VEEBIRAKENDUSTE LOOMINE

MTAT.03.230 (6EAP)

7. Loeng

Helle Hein

Teema: JSP - JavaServer Pages
2. osa

Täna loengus:

- JSP – Beans (oad)
- Tavamärgised (Custom Tags)

- JSTL

JSP konstruktsioonide kasutamine

Lihtne rakendus

Keerukas rakendus

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

http://www.coreservlets.com/

JavaBean on Java klass, mis rahuldab järgmisi tingimusi:

- Tal peab olema argumentideta **public** konstruktor;
- Klass peab olema **public**;
- Võib olla üks või mitu **public** meetodit, mis kirjeldavad mingit omadust (property) – getter-setter (piilumeetodid);
- Isendimuutujad peavad olema **private**;
- Väärtusi peab saama muuta läbi piilumeetodite **getXxx()** and **setXxx()**
- Kui klassis on meetod **getName()**, mis tagastab String'i, siis öeldakse, et klassil on omadus nimega name.
- **boolean** omaduste korral meetodi nimeks **isXxx()** mitte **getXxx()**
- ...

Näide

```
public class SimpleCountingBean {
 private int count;
 public SimpleCountingBean() {
  count = 0;
 }
 public int getCount() {
  count++;
  return count;
 }
}
```

- klass tuleb varem kompileerida
- paketis
- rakendada liidest Serializable

Miks kasutada piilumeetodeid, mitte public muutujaid

- JavaBean korral ei tohi kasutada `public` isendimuutujaid

```
public double speed;
```

asendada

```
private double speed;

public double getSpeed() {
  return(speed);
}
public void setSpeed(double newSpeed) {
  speed = newSpeed;
}
```

Miks kasutada piilumeetodeid, mitte public muutujaid

1) Võib seada väärtustele piiranguid

```
public void setSpeed(double newSpeed) {
  if (newSpeed < 0) {
    sendErrorMessage(...);
    newSpeed = Math.abs(newSpeed);
  }
  speed = newSpeed;
}
```

Miks kasutada piilumeetodeid, mitte `public` muutujaid

2) Võib muuta sisemist esitust ilma liidest muutmata

```
// Kasutada uut mõõtühikut(kph, mitte mph)

public void setSpeed(double newSpeed) {
  speedInKPH = convert(newSpeed);
  speed = newSpeed;
}

public void setSpeedInKPH(double newSpeed) {
  speedInKPH = newSpeed;
}
```

Miks kasutada piilumeetodeid, mitte public muutujaid

3) Võib rakendada kõrvalmõjusid

```
public double setSpeed(double newSpeed) {
  speed = newSpeed;
  updateSpeedometerDisplay();
}
```

JavaBean kasutamine JSPs

- Formaat

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

- Eesmärk
  - Lubada Java klassidest isendite loomist ilma Java programmeerimiseta (XML-ühilduv süntaks)

- Märkused
  - JSP tegevus

```
<jsp:useBean id= "raamat1" class= "riiul.Raamat" />
```

  on ekvivalentne koodiga (mitte alati)

```
<% riiul.Raamat raamat1 = new riiul.Raamat(); %>
```

Bean omadustele juurdepääs

- Formaat

```
<jsp:getProperty name="name" property="property" />
```

- Eesmärk
  - Omadustele juurdepääs (pöördumine `getXxx()` meetodi poole) ilma Java koodita
- Märkus:

```
<jsp:getProperty name="raamat1" property="pealkiri" />
```

on ekvivalentne JSP avaldisega

```
<%= raamat1.getPealkiri() %>
```

Bean omaduste seadmine

- Formaat

```
<jsp:setProperty name="name"
                 property="property"
                 value="value" />
```

- Eesmärk
  - Seada Bean omaduste väärtusi (st. pöördumine **setXxx**() poole) ilma Java otsese programmeerimiseta
- Märkused

```
<jsp:setProperty name="raamat1"
                 property="pealkiri"
                 value="Beginning JSP" />
```

on ekvivalentne

```
<% raamat1.setPealkiri("Beginning JSP"); %>
```

- Kuid `<jsp:useBean` –il kaks lisaeelist:
  - Päringu parameetritest on lihtsam objekte kätte saada
  - Lihtsam objektide jagamine lehtede ja servlettide vahel

foo.Person
public String getName() public void setName(String name)

Servleti `doPost( ..)` meetodis

```
foo.Person p = new foo.Person();
p.setName("Tom");
request.setAttribute("person", p);
```

JSP kood:

```
<%= ((foo.Person) request.getAttribute("person")).getName() %>
```

Kasutades JavaBean-i

```
<jsp:useBean id = "person" class = "foo.Person" scope =
request/>
<jsp:getProperty name = "person" property = "name " />
```

Näide

StringBean.java

```
package riiul;

public class StringBean {

   private String message = "Teateid pole";

   public String getMessage() {
      return(message);
   }

   public void setMessage(String message) {
      this.message = message;
   }
}
```

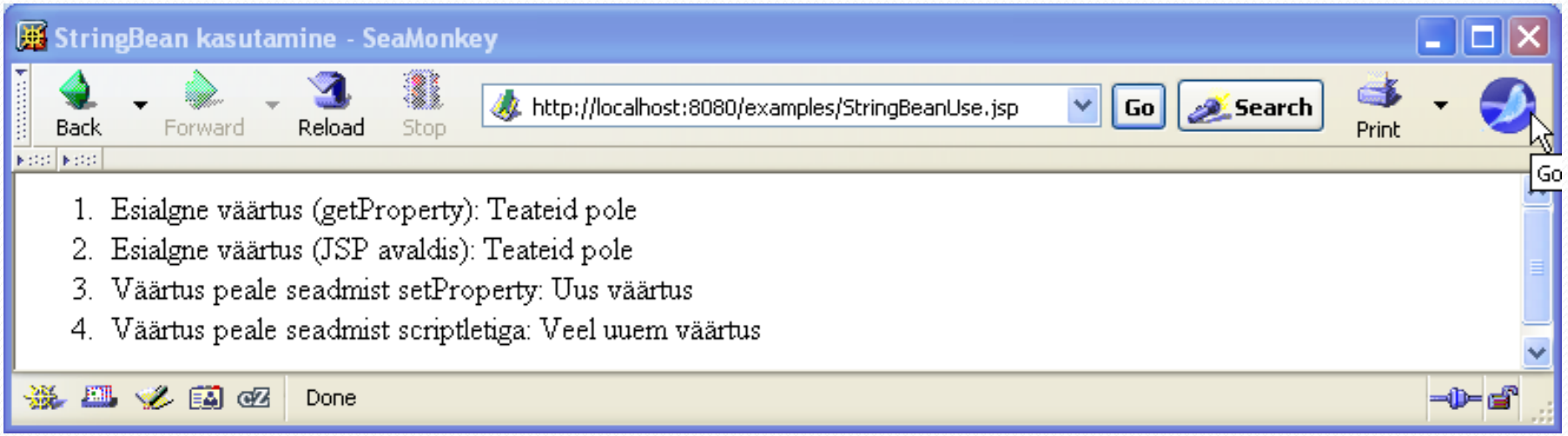
- Paigutada kataloogi
  - *Tomcat_install_dir*\webapps\example\WEB-INF\classes\riiul

- Bean'id (ja abiklassid) peavad *alati* olema paketis!

JSP leht, mis kasutab `StringBean`

```
<ol>
<jsp:useBean id="stringBean" class="riiul.StringBean" />
<li>Esialgne v&auml;&auml;rtus (getProperty):
    <jsp:getProperty name="stringBean"
                     property="message" /></li>
<li> Esialgne v&auml;&auml;rtus (JSP avaldis):
    <%= stringBean.getMessage() %></li>
<li><jsp:setProperty name="stringBean"
                     property="message"
                     value="Uus v&auml;&auml;rtus" />
    V&auml;&auml;rtus peale seadmist setProperty:
    <jsp:getProperty name="stringBean"
                     property="message" /></li>
<li><% stringBean.setMessage("Veel uuem v&auml;&auml;rtus");%>
 V&auml;&auml;rtus peale seadmist scriptletiga:
    <%= stringBean.getMessage() %></li>
 </ol>
```

JSP leht, mis kasutab `StringBean`



Bean omaduste seadmine 1: Tüübiteisendus ja omistamine

Failid: SaleEntry.java
SaleEntry1.jsp

```
<!DOCTYPE ...>
...
<jsp:useBean id="entry" class="sale.SaleEntry" />

<%-- setItemID ootab String'i --%>
<jsp:setProperty
    name="entry"
    property="itemID"
    value='<%= request.getParameter("itemID") %>' />
```

sale.SaleEntry
private String itemId - set private double discountCode - get, set private int numItems - get, set) private double itemCost - get private double totalCost - get

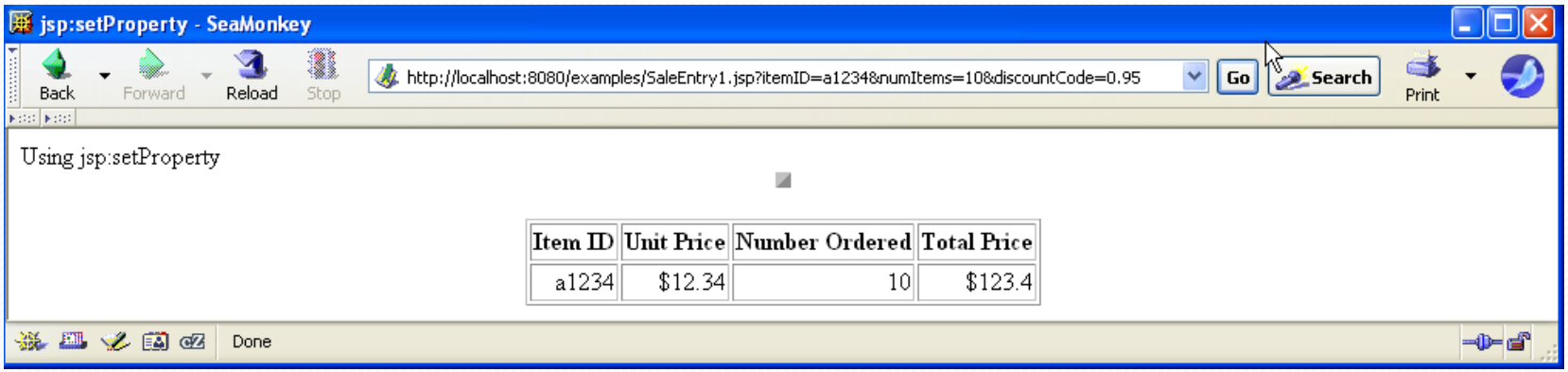
Bean omaduste seadmine 1: Tüübiteisendus ja omistamine

```
<%
int numItemsOrdered = 1;
try {
  numItemsOrdered =
    Integer.parseInt(request.getParameter("numItems"));
} catch(NumberFormatException nfe) {}
%>
<%-- setNumItems ootab int väärtust --%>
<jsp:setProperty
    name="entry"
    property="numItems"
    value="<%= numItemsOrdered %>" />
```

Bean omaduste seadmine
1: Tüübiteisendus ja omistamine

```
<%
double discountCode = 1.0;
try {
  String discountString =
    request.getParameter("discountCode");
  discountCode = Double.parseDouble(discountString);
} catch(NumberFormatException nfe) {}
%>
<%-- setDiscountCode ootab double väärtust --%>
<jsp:setProperty
    name="entry"
    property="discountCode"
    value="<%= discountCode %>" />
```

Bean omaduste seadmine
1: Tüübiteisendus ja omistamine



http://localhost:8080/examples/SaleEntry1.jsp?**itemID**=a1234&**numItems**=10&**discountCode**=0.95

2: Omaduste sidumine päringu parameetritega

- Kasutada `jsp:setProperty` atribuuti `param`
  - Väärtus peab tulema päringu parameetrist
  - Automaatne tüübiteisendus
    - `boolean, Boolean, byte, Byte, char, Character, double, Double, int, Integer, float, Float, long, Long.`

2: Omaduste sidumine päringu parameetritega

```
<jsp:useBean id="entry"
             class= "sale.SaleEntry" />
<jsp:setProperty
    name="entry"
    property="itemID"
    param="itemID" />
<jsp:setProperty
    name="entry"
    property="numItems"
    param="numItems" />
<jsp:setProperty
    name="entry"
    property="discountCode"
    param="discountCode" />
```

2: Omaduste sidumine päringu kõikide parameetritega

- Kasutada `jsp:setProperty` sees `"*"`
  - Väärtused tulevad päringu kõikidest parameetritest, mille nimed peavad sobima omaduste nimedega

```
<jsp:useBean id="entry"
             class="sale.SaleEntry" />
<jsp:setProperty name="entry" property="*" />
```

Kõik nimed peavad täpselt kokku langema!

Bean-ide jagamine

- Atribuut `scope`
  - Seotud meetodi `_jspService()` lokaalse muutujaga
  - `<jsp:useBean id="…" class="…"`
    `scope="…" />`
- Skoop võimaldab mitmel servletil või JSP lehel jagada andmeid
- Atribuudil scope neli võimalikku väärtust:

`page, request, session, application`

Atribuudi `scope` väärtused

- **`page`** -
- `<jsp:useBean … scope="page"/>` või
  `<jsp:useBean…>`
  - Vaikimisi väärtus. Bean pannakse objekti `PageContext`. Juurdepääs `<jsp:getProperty.. <jsp:setProperty..`
- **`application`**
  `<jsp:useBean … scope="application"/>`
  - Bean pannakse objekti `ServletContext`. Juurdepääs on muutuja `application` kaudu või `getServletContext().getAttribute(..)`. Objekti `ServletContext` jagatakse rakenduse kõigi servlettide vahel.

Atribuudi `scope` väärtused

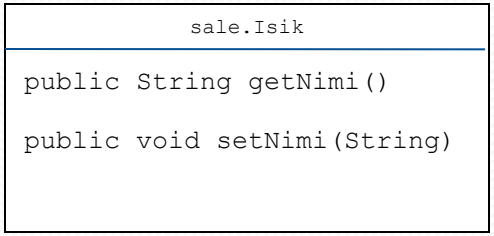
- **`session`**

  `<jsp:useBean … scope="session"/>`

  - Bean pannakse objekti `HttpSession` , juurdepääs meetoditega `getAttribute()` ja `setAttribute()`
- **`request`**

  `<jsp:useBean … scope="request"/>`

  - Bean pannakse objekti `ServletRequest()` päringu ajaks, kättesaadav meetodiga `getAttribute()`



Meil on näiteks servleti kood:

```
public void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException,
ServletException {
  sale.Isik p = new sale.Isik();
  p.setNimi(”Mart”);
  request.setAttribute(”isik”, p);
  RequestDispatcher view =
       request.getRequestDispatcher(“tulem.jsp“);
  view.forward(request, response);
}
```

JSP kood skriptimisega

```
<html><body>
Isik:
<%= ((sale.Isik) request.getAttribute("isik")).getNimi() %>
</body></html>
```

JSP kood märgistega:

```
<html><body>
<jsp:useBean id = "isik" class="sale.Isik" scope="request"/ >
Isik:
<jsp:getProperty name="isik"  property = "nimi"/>
</body></html>
```

Tulemuseks: Isik: Mart

Bean-i tingimuslik loomine

- `jsp:useBean` tulemusena luuakse Bean uus isend siis ja ainult siis, kui sama `id` ja skoobiga Bean-klassi isendit ei ole.
- Kui sama `id` ja skoobiga Bean on olemas, siis antakse viit sellele tagasi.
- Kui

```
<jsp:useBean ...>statements</jsp:useBean>
```

siis `statements` (`jsp:setProperty` ...) täidetakse ainult siis, kui luuakse uus Bean isend.

Bean-i tingimuslik loomine

```
public class AccessCountBean {
  private String firstPage;
  private int accessCount = 0;

  public String getFirstPage() {
    return(firstPage);
  }
  public void setFirstPage(String firstPage) {
    this.firstPage = firstPage;
  }
  public int getAccessCount() {
    return(accessCount);
  }
  public void setAccessCount (int a) {
    accessCount += a;
  }
}
```

Bean-i tingimuslik loomine:

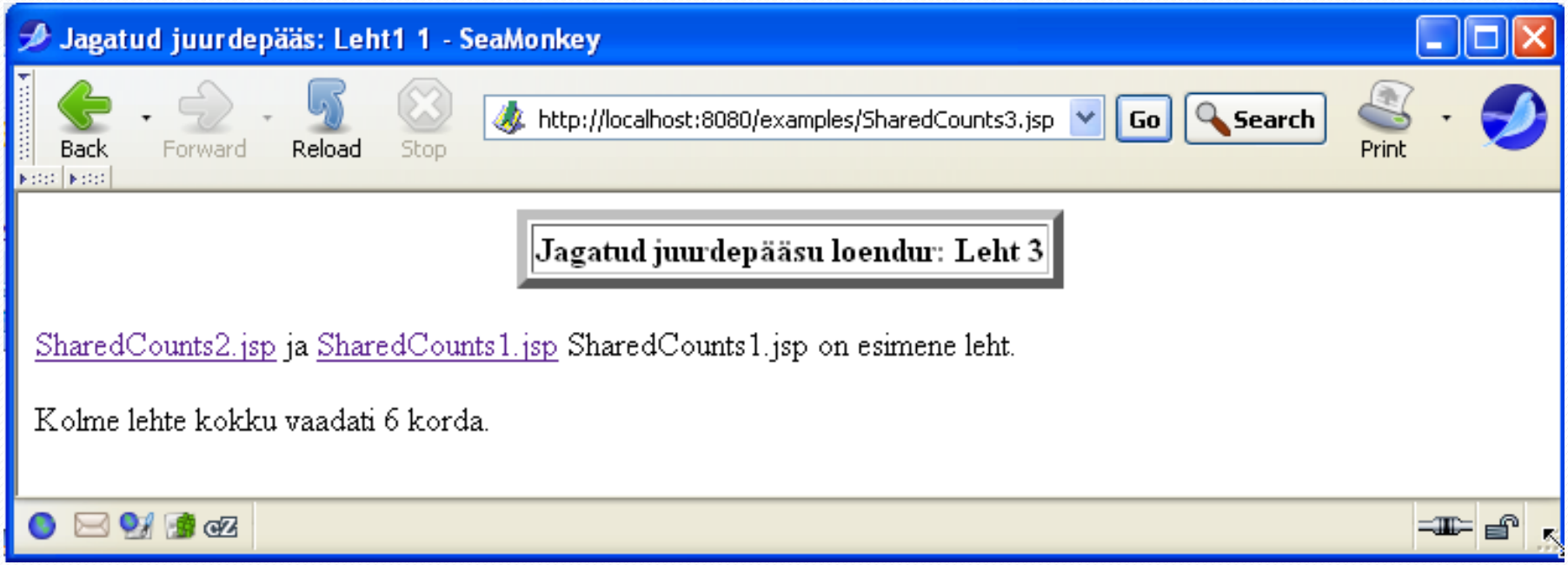
application

SharedCounts1.jsp
SharedCounts2.jsp
SharedCounts3.jsp

```
<jsp:useBean id="counter"
             class="sale.AccessCountBean"
             scope="application">
  <jsp:setProperty name="counter"
                   property="firstPage"
                   value= "SharedCounts1.jsp" />
</jsp:useBean>
<jsp:getProperty name="counter" property="firstPage" />
avati.
<jsp:setProperty name="counter" property="accessCount"
  value="1"/>
<a href="SharedCounts2.jsp">SharedCounts2.jsp</a> ja
<a href="SharedCounts3.jsp">SharedCounts3.jsp</a>
<p>Kolme lehte kokku vaadati kokku
   <jsp:getProperty name="counter" property="accessCount" />
korda.
</p>
```

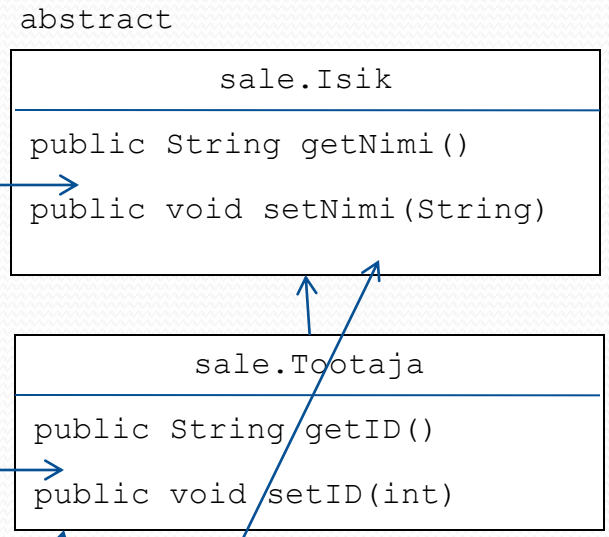
DEMO

SharedCounts1.jsp
SharedCounts2.jsp
SharedCounts3.jsp



Olgu meil vorm:

```
<html><body>
<form action="TestBean.jsp">
Nimi: <input type = "text" name="nimi"/>
ID: <input type = "text" name="ID"/>
<input type = "submit" />
</form>
</html></body>
```



JSP kood

TestBean.jsp

```
<html><body>
<jsp:useBean id ="isik" type = "sale.Tootaja"  class ="sale.Tootaja">
   <jsp:setProperty name= "isik" property = "*"/>
</jsp:useBean>
Nimi: <jsp:getProperty name= "isik" property = "nimi"/>
ID: <jsp:getProperty name= "isik" property = "ID"/>
</html></body>
```

Olgu meil omadus primitiivist erinev :

Servleti kood:

```
public void doGet(HttpServletRequest  request,
HttpServletResponse    response) throws
        IOException,   ServletException {
  sale.Isik p = new sale.Isik();
  p.setNimi("Mart");
  sale.Koer koer = new sale.Koer();
  koer.setNimi("Reks");
  p.setKoer(koer);
  request.setAttribute("isik", p);
  RequestDispatcher view =
        request.getRequestDispatcher("tulem.jsp");
  view.forward(request, response);
}
```

sale.Isik
public String getNimi()
public void setNimi(String)
public Koer getKoer()
public void setKoer(Koer)

sale.Koer
public String getNimi()
public void setNimi(String)

JSP kood skriptiga

tulem.jsp

```
<html><body>
 Koera nimi:
<%= ((sale.Isik) request.getAttribute("isik")).getKoer().getNimi() %>
</html></body>
```

JSP kood EL-ga

EL

```
<html><body>
Koera nimi: ${isik.koer.nimi}
</html></body>
```

Viit muutujale **`isik`** lahendatakse kahel viisil:

- Kui muutuja nimi on identne ühega EL valmisobjektidest (*implicit objects*) , siis muutujale omistatakse viit vastavale objektile.
- Kui muutuja nimi ei sobi ühegagi EL ilmutamata objektidest, siis otsitakse nime JSP valmisobjektide atribuutide seast selles järjekorras:
`page, request, sessioon, application.`

EL valmisobjekt	**Esindab**
`pageContext`	`JSP ilmutamata objektide konteiner`
`pageScope`	`Väärtus on kättesaadav` *`page.getAttribute()`*
`requestScope`	*`request.getAttribute()`*
`sessionScope`	*`session.getAttribute()`*
`applicationScope`	*`application.getAttribute()`*
`param`	*`request.getParam()`*
`paramValues`	*`request.getParamValues()`*
`header`	*`request.getHeader()`*
`headerValues`	*`request.getHeaderValues()`*
`cookie`	*`request.getCookies()`*
`initParam`	*`application.getInitParameter()`*

Vahekokkuvõte

- `jsp:useBean` eelised
  - Peidab Java süntaksi
  - Lihtsustab päringu parameetrite seostamist Java objektidega (Bean properties)
  - Lihtsustab objektide jagamist paljude päringute, servlettide/JSP lehtede vahel
- `jsp:useBean`
  - Loob või tagastab juurdepääsu JavaBean objektile
- `jsp:getProperty`
  - Tagastab Bean omaduse (st. `getXxx`) servleti väljundisse
- `jsp:setProperty`
  - Seab Bean omaduse (st. edastab väärtuse meetodile `setXxx`)

Tavamärgised

Defineerimiseks vajalikud kolm komponenti:

- Märgise käsitleja Java klass (`.java`)
  - Peab rakendama liidest `javax.servlet.jsp.tagext.Tag`
  - Laiendab klassi `TagSupport` või `BodyTagSupport`
  - `.class` fail servlettidega samasse kausta
- Märgiseteegi kirjeldaja (`.tld`)
  - XML fail, mis kirjeldab märgise nime, atribuudid ja märgise käsitleja klassi - `.jsp` failidega samasse kausta
- JSP fail (`.jsp`)
  - Impordib märgiseteegi
  - Defineerib märgise prefiksi
  - Kasutab märgiseid

Märgise käsitleja Java klass - Näide

ExampleTag1.java

```
package sale.tags;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;

public class ExampleTag1 extends TagSupport {
public int doStartTag() {
try {
  JspWriter out = pageContext.getOut();
  out.print("Lihtne tavam&auml;rgis");
} catch(IOException ioe) {
System.out.println("Viga m&auml;rgises ExampleTag: " + ioe);
}
return(SKIP_BODY);
}
}
```

Märgiseteegi kirjeldaja - Näide Example1.tld

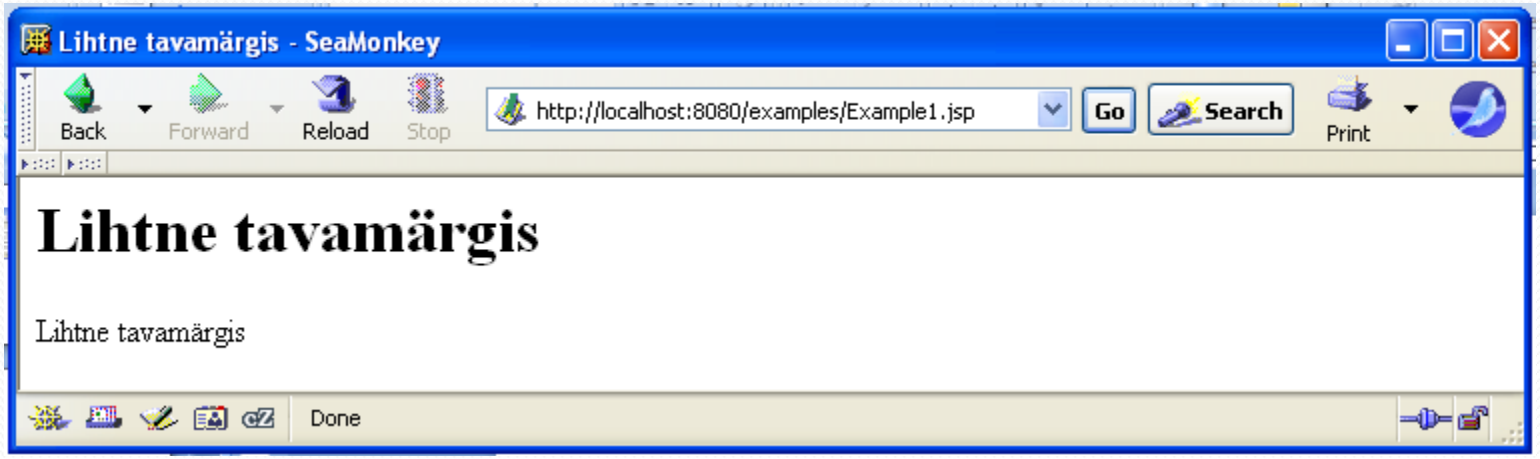
```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library
1.1//EN"
"http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">
<taglib>
  <tlibversion>1.0</tlibversion>
  <jspversion>1.1</jspversion>
  <shortname>lyhinimi</shortname>
  <urn></urn>
  <info>Näide TLD failist
  </info>
  <tag>
    <name>example</name>
    <tagclass>sale.tags.ExampleTag1</tagclass>
    <info>Lihtne näide: lisab teksti väljundisse</info>
    <bodycontent>EMPTY</bodycontent>
  </tag>
</taglib>
```

Tavamärgised: JSP fail - Näide

Example1.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
  <HEAD>
  <%@ taglib uri="Example1.tld" prefix="ex" %>
  <TITLE><ex:example /></TITLE>
  </HEAD>
  <BODY>
    <H1><ex:example /></H1>
    <ex:example />
  </BODY>
</HTML>
```

Tavamärgised: JSP fail - Näide



Tavamärgised: märgiseteegi kirjeldaja (`.tld`)

- Alguses XML päis ja `DOCTYPE`
- `taglib`
- Iga märgis kirjeldatakse tag elemendiga:
  - `name` - märgise nimi, meie näites

    ```
    <name>example</name>
    ```

  - `tagclass` – vastav Java klass, meie näites

    ```
    <tagclass>sale.tags.ExampleTag</tagclass>
    ```

  - `info` - kirjeldus, meie näites

```
<info>Lihtne näide: lisab teksti väljundisse</info>
```

Märgise kasutamine JSP failides

- Importida märgiste teek
  - TLD faili asukoha määramine

```
<%@ taglib uri="Example1.tld" prefix="ex" %>
```

  - Prefiksi (nimeruumi) defineerimine

```
<%@ taglib uri="Example1.tld" prefix="ex" %>
```

- Märgise kasutamine

```
<prefix:tagName />
```

  - Märgise nimi TLD failist
  - Prefiks tuleneb direktiivist `taglib`
  - Nt `<ex:example />`

Märgise atribuudid (.jsp)

Võib kasutada ka atribuute

```
<prefiks:name
      attribute1="value1"
      attribute2="value2"
      ...
      attributeN="valueN"
   />
```

Märgise atribuudid

- Atribuudi `attribute1` olemasolu kutsub esile meetodi `setAttribute1` poole pöördumise
  - Väärtus antakse edasi `String` kujul
- Näiteks,
  - Kui

```
<prefix:tagName attribute1="Test" />
```

siis lisada Java klassi meetod

```
public void setAttribute1(String value1) {
    doSomethingWith(value1);
}
```

Atribuudid märgiseteegi kirjeldajas (`.tld`)

- Element `tag` peab sisaldama märgist `attribute`
- Element `attribute` sisaldab kolme märgist
  - **`name`** – nimi `<name>length</name>`
  - **`required`** – kas atribuut peab olema
  - `<required>false</required>`
  - **`rtexprvalue`** – kas väärtus võib olla JSP avaldis
    `<%= expression %>`

  Vaikimisi `false`, st väärtus peab olema `String`.

Märgis koos sisuga

- Lihtsaim märgis

```
<prefix:tagName />
```

- Märgis koos atribuutidega

```
<prefix:tagName att1="val1" att2="val2" ... />
```

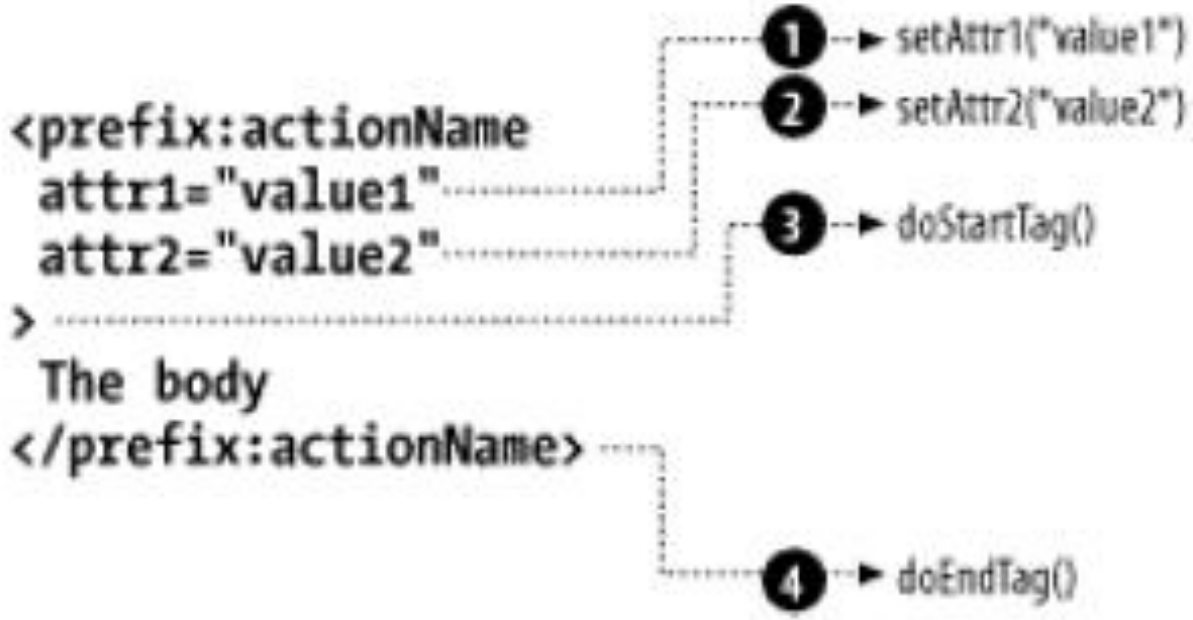
- Märgis koos sisuga

```
<prefix:tagName>
  JSP sisu
  </prefix:tagName>

<prefix:tagName att1="val1" ... >
  JSP sisu
  </prefix:tagName>
```

Märgisekäsitleja (Java) klass

- `doStartTag`
  - Tagastab tavaliselt `EVAL_BODY_INCLUDE` mõnikord `SKIP_BODY`
- `doEndTag`
  - Pöördutakse peale sisu (`body`) töötlemist
  - Tagastab `EVAL_PAGE`



Näide

HeadingTag.java

```
package sale.tags;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;
public class HeadingTag extends TagSupport {
  private String bgColor;
  private String color = null;
  public void setBgColor(String bgColor) {
    this.bgColor = bgColor;
  }
  public void setColor(String color) {
    this.color = color;
  }
```

Näide jätkub

HeadingTag.java

```
public int doStartTag() {
   try {
     JspWriter out = pageContext.getOut();
     out.print("<TABLE BORDER=" + border +
               " BGCOLOR=\"" + bgColor + "\"" +
               " ALIGN=\"" + align + "\"");
     if (width != null) {
       out.print(" WIDTH=\"" + width + "\"");
     }
    out.print("><TR><TH>");
    out.print("<SPAN STYLE=\"" +
     ...
   } catch(IOException ioe) {
     System.out.println("Error in HeadingTag: " + ioe);
   }
   return(EVAL_BODY_INCLUDE); // Include tag body
 }
```

Näide jätkub

HeadingTag.java

```
public int doEndTag() {
  try {
    JspWriter out = pageContext.getOut();
    out.print("</SPAN></TABLE>");
  } catch(IOException ioe) {
    System.out.println("Viga failis HeadingTag: " + ioe);
  }
  return(EVAL_PAGE); // Continue with rest of JSP page
}
```

Näide jätkub

HeadingTag.tld

```
…<taglib>
  <shortname>lyhi</shortname>
  <urn></urn>
  <info>Märgiseteek pealkirjade jaoks</info>
    <tag>
      <name>heading</name>
      <tagclass>sale.tags.HeadingTag</tagclass>
      <info>Väljastab üheelemendilise tabeli pealkirjana</info>
      <bodycontent>JSP</bodycontent>
      <attribute>
        <name>bgColor</name>
        <required>true</required>
      </attribute>
      <attribute>
        <name>color</name>
        <required>false</required>
      </attribute> …
```

Näide jätkub

HeadingTag.jsp

```
<%@ taglib uri="HeadingTag.tld" prefix="hh" %>
<hh:heading bgColor="#C0C0C0">
  Vaikepealdis
</hh:heading>
<p>
<hh:heading bgColor="BLACK" color="WHITE">
  Valge mustal
</hh:heading> </p>
<p>
<hh:heading bgColor="#EF8429" fontSize="60" border="5">
  Suur värviline
</hh:heading> </p>
<p>
<hh:heading bgColor="CYAN" width="100%">
  Laia taustaga
</hh:heading> </p>
...
```


Märgiste abiga genereeritud pealdised - SeaMonkey
Back
Forward
Reload
Stop
http://localhost:8080/examples/HeadingTag.jsp
Go
Search
Print
Vaikepealdis
Valge mustal
Suur värviline
Laia taustaga
Mittestandardse fondiga
Done

Märgise sisu valikuline täitmine

- Kui ei taha sisu täita, siis

  `doStartTag` tagastab `SKIP_BODY`

- Kui sisu ka täita, siis

  `doStartTag` tagastab `EVAL_BODY_INCLUDE`

- Saab otsustada ka päringu ajal

  - Päringu objekti saab kätte:

```
public int doStartTag() {
ServletRequest request = pageContext.getRequest();
//otsustada päringu põhjal, mida tagastada…
```

Tavamärgiste ja koodielementide võrdlus

- Tavamärgistel on lihtsam süntaks; koodielemendid on kirjutatud Javas
- Lihtsam siluda, vähem vigu
- Võimaldavad vältida Javat JSP lehel

Tavamärgis vs Bean

- Bean'id ei saa käsitleda JSP sisu, märgised saavad
- Keerulisi tegevusi saab märgistega edukamalt lihtsustada
- Märgiste defineerimine töömahukam

Avatud koodiga märgisteteegid http://jakarta.apache.org/taglibs/

JSP Standard Tag Library (JSTL)

- Formaadid (I18N)
- Andmebaasidele juurdepääs
- E-meili saatmine
- Kuupäev/kellaaeg
- Vormi väljade valideerimine
- Perl regulaaravaldised
- Andmete eraldamine teistelt veebilehtedelt
- XSL transformatsioonid
- …

Mõned JSTL tegevused `core` teegist

Tegevus	**Eesmärk**
`set`	muutujale väärtuse omistamine, muutuja loomine;
`remove`	muutuja kustutamine;
`out`	andmete kirjutamine ilmutamata objekti `out`, escaping XML special characters;
`url`	URL loomine koos päringusõnega;
`if`	tingimuslause (`if-then`);
`choose`	tingimus (valiku)lause (`if-then-elseif`);
`forEach`	tsükkel üle kollektsiooni elementide.

<c:set >

```
<c:set var="visits" scope="application" value="${visits+1}"/>
```

`var` – lokaalne (skoobiga) muutuja

Skoobid: `page, request, session, application`
(vaikimisi skoop – `page`)

- Tegevusega `set` saab omistada väärtuse ühele kolmest muutuja tüübist: skoobiga muutuja, `java.util.Map` element, `JavaBeans` objekti omadus.

- Kui muutujat pole olemas, siis ta luuakse. `Java.util.Map` koosneb võti-väärtus (`key-value`) paaridest.

<c:set >

- Atribuut `target` määrab, millise skoobiga Map objekti elemente tahetakse muuta.

```
<c:set target="${applicationScope}" property="visits"
       value="${visits+1}"/>
```

- Kui avaldises määratud võtit ei eksisteeri, siis see luuakse automaatselt.

<c:remove >

- Tegevusel `remove` on kaks atribuuti: *var* ja *scope*.
- Ta eemaldab skoobiga muutuja antud skoobiga objektist.
- Kui skoopi pole antud, siis muutuja otsimise järjekord objektidest on järgmine: *page, request, session, application*.
- Muutuja eemaldatakse esimesest leitud skoobist.

```
<c:remove var="arv" scope="session"/>
```

<c:out >

- Tegevusel `out` on atribuut `value`, mille väärtus kirjutatakse JSP ilmutamata objekti `out`.

- Vaikimisi XML erimärgid (`< > & " '`) asendatakse vastavate koodidega (are escaped).

- Kui väärtus on `null`, siis kirjutatakse tühi sõne (mis on vaikimisi väärtus).

- Vaikimisi väärtust saab muuta atribuudiga `default`.

```
<c:set var="customerTable" scope="application">
<table border="1">
    <c:forEach var="customer" items="${customers}">
    <tr>
  <td>${customer.lastName}</td>
  <td><c:out value="${customer.address}" default="no address
    specified"/></td>
  <td>
    <c:out value="${customer.address}">
  <font color="red">no address specified</font>
</c:out>
      </td>
</tr>
  </c:forEach>
</table>
</c:set>
```

The content between <c:set> and /c:set is saved as ${customerTable}.

```
<c:out value="${customerTable}" escapeXml="false"/>
```

<c:url >

- Tegevuse `url` ainsaks nõutud atribuudiks on `value`, mille väärtuseks võib olla kas absoluutne või relatiivne URL.

- Tegevus võib sisaldada ka parameetreid, mis lisatakse päringusõnena

- Olgu meil rakenduse teeks `/minuRakendus`, siis JSP märgistega

```
<c:url value="/ida"/>
  <c:param name="kasutajanimi" value="Ken Som"/>
<c:url/>
```

saadetakse objekti `out` sõne

`/minuRakendus/ida?kasutajanimi=Ken+Som`.

<c:if >

Tegevus `if` sisaldab atribuuti `test`:

```
<c:if test="${empty visits}">
  <c:set var="visits" scope="application" value="0"/>
</c:if>
```

Muutujale `visits` (skoobiga `application`) omistatakse `0` kui `visits` ei ole olemas üheski skoobis.

Teine variant:
Tõeväärtus omistatakse muutujale, mis on määratud atribuudiga `var`:

```
<c:if test="${visits gt 3}" var="testResult">
 See tekst läheb väljundisse kui tingimus toene.
</c:if>
```

<c:choose>

Tegevusel `if` ei ole `else-varianti`, selleks on tegevus `choose`, millel on alamtegevused `when` ja `otherwise`.

Näide tegevuse `choose` kasutamisest:

```
<c:choose>
  <c:when test="${param.gender == 'f'}">
      Female
  </c:when>
  <c:otherwise>
      Male
  </c:otherwise>
</c:choose>
```

<c:forEach>

JSTL põhiliseks iteraatoriks on tegevus forEach. Süntaksit iseloomustab järgmine näide:

```
<c:forEach var = "i" begin="2" end="8" step="2" >
  ${i}
</c:forEach>
```

Teine variant: itereerimine üle andmestruktuuride (massiivid või suvalised klassid, mis rakendavad liideseid java.util Map, Collection, Iterator, Enumeration.

```
<ul>
  <c:forEach var="aHeader" items="${header}">
    <li><c:out value="${aHeader}" /></li>
  </c:forEach>
</ul>
```