
Back
Forward
Reload
Stop
p://localhost:8080/examples/servlets/servlet/RegistrationServlet?firstName=Helle&lastName=Hein
Go
Täname, et registreerisite
Eesnimi: Helle
Perenimi: Hein
Done

Front Controller - Eelkontrollija

Definitsioon

– MVC käsitlus, kus üks servlett (või filter) on kogu rakenduse alguspunktiks. Eelkontrollija suunab päringud kõikidele teistele.

- **Näited**

– MVC rakendus, kus on ainult üks servlett
– Struts
– JSF

Intercepting Filter - vahefilter

Definitsioon

– Käsitlus, kus muudetakse:
- Päringuid (enne nende jõudmist servletti või JSP lehele)
- Vastuseid (enne nende jõudmist kliendini)

- **Näited**

– Servlett ja JSP filtrid

Filtrid

- Seostatakse suvalise arvu servlettide või JSP lehtedega

- Filter kontrollib servletile või JSP lehele tulevat päringut ning seejärel:

– Kaasab servleti või JSP lehe (ressursi) normaalselt

– Kaasab ressursi muudetud päringuga.

– Kaasab ressursi muudetud vastusega kliendile.

– Takistab ressursi kaasamist ja suunab teisele ressursile (või genereerib väljundi, ..)

Filtrite eelised

- **Varjavad ühist käitumist**

– Nt. meil on 50 servletti või JSP lehte, mis vajavad pakkimist, et hoida kokku allalaadimise aega. Teha üks pakkimise filter ja rakendada seda kõigile 50 ressursile.

- **Eraldavad kõrgtasemel otsuseid juurdepääsuks esitusest**

– Nt. on vaja blokeerida juurdepääs kindlatelt lehtedelt ilma neid lehti muutmata. Luua üks piirangu filter ja rakendada seda nii paljudele lehtedele nagu vaja.

- **Rakendada väikesi muudatusi paljudele ressurssidele**

– On olemas ressursid, milles on vaja ainult firma nime muuta. Teha stringiasendusfilter ja rakendada seda kus vaja.

Filtrite loomine

1. Luua klass, mis rakendab liidest Filter.
– Meetodid: `doFilter, init, destroy`

2. Panna funktsionaalsus meetodisse doFilter.
– Argumendid: `ServletRequest, ServletResponse, FilterChain`

3. Pöörduda `FilterChain` meetodi `doFilter` poole.
– See kaasab järgmise filtri või tegeliku ressursi.

4. Registreerida filter vajalike servlettide ja JSP lehtede külge.
– Kasutada filter ja filter-mapping märgiseid failis `web.xml`.

Filtrite loomine

```
public void doFilter(ServletRequest request, ServletResponse
      response, FilterChain chain) throws ServletException,
      IOException
  {
     ….
     chain.doFilter(request,response);
  }
```

Filtrite loomine - näide

```
package filtrid;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
public class ReportFilter implements Filter {
  public void doFilter(ServletRequest request,ServletResponse
      response,FilterChain chain) throws ServletException,
      IOException {
    HttpServletRequest req = (HttpServletRequest)request;
    System.out.println(req.getRemoteHost() +
      " tried to access " +req.getRequestURL() +
      " on "+new Date() + ".");
    chain.doFilter(request,response);
}
```

Filtrite loomine - näide

```
public void init(FilterConfig config)
throws ServletException {
}
public void destroy() {
}
}
```

Filtrid - registreerimine

```
…
<web-app…>

<filter>
  <filter-name>Reporter</filter-name>
  <filter-class>
    filtrid.ReportFilter
  </filter-class>
</filter>
```

Filtrid – seostamine konkreetse URL-iga

```
<filter-mapping>
  <filter-name>Reporter</filter-name>
  <url-pattern>/index.jsp</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>Reporter</filter-name>
  <servlet-name>TodaysSpecial</servlet-name>
</filter-mapping>
…
</web-app>
```

Vahekokkuvõte

- Kasutada MVC (Model 2) lähenemist kui:
  - Ühele päringule võib olla mitu erineva kujuga vastust
  - Paljudel lehtedel on sarnane töötlus
- Arhitektuur
  - Servlett vastab esialgsele päringule
  - Servlett teeb tegeliku töötluse ja paigutab tulemused JavaBean-idesse
    - JavaBeanid paigutatakse kas `HttpServletRequest`, `HttpSession` või `ServletContext`-i
  - Servlett suunab JSP lehele `RequestDispatcher` meetodiga `forward`
  - JSP leht loeb andmeid ubadest kasutades`jsp:useBean` koos sobiva skoobiga (`request`, `session` või `application`)

Veebirakenduste paigaldamine

- Struktuur

– Servletid, JSP lehed, HTML failid, klassid, JavaBean-id, märgisteteegid, paigutatakse kindlasse kataloogihierarhiasse või faili.

- Ühine URL prefiks

–– http://host/webAppPrefix/tooted/...

- `web.xml` koordineerib palju aspekte

Struktuur

- JSP ja tavaline veebisisu (HTML, stiililehed, pildid jne.):

– Rakenduse põhikataloogi või selle alla

- Servletid, JavaBean-id (pakkimata), klassid:

– WEB-INF/classes (kui paketis, siis vastavas alamkataloogis)

- JAR failid:

– WEB-INF/lib.

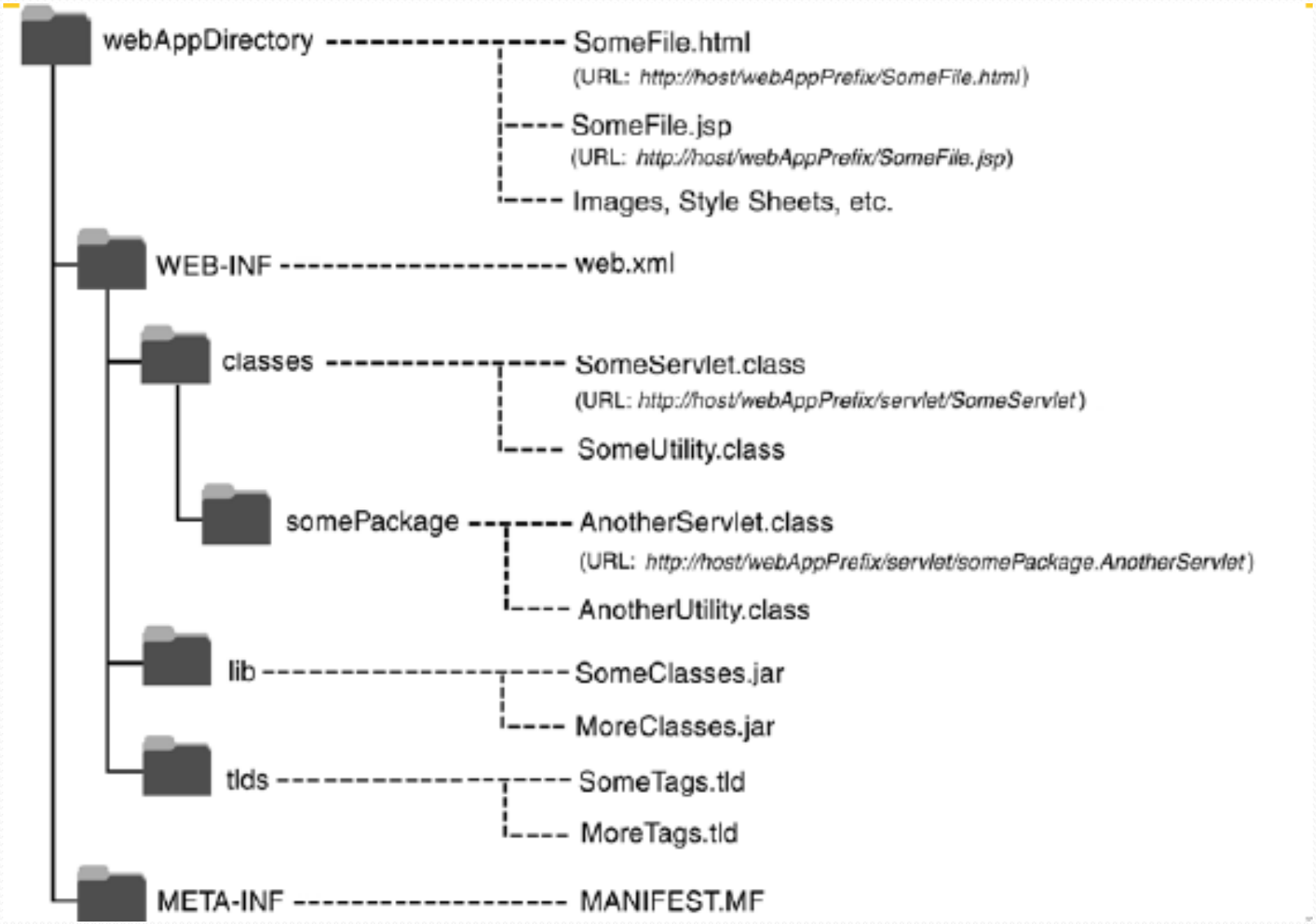
`web.xml`:

– WEB-INF

- Märgiseteegi failid (.tld):

– Rakenduse põhikataloogi või selle alla

- Kataloog WEB-INF ei ole klientidele otseselt kättesaadav


webAppDirectory
SomeFile.html
(URL: http://host/webAppPrefix/SomeFile.html)
SomeFile.jsp
(URL: http://host/webAppPrefix/SomeFile.jsp)
Images, Style Sheets, etc.
WEB-INF
web.xml
classes
SomeServlet.class
(URL: http://host/webAppPrefix/servlet/SomeServlet)
SomeUtility.class
somePackage
AnotherServlet.class
(URL: http://host/webAppPrefix/servlet/somePackage.AnotherServlet)
AnotherUtility.class
lib
SomeClasses.jar
MoreClasses.jar
tlds
SomeTags.tld
MoreTags.tld
META-INF
MANIFEST.MF

web.xml -Deployment Descriptor

- Tomcat jälgib `web.xml` ja laeb uuesti veebirakenduse kui web.xml muutub.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=
"http://java.sun.com/xml/ns/j2ee web-app_2_4.xsd"
version="2.4">

<!-- Siia kõik vajalikud elemendid -->
</web-app>
```

`web.xml` -Deployment Descriptor

Selline järjekord on oluline Java Servlet versioon 2.3 korral

- `icon`
- `display-name`
- `description`
- `distributable`
- `context-param`
- `filter`
- `filter-mapping`
- `listener`
- `servlet`
- `servlet-mapping`
- `session-config`
- `mime-mapping`

Tomcat 5.5 ja 6.0 korral:
`http://wiki.metawerx.net/wiki/Web.xml`

web.xml -Deployment Descriptor

- welcome-file-list
- error-page
- taglib
- resource-env-ref
- resource-ref
- security-constraint
- login-config
- security-role
- env-entry
- ejb-ref
- ejb-local-ref

```
<servlet>
  <servlet-name>MyName</servlet-name>
  <servlet-class>myPackage.MyServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>MyName</servlet-name>
  <url-pattern>/MyAddress</url-pattern>
</servlet-mapping>
```

URL

- **http://hostname/webappName/MyAddress**

Kellel on vaja seadistada servleti käitumist?

- **Autor**
  – Muuta koodi

- **Lõppkasutaja**
  – Sisestada väärtus HTML vormi

- **Paigaldaja**
  – Panna initsialiseerimisparameetreid faili web.xml

Initsialiseerimine

- Servletid
  - `ServletConfig.getInitParameter` meetodist `init`
- JSP lehed
  - `ServletConfig.getInitParameter` meetodist `jspInit`

  Kasutada `jsp-file` märgise `servlet-class` asemel.
- ServletContext
  - `ServletContext.getInitParameter`
- Filtrid

- Kuularid (listeners)

Servleti initsialiseerimisparameetrid

```
<servlet>
  <servlet-name>InitTest</servlet-name>
  <servlet-class>arvud.InitServlet</servlet-class>
  <init-param>
    <param-name>firstName</param-name>
    <param-value>Larry</param-value>
  </init-param>
  <init-param>
    <param-name>emailAddress</param-name>
    <param-value>ellison@microsoft.com</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>InitTest</servlet-name>
  <url-pattern>/showInitValues</url-pattern>
</servlet-mapping>
```

Servleti initsialiseerimisparameetrite lugemine

```
public class InitServlet extends HttpServlet {
private String firstName, emailAddress;
  public void init() {
    ServletConfig config = getServletConfig();
    firstName = config.getInitParameter("firstName");
    if (firstName == null) {
      firstName = "Missing first name";
    }
    emailAddress = config.getInitParameter("emailAddress");
    if (emailAddress == null) {
      emailAddress = "Missing email address";
    }
  }
public void doGet(…) … { … }
```

JSP initsialiseerimisparameetrid

```
<servlet>
  <servlet-name>InitPage</servlet-name>
  <jsp-file>/InitPage.jsp</jsp-file>
  <init-param>
    <param-name>firstName</param-name>
    <param-value>Bill</param-value>
  </init-param>
  <init-param>
    <param-name>emailAddress</param-name>
    <param-value>gates@oracle.com</param-value>
  </init-param>
</servlet>
servlet-mapping>
  <servlet-name>InitPage</servlet-name>
  <url-pattern>/InitPage.jsp</url-pattern>
</servlet-mapping>
```

JSP initsialiseerimisparameetrite lugemine- negatiivne näide

```
<UL>
<LI>First name: <%= firstName %>
<LI>Email address: <%= emailAddress %>
</UL>
…
<%!
private String firstName, emailAddress;
public void jspInit() {
ServletConfig config = getServletConfig();
firstName = config.getInitParameter("firstName");
if (firstName == null) { firstName = "No first name"; }
emailAddress = config.getInitParameter("emailAddress");
if (emailAddress == null) { emailAddress = "No email"; }
}
%>
```

Rakenduse initsialiseerimisparameetrid

```
<context-param>
  <param-name>support-email</param-name>
  <param-value>blackhole@mycompany.com</param-value>
</context-param>
```

Rakenduse `Welcome` lehed

```
<welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>index.html</welcome-file>
</welcome-file-list>
```

Rakenduse vealehed

```
<web-app...>
  <error-page>
    <error-code>404</error-code>
    <location>/WEB-INF/NotFound.jsp</location>
  </error-page>
...
</web-app>
```

Seansi juhtimine

- Lõpetamine

```
session.invalidate()
```

- Ajapiirang

```
session.setMaxInactiveInterval(...)
```

- Ajapiirang rakenduse piires

```
<session-config>
  <session-timeout>120</session-timeout>
</session-config>
```

Skriptimise või EL keelamine

Skriptimise keelamine

```
<jsp-property-group>
  <url-pattern>*.jsp</url-pattern>
  <scripting-invalid>true</scripting-invalid>
</jsp-property-group>
```

EL keelamine

```
<jsp-property-group>
  <url-pattern>*.jsp</url-pattern>
  <el-ignored>true</el-ignored>
</jsp-property-group>
```