MVC

Internet programiranje
IV godina, ETF Banjaluka

MVC

- MVC – arhitektura koja predviđa postojanje odvojenih komponenata zaduženih za prezentacionu i poslovnu logiku sistema
- MVC pattern je arhitektura aplikacije koja se sastoji iz 3 dijela:
  - Model – predstavlja podatke (data) i poslovna (business) pravila koja se upotrebljavaju za manipulaciju podacima – sloj sistema "najbliži" bazi podataka
  - View – prezentacioni sloj koji je zadužen za omogućavanje interakcije krajnjeg korisnika sa ostatkom sistema – sastoji se od elemenata korisničkog interfejsa, npr. tekstualnih polja, padajućih menija – ne postoji jednoznačno mapiranje između Model i View slojeva, jer je broj „pogleda" prema modelu podataka neograničen i zavisi jedino od potreba korisničkog interfejsa
  - Controller – kontroliše interakciju između Model-a i View-a na bazi korisničkih događaja (miš, tastatura) – sve promjene na pojedinim pogledima su inicirane od strane kontrolera u zavisnosti od trenutnog konteksta aplikacije i odgovarajućih podaka kojima model raspolaže

MVC – implementacija

- jednostavna implementacija pomoću JSP i Servleta
- Model – kolekcija Java klasa (bean-ova) koje su namijenjene za manipulaciju podacima – ovakva klasa može da radi sa bilo kojim korisničkim interfejsom (konzola, GUI interfejs, Web aplikacija)
- View – predstavljen JSP stranom – podaci se prenose pomoću HttpServletRequest, HttpSession i ServletContext-a – čita podatke iz bean-ova koristeći jsp:useBean
- Controller – Servlet komunicira sa modelom, popunjava HttpServletRequest, HttpSession i ServletContext odgovarajućim podacima (u formi bean-ova) i prosljeđuje ih view-u (JSP stranama) koristeći RequestDispatcher (metodu forward)
- prednosti:
  - Controller komunicira sa modelom, ali ne mora da generiše HTML ili XHTML kod
  - JSP ne komunicira sa modelom, jer podatke dobija od Controller-a, tako da JSP služi samo za kreiranje HTML ili XHTML koda

MVC – frameworks

- .NET
  - ASP.NET MVC
  - ASP.NET WEBAPI
  - Maverick.NET
  - Monorail
  - ProMesh.NET Open source MVC Web Application Framework for .NET 2.0 ili viši
  - PureMVC Framework za C#
- Java
  - Aranea
  - Cocoon
  - Grails
  - JSF
  - Oracle Application Framework
  - Spring MVC Framework
  - Struts
  - Stripes
  - Tapestry
  - WebObjects
  - WebWork
  - Wicket
  - PureMVC Framework for Java
  - LongJump
  - Sofia
  - Struts2

MVC – frameworks

- Python
  - Django
  - Enthought
  - Pylons
  - TurboGears
  - web2py
  - Zope
  - PureMVC
- Ruby
  - Camping
  - Merb
  - Nitro
  - Ramaze
  - Ruby on Rails
  - Monkeybars

MVC – frameworks

- PHP
  - CakePHP
  - CodeIgniter
  - Concrete5
  - Drupal
  - FUSE
  - Joomla – open source Content Management System koji koristi MVC model za svoje ekstenzije – komponente i module
  - Kohona
  - Kumbia
  - LISA MVC
  - MVCnPHP
  - Nette
  - Odin
  - Orinoco
  - PHPonTrax
  - Prado
  - Solar
  - Symfony
  - Zend
  - ZNF
  - Zoop
  - SimpleTools
  - PureMVC

JSP Model 2

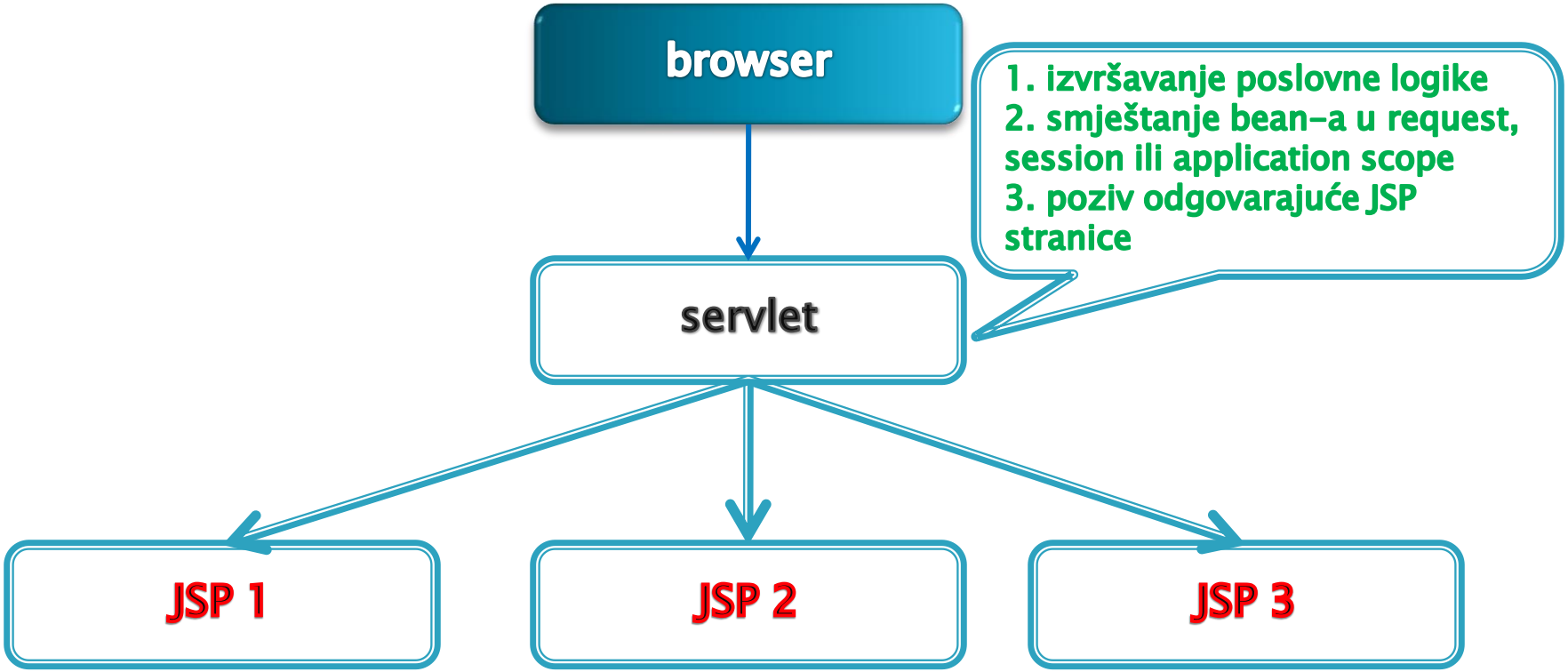
JSP Model 2

- Servleti rade dobro u sljedećim slučajevima:
  - kada je izlaz binaran – npr. slika.
  - kada ne postoji izlaz – redirekcija, prosljeđivanje...
- JSP radi dobro u sljedećim slučajevima:
  - kada izlaz uglavnom čine tekstualni podaci – npr. HTML
  - kada je format/izgled stranice uglavnom nepromjenljiv
- MVC arhitektura potrebna kada:
  - isti zahtjev rezultira različitim rezultatima
  - je razvojni tim mnogobrojan, pri čemu su različiti članovi tima zaduženi za web segment i segment poslovne logike
  - kada se izvršava komplikovano procesiranje, a izgled je relativno fiksan

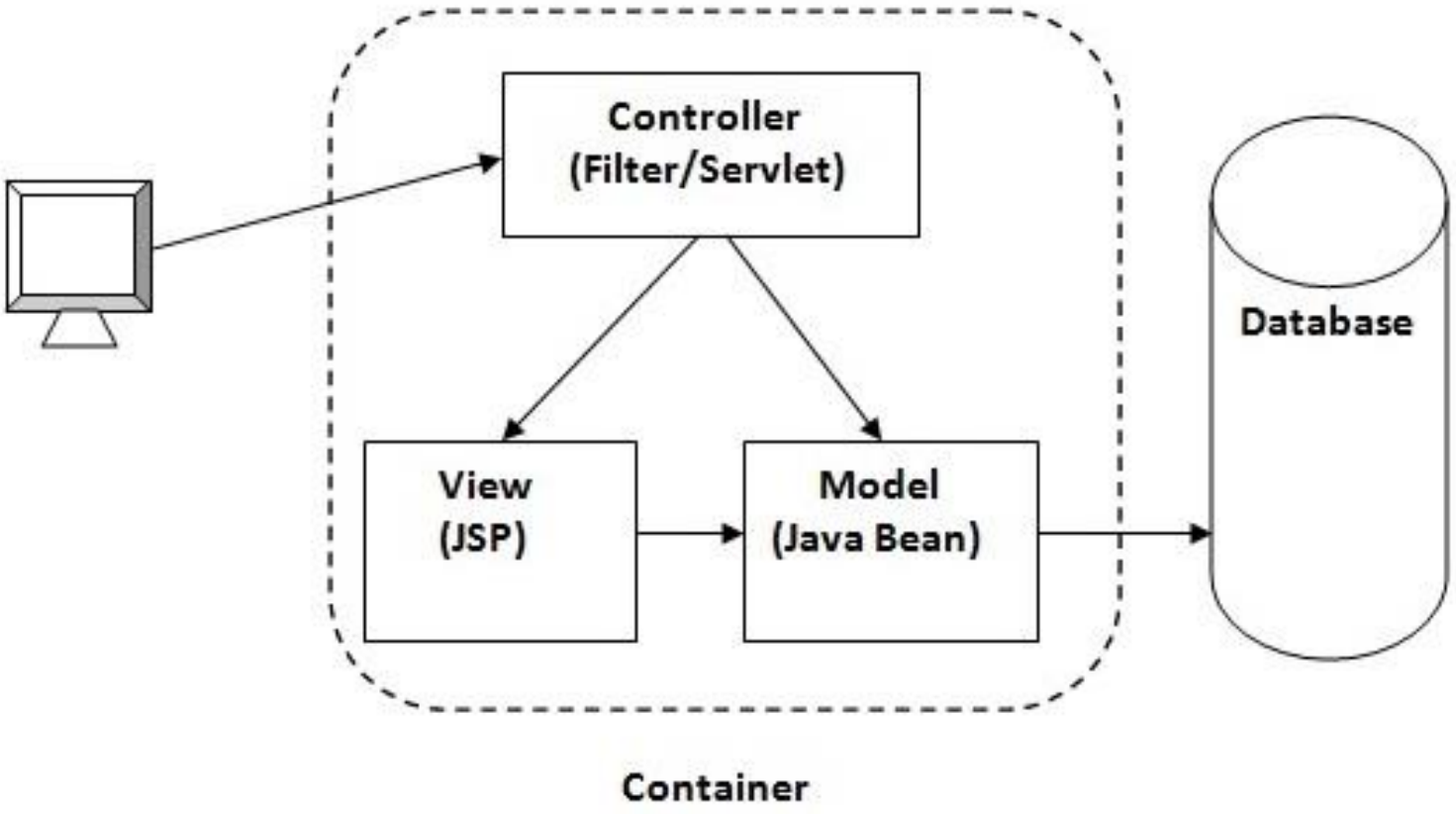
JSP Model 2

- Moguće je koristiti određeni framework – ponekad je upotreba frameworka korisna
  - Struts
  - JavaServer Faces (JSF)
- Njihovo korištenje nije obavezno!
- MVC je moguće implementirati pomoću RequestDispatcher-a i dobiti sasvim zadovoljavajuće rezultate za jednostavne i srednje kompleksne aplikacije
- MVC u potpunosti mijenja dizajn sistema
- naziva se i Model 2 pristup

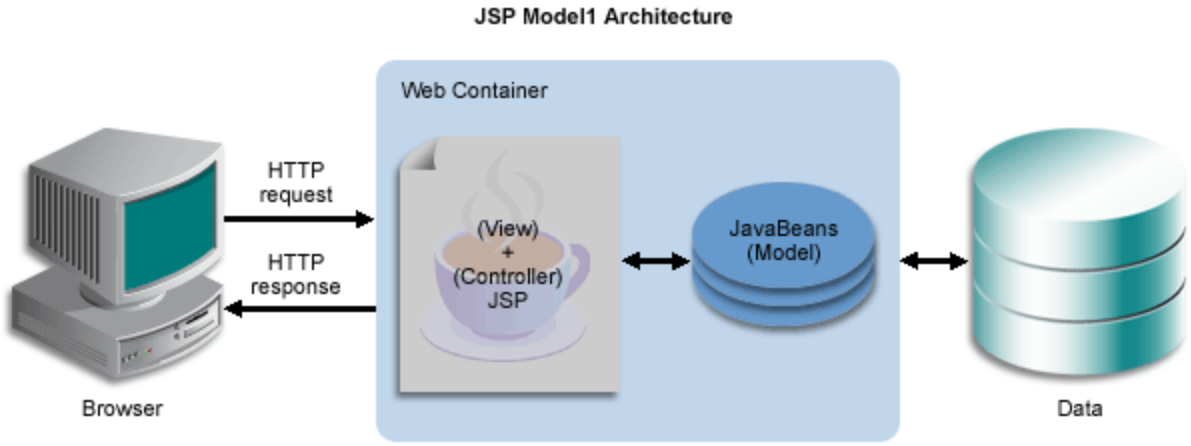
JSP Model 2

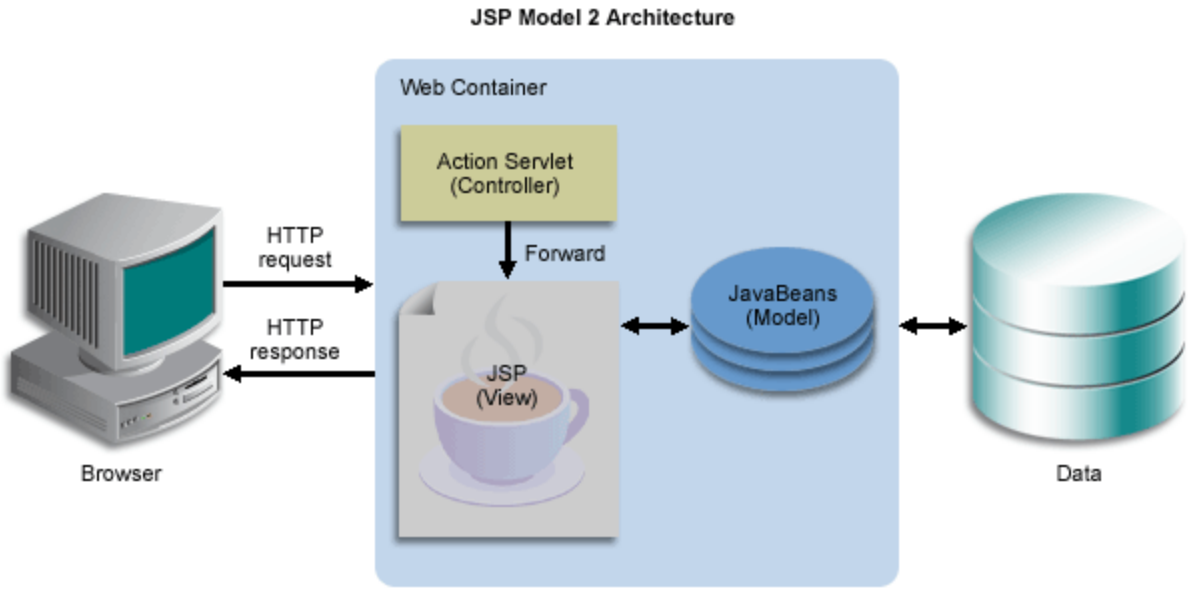

browser
1. izvršavanje poslovne logike
2. smještanje bean-a u request, session ili application scope
3. poziv odgovarajuće JSP stranice
servlet
JSP 1
JSP 2
JSP 3

JSP Model 2


Controller
(Filter/Servlet)
View
(JSP)
Model
(Java Bean)
Database
Container

JSP Model 2


JSP Model1 Architecture
Web Container
HTTP request
HTTP response
(View) + (Controller) JSP
JavaBeans (Model)
Browser
Data


JSP Model 2 Architecture
Web Container
Action Servlet (Controller)
Forward
HTTP request
HTTP response
JSP (View)
JavaBeans (Model)
Browser
Data

JSP Model 2

- 1. Definisati beanove da bi prezentovali podatke
- 2. Koristiti servlet da bi se prihvatio zahtjev
  - Servlet čita parametre zahtjeva, provjerava unijete podatke, itd.
- 3. Postaviti beanove
  - Servlet realizuje poslovnu logiku (kod specifičan za aplikaciju) ili kod za pristup podacima da bi došao do rezultata. Rezultati se smještaju u beanove koji su definisani u okviru koraka 1.
- 4. Postaviti bean u okviru zahtjeva, sesije ili konteksta servleta – servlet poziva setAttribute metodu nad objektima zahtjeva, sesije ili aplikacije (konteksta servleta) da bi smjestio referencu na bean-ove koji predstavljaju rezultat zahtjeva.

JSP Model 2

- 5. Proslijediti zahtjev JSP stranici - navigacija
  - Servlet prepoznaje koja JSP stranica odgovara datoj situaciji i koristi metod forward RequestDispatcher objekta da bi predao kontrolu datoj stranici.
- 6. Preuzeti podatke iz beana.
  - JSP stranica pristupa beanu pomoću jsp:useBean i odgovarajuće opsege važenja iz koraka 4. Stranica tada koristi jsp:getProperty da bi pristupila property-jima bean-a
  - Moguće je i korištenje JSP EL-a
  - JSP stranica ne kreira ili modifikuje bean; ona samo prihvata i pokazuje podatke koje servlet kreira.

JSP Model 2

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

   String operation = request.getParameter("operation");
   if (operation == null) {
        operation = "unknown";
   }
   String address;
   if (operation.equals("order")) {
        address = "/WEB-INF/order.jsp";
   } else if (operation.equals("cancel")) {
        address = "/WEB-INF/cancel.jsp";
   } else {
        address = "/WEB-INF/unknownOperation.jsp";
   }
   RequestDispatcher dispatcher =
   request.getRequestDispatcher(address);
   dispatcher.forward(request, response);
}
```

jsp:useBean u MVC-u

- JSP stranice ne bi trebale da kreiraju objekte
  - Servlet, a ne JSP stranica, trebalo bi da kreira sve objekte koji predstavljaju podatke. Da bi se garantovalo da se u okviru JSP stranice neće kreirati objekti, trebalo bi koristiti

```
<jsp:useBean ... type="package.Class" />
```

  - umjesto

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

- JSP stranica ne bi trebalo da modifikuje objekte
- Potrebno je koristiti jsp:getProperty, ali ne jsp:setProperty

jsp:useBean oblasti važenja

- request
  - <jsp:useBean id="..." type="..." scope="request" />
- session
  - <jsp:useBean id="..." type="..." scope="session" />
- application
  - <jsp:useBean id="..." type="..." scope="application" />
- page
  - <jsp:useBean id="..." type="..." scope="page" />
  - ili samo
  - <jsp:useBean id="..." type="..." />
  - Ova oblast važenja se ne korsiti u okviru MVC (Model 2) arhitekture

request-bazirano dijeljenje podataka

- Servlet

```
ValueObject value = new ValueObject(...);
request.setAttribute("key", value);
RequestDispatcher dispatcher =
request.getRequestDispatcher ("/WEBINF/
SomePage.jsp");
dispatcher.forward(request, response);
```

- JSP 1.2

```
<jsp:useBean id="key"
  type="somePackage.ValueObject"
scope="request" />
<jsp:getProperty name="key"
  property="someProperty" />
```

- JSP 2.0

```
{key.someProperty}
```

session-bazirano dijeljenje podataka

- Servlet

```
ValueObject value = new ValueObject(...);
HttpSession session = request.getSession();
session.setAttribute("key", value);
RequestDispatcher dispatcher =
request.getRequestDispatcher ("/WEBINF/
SomePage.jsp");
dispatcher.forward(request, response);
```

- JSP 1.2

```
<jsp:useBean id="key" type="somePackage.ValueObject"
scope="session" />
<jsp:getProperty name="key" property="someProperty" />
```

- JSP 2.0

```
{key.someProperty}
```

session-bazirano dijeljenje podataka – sendRedirect varijacija

- Moguće je koristiti response.sendRedirect umjesto RequestDispatcher.forward
- Pri korištenju sendRedirect:
  - Korisnik vidi URL JSP stranice (ako se koristi RequestDispatcher.forward korisnik vidi URL servleta)
- Prednosti sendRedirect:
  - Korisnik može da zapamti (bookmark) adresu JSP stranice
- Nedostaci sendRedirect
  - Korisnik može da posjeti JSP stranici bez pristupanja servletu, pa JSP podaci mogu da budu nedostupni
  - U okviru JSP stranice trebalo bi detektovati ovu situaciju

application-bazirano dijeljenje podataka

- Servlet

```
synchronized(this) {
  ValueObject value = new ValueObject(...);
  getServletContext().setAttribute("key", value);
  RequestDispatcher dispatcher
  =request.getRequestDispatcher
  ("/WEB-INF/SomePage.jsp");
  dispatcher.forward(request, response);
}
```

- JSP 1.2

```
<jsp:useBean id="key" type="somePackage.ValueObject"
scope="application" />
<jsp:getProperty name="key" property="someProperty" />
```

- JSP 2.0

```
{key.someProperty}
```

Forward i include

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

- Dozvoljeno, ali se ne preporučuje
  - Poslovna i kontrolna logika pripada servletima
  - Zadatak JSP stranice je prezentacija korisniku

Forward i include

- Pomoću metoda forward objekta RequestDispatcher:
  - Kontrola se nepovratno predaje novoj stranici
  - Originalna stranica ne može generisati bilo kakav izlaz
- Pomoću metoda include objekta RequestDispatcher:
  - Kontrola se trenutno predaje novoj stranici
  - Originalna stranica može generisati izlaz prije i poslije uključene stranice
  - Originalni servlet ne vidi izlaz uključene stranice
  - Korisno za portale: JSP prikazuje dijelove, ali se dijelovi uređuju različito za različite korisnike

include

```
response.setContentType("text/html");
String firstTable, secondTable, thirdTable;
if (someCondition) {
    first = "/WEB-INF/Sport.jsp";
    second = "/WEB-INF/Stock.jsp";
    third = "/WEB-INF/Weather.jsp";
} else if (...) { ... }
RequestDispatcher dispatcher =
request.getRequestDispatcher("/WEB-INF/Header.jsp");
dispatcher.include(request, response);
dispatcher =
request.getRequestDispatcher(first);
dispatcher.include(request, response);
dispatcher =
request.getRequestDispatcher(second);
dispatcher.include(request, response);
dispatcher =
request.getRequestDispatcher(third);
dispatcher.include(request, response);
dispatcher =
request.getRequestDispatcher("/WEB-INF/Footer.jsp");
dispatcher.include(request, response);
```