ИНТЕРНЕТ ПРОГРАМИРАЊЕ
JAVA SERVER PAGES (MVC) M2
ПРЕДАВАЊА

САДРЖАЈ

MVC – Model view controller

MVC – arhitektura koja predviđa postojanje odvojenih komponenata zaduženih za prezentacionu i poslovnu logiku sistema

MVC podrazumijeva postojanje sljedeća 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

Varijacija MVC arhitekture:

- HMVC (Hierarchical model-view-controller)
- MVA (Model-View-Adapter)
- MVP (Model-View-Presenter arhitektura)
- MVVM (Model-View-ViewModel)

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:

- kontroler komunicira sa modelom, ali ne mora da generiše HTML kod
- JSP ne komunicira sa modelom, jer podatke dobija od Controller-a, tako da JSP služi samo za kreiranje HTML koda

JSP M2

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

dobre strane MVC arhitekture:
- model je moguće prikazati na različite načine
- lako dodavanje novog prikaza podataka
- lakša izmjena interakcije sa korisnikom
- reupotreba koda
- lakše testiranje pojedinačnih dijelova
- prikaz podataka je odvojen od logike obrade podataka
- podjela rada

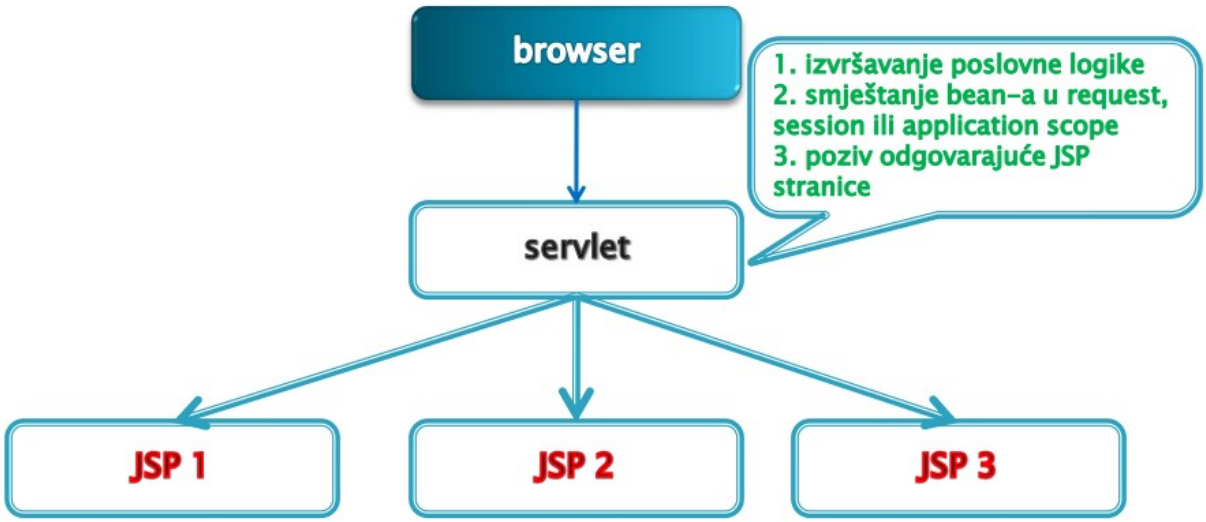
Moguće je koristiti određeni framework – ponekad je upotreba frameworka korisna
- Struts
- JavaServer Faces (JSF)

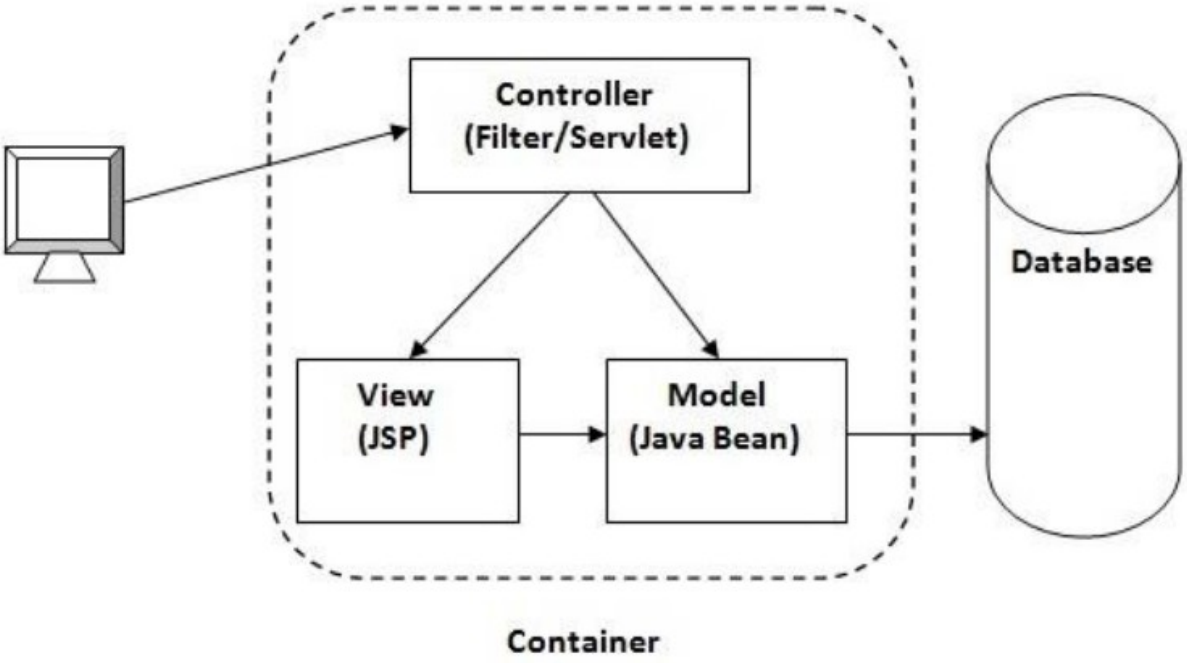
Njihovo korištenje nije obavezno!

MVC je moguće implementirati pomoću RequestDispatcher-a i dobiti sasvim zadovoljavajuće rezultate za jednostavne i srednje kompleksne aplikacije

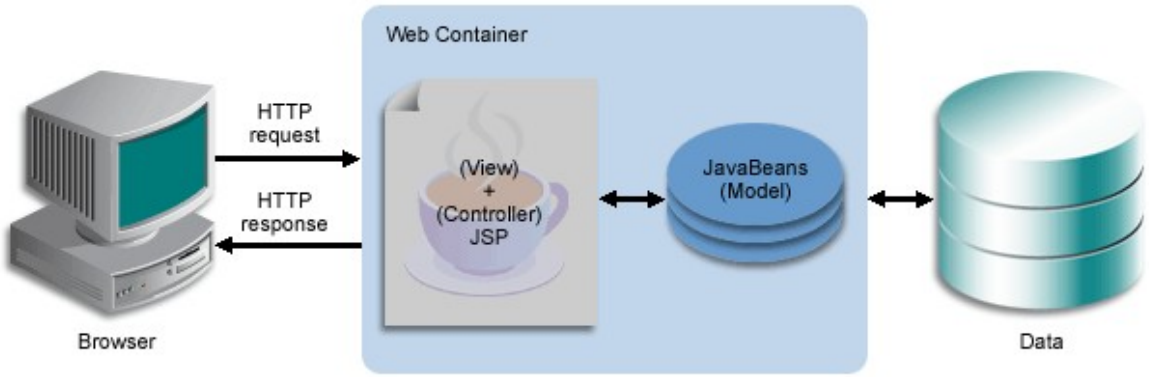
MVC u potpunosti mijenja dizajn sistema

naziva se i Model 2 pristup

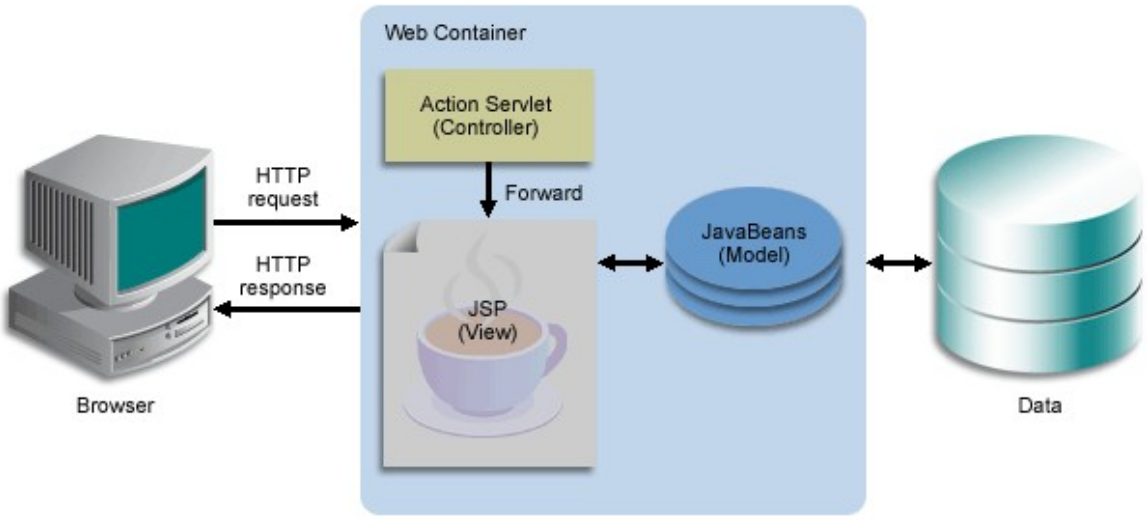

browser
1. izvršavanje poslovne logike
2. smještanje bean-a u request, session ili application scope
3. poziv odgovarajuće JSP stranice
servlet
JSP 1
JSP 2
JSP 3


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

JSP Model1 Architecture



JSP Model 2 Architecture



JSP M2 – КОРАЦИ

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.
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.

```
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);
}
```

USE BEAN – MVC MODEL

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

REQUEST BASED DATA SWAP

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

SESSION-BASED DATA DISTRIBUTION

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}
```

SEND REDIRECT

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 stranicu bez pristupanja servletu, pa JSP podaci mogu da budu nedostupni
- U okviru JSP stranice trebalo bi detektovati ovu situaciju

APLLICATION-BASED DATA SHARE

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

- zahtjev se prenosi na druge resurse na istom serveru radi daljeg procesiranja
- web čitač nije uključen u ovaj proces
- u web čitaču se ne vidi adresa resursa koji generiše odgovor, već adresa resursa na koji je zahtjev upućen
- ako je potrebno koristiti iste podatke u drugom resursu, može se upotrebiti request.setAttribute ()
- brži od sendRedirect, jer nema dodatnog zahtjeva

sendRedirect

- web čitač šalje novi zahtjev na osnovu prethodnog odgovora servera
- u web čitaču se vidi adresa resursa koji generiše odgovor, jer je web čitač uputio novi zahtjev
- ako je potrebno koristiti iste podatke u drugom resursu, može se upotrebiti sesija ili proslijediti podatke u zahtjevu (dio URL-a)
- sporiji od forward, jer postoji dodatni zahtjev

```
<% 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

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 (uključenoj 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

```
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);
```