ⰂⰏⰑⰉⰓⰅⰐⰉ ⰓⰀⰆⰄⰐⰀⰓⰄⰏⰑⰉ

ⰒⰑⰌ, ⰆⰐⰄⰏ, ⰆⰐⰀⰑⰀⰞⰅⰀⰐⰄ

ⰒⰀⰓⰉⰂ ⰏⰀⰃⰀⰀⰓ

Otvoreno računarstvo

- Tehnologije za izradu aplikacija Weba (nastavak)
  - Aplikacije iz svijeta Jave EE
    - Model-View-Controller (MVC)
    - JavaServer Pages (JSP)
    - JavaBeans

Mario Žagar

Model-View-Controller (MVC)

Uzorci dizajna (*design patterns*)

- Uzorci dizajna (*design patterns*) pripadaju u skupinu arhitekturalnih uzoraka
  - Arhitekturalni uzorci su programski predlošci – koncepti koji predočavaju dobro-poznata rješenja arhitekturalnih problema u programskom inženjerstvu
  - Opisuju elemente i njihove relacije te ograničenja korištenja
- Uzorak dizajna nije arhitektura već koncept
  - Primjeri najbolje prakse kako treba graditi arhitekturu
- Isti uzorak dizajna se može implementirati u nizu raznih arhitektura koje ispoljavaju iste karakteristike
  - Definirani kao “striktno opisani i široko dostupni”

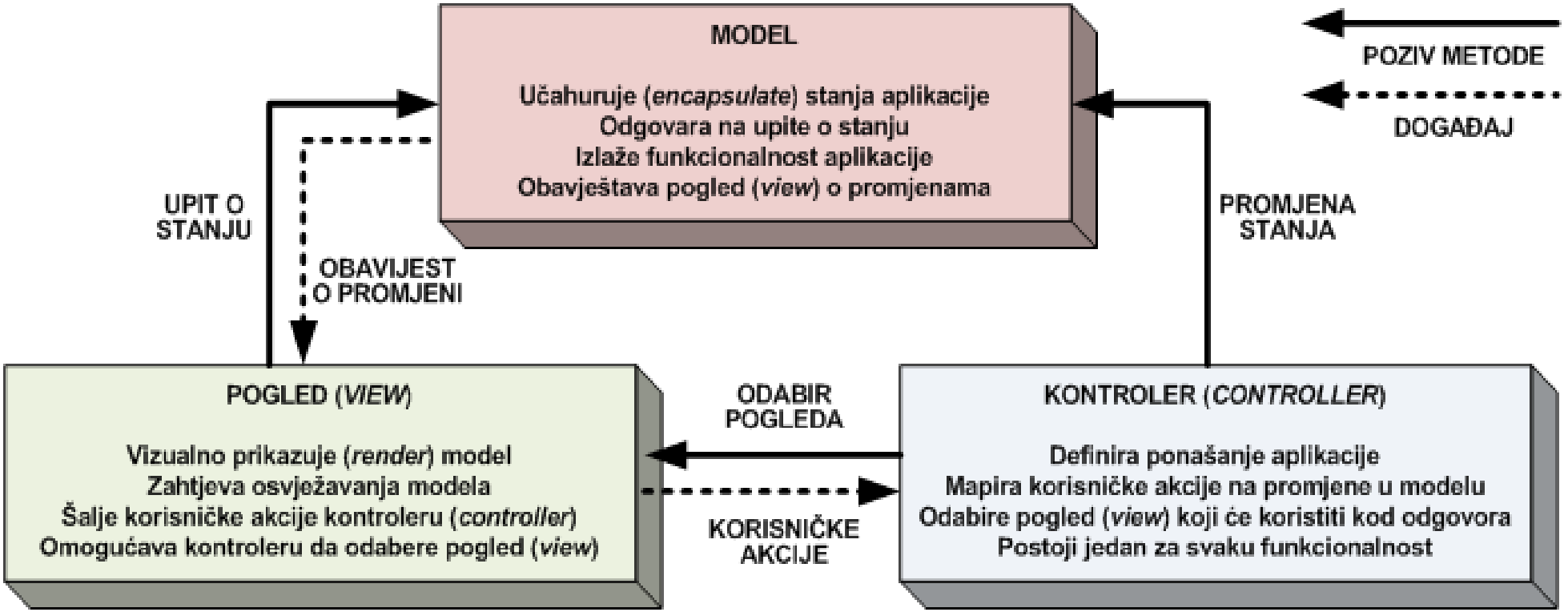
Model-View-Controller uzorak dizajna

- **Izolacija poslovne logike od korisničkog sučelja**
  - Razdvajanje korisničke interakcija, prezentacije podataka, poslovne logike i pristupa podacima
- **Smanjenje međuzavisnosti slojeva** aplikacije
  - Jednostavnije promjene vizualnog dizajna aplikacije, ali i poslovnih pravila, bez utjecaja promjene jednog na doradu drugog
- Povećanje mogućnosti **ponovnog korištenja** dijelova aplikacije
- Jednostavnije dorade novih funkcionalnosti
- Smanjenje ponavljanja dorada prilikom promjena
  - Nema *copy-paste*, jer je kôd na jednom mjestu

MVC uzorak dizajna

- MVC dijelovi:
  - **Model** – predstavlja poslovnu logiku i podatke
  - **Pogled** (View) – predstavlja korisničko sučelje
    - Može postojati više korisničkih sučelja
  - **Kontroler** (Controller) – komunikacija modela i pogleda
    - Povezuje korisničko sučelje i poslovnu logiku
- MVC – konkretne koristi:
  - Ponovno korištenje kôda – poslovna logika uvijek na jednom mjestu
  - Kraće vrijeme programiranja
  - Paralelni razvoj (timski rad)
  - Moguće velike promjene jednog sloja (bilo kojeg) bez utjecaja na druge

MVC – prikaz

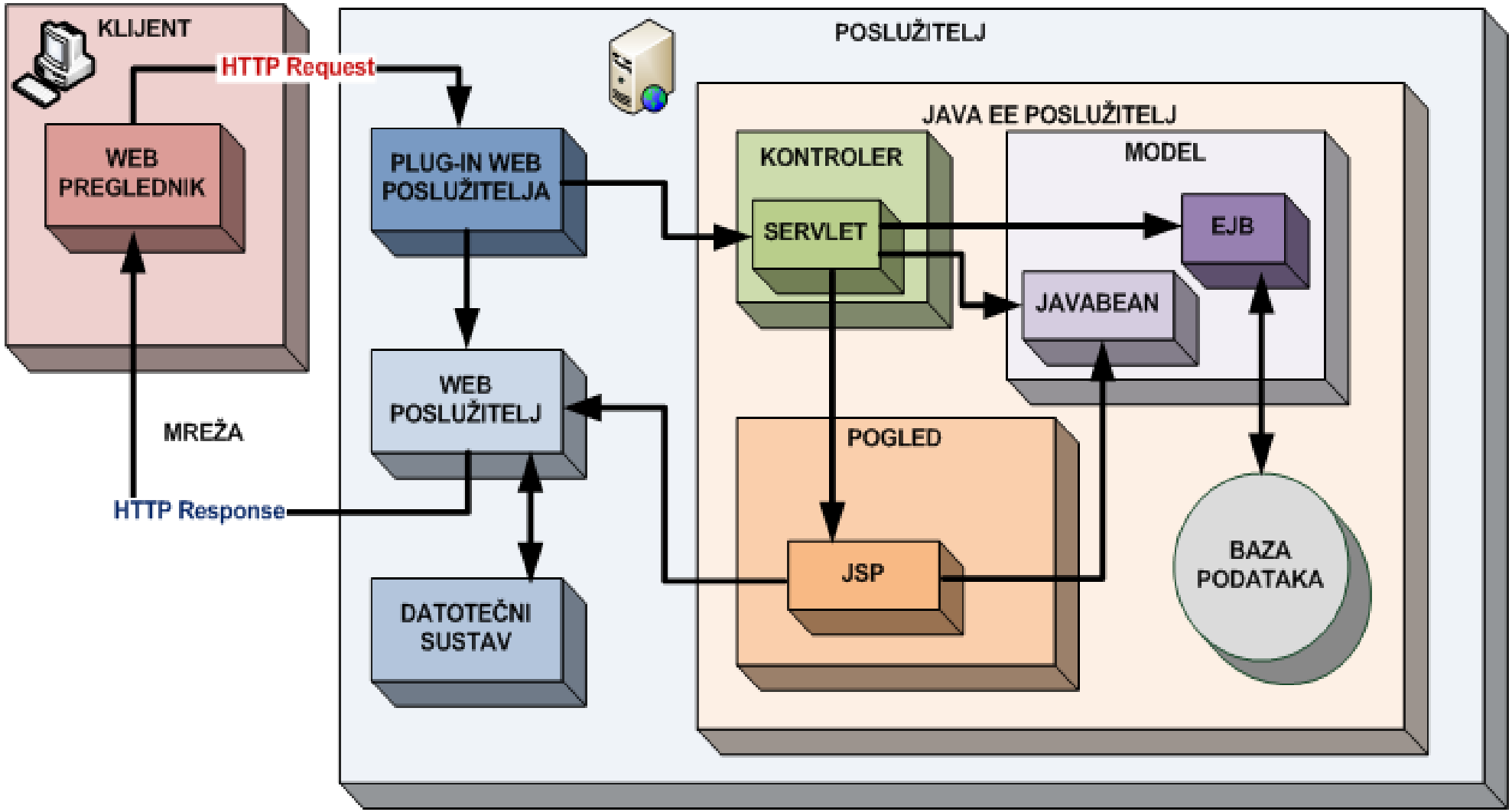


- Više na: http://java.sun.com/blueprints/patterns/MVC-detailed.html

MVC – proširenje prikaza

- Uz 3 osnovna MVC sloja, dodan četvrti sloj
  - **sloj postojanosti** (*persistence*)
  - ostvaruje se najčešće bazom podataka
- U Java EE arhitekturi
  - Pogled i kontroler: **JavaServer Pages** (**JSP**) i **Servlet**
  - Model: **JavaBean** i **Enterprise JavaBean** (**EJB**)
  - Postojanost: pohrana podataka (relacijska baza podataka i sl.)

MVC – primjena na Java EE



Sadržaj dinamičke Web stranice

- Web stranica (HTML kôd) se generira iz:
  - Statičkog, ne-personaliziranog sadržaja – uvijek isti
  - Dinamičkog – ovisnog o trenutnom stanju aplikacije
- **Osnovni izgled stranice** - najčešće potpuno **statičan**:
  - Razmještaj (*layout*)
  - Dizajn – CSS
  - Stalni grafički elementi – slike i sl.
- **Sadržaj stranice** - **dinamičan**:
  - Tekst, podaci
  - Moguće korištenje XML + XSL

Java Server Pages (JSP)

JavaServer Pages (JSP) - ideja

- **Miješanje statičkog** HTML kôda s **dinamički generiranim** HTML kôdom **pomoću oznaka**
  - Statički kôd je čisti HTML ili XML
  - Dinamički kôd se generira skriptnim jezikom
- Dozvoljava uporabu skriptnog jezika na poslužiteljskoj strani (*server-side scripting*)
- JSP datoteka (.jsp) posljedično sadrži:
  - HTML ili XML oznake
  - JSP sintaksu
- JSP specifikacija
  - Prvo izdanje 1999.g.
  - Aktualna inačica JavaServer Pages 2.1

JSP – sintaksa

- Sadrži:
  - **Direktive**
    - naredbe JSP stroju ili prevodiocu
  - **Skripta**
    - **Deklaracije** – dodatne metode i varijable kreirane u JSP Servletu
    - **Skriptleti** – ugrađeni Java kôd
    - **Izrazi** – Java kôd koji rezultira znakovnim nizom (String)
      - Alternativa: **JSP Expression Languages** (JSP EL)
        - Danas: **unified expression language** (unified EL)
  - **Akcije**
    - Standardne: korištenje beanova, tok procesa
    - Dodatne korisničke akcije

Razlike JSP – Servlet

- JSP se koriste za:
  - prikaz sadržaja generiranog Web aplikacijom
  - određivanje vrste procesiranja
  - validaciju ulaznih podataka
  - rad s poslovnim objektima za pristup podacima i izvođenje procesa
  - kontrolu toka Web aplikacije
- JSP i Servleti sadržavaju niz istih značajki
  - JSP se zapravo (neprimjetno) prevode u Servlete
- JSP ograničenje – samo za tekstualne podatke
  - Servleti se i dalje koriste kao sučelje prema aplikacijama, appletima, kao i za binarne podatke
    - Primjer: proizvesti i isporučiti PDF datoteku može samo Servlet

JSP – prednosti

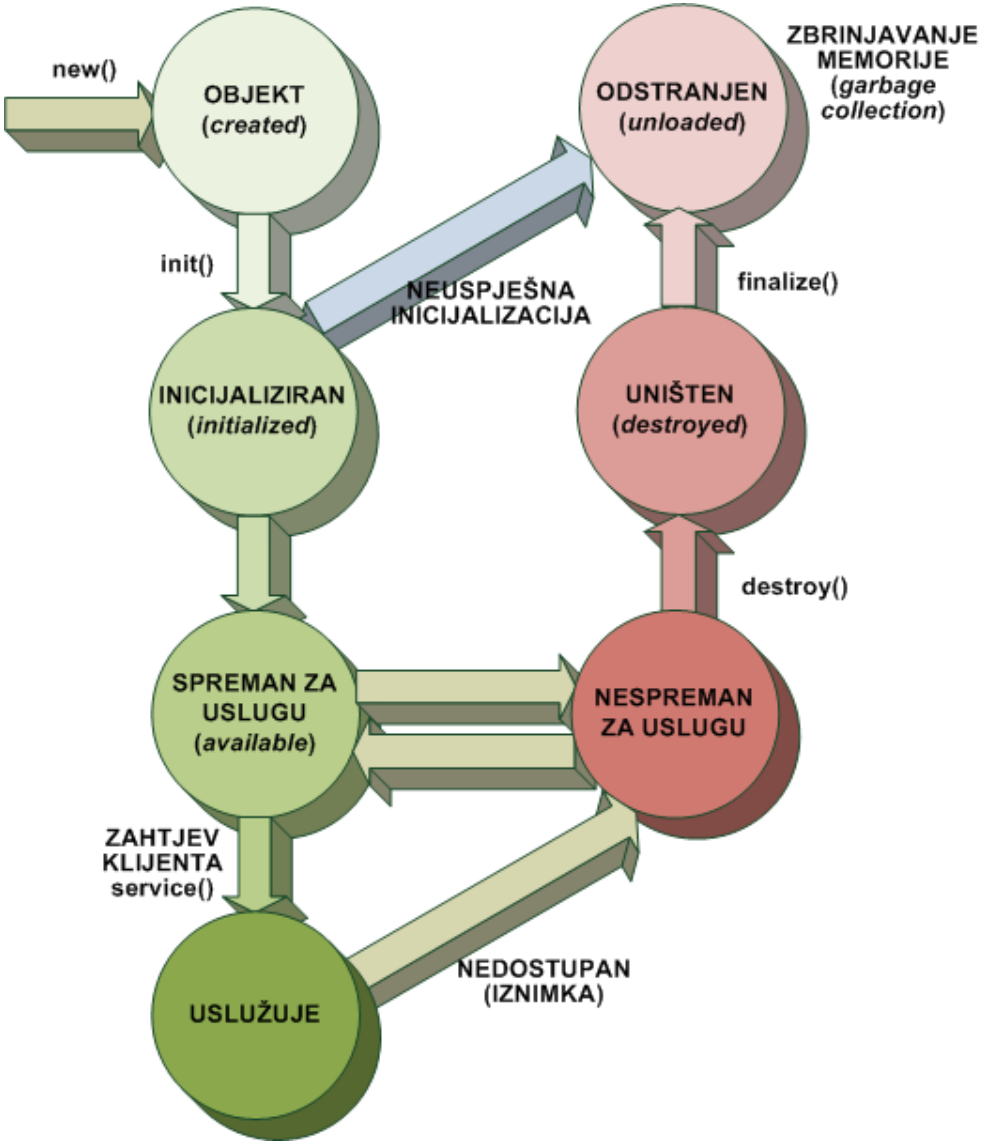
- Odvajanje statičkog i dinamičkog sadržaja
  - Logika za generiranje dinamičkog sadržaja odvaja se u vanjske JavaBeans komponente
  - Promjena prezentacijskog predloška JSP stranice uzrokuje automatsko prevođenje i ponovno učitavanje stranice
- Jednostavno prenošenje između raznih platformi
- Java EE standardiziranost
  - Preporuka za prezentaciju dinamičkih podataka izgrađena na (poznatoj) Servlet tehnologiji
  - Ponovno korištenje komponenata i biblioteka oznaka
- Podrška vrhunskim dizajnerskim/programerskim alatima

JSP – izvršni model

- JSP stranice se izvršavaju u Web sadržniku
  - Sadržnik preusmjera zahtjeve klijenta prema odgovarajućoj JSP stranici i vraća odgovora klijentu
- JSP stranice se prevode u Servlete i izvršavaju
  - Proces zvan PageCompilation
  - **JSP izvorna stranica** – dizajnerski model pisan oznakama
  - Prevedeni **JSP Servlet** (Servlet nastao iz JSP) – izvršni model
- Koraci izvođenja:
  - Parsiranje JSP izvorne stranice
  - Generiranje izvršnog kôda JSP Servleta
  - Prevođenje JSP Servleta, učitavanje i izvršavanje

JSP – životni ciklus

- JSP se prevodi u Servlet
- JSP Servlet ima isti životni ciklus kao i Servlet
- Pri prvom zahtjevu za stranicom se kreira novi JSP Servlet
- Pri svakom sljedećem zahtjevu se koristi postojeća instanca JSP Servleta
  - Osim ako se JSP izmijenio



JSP Servlet – primjer

- **JSP izvorna stranica**

```
<%@ page import="hr.fer.or" %>


<html><head>
    <%! int temp = 1;%>
    <% int temp2 = 2; %>
  </head>
  <body>
    <h1><%= toString( "tekst" + 1 ) %></h1>
  </body>
</head>
```

- **JSP Servlet**

```
package jsp_servlet;
 import ...
 import hr.fer.or;
 class _myservlet implements javax.servlet.Servlet,
      javax.servlet.jsp.HttpJspPage {
  int temp = 1;
  ...
   public void _jspService( javax.servlet.http.HttpServletRequest
     request, javax.servlet.http.HttpServletResponse response )
     throws javax.servlet.ServletException, java.io.IOException {
    ...
   javax.servlet.jsp.JspWriter out = pageContext.getOut();
   try {
     out.print( "<html><head>\r\n" );
     ...
     int temp2 = 2;
     ...
     out.print( "</head>\r\n" );
     out.print( "<body>\r\n" );
     out.print( "<h1>" );
     out.print( toString( "tekst" + 1 ) );
     out.print( "</h1>\r\n" );
     ...
```

JSP – vidljivost atributa

- Pristup atributima – 4 razine vidljivosti:
  - **Stranica** (*page*) – samo trenutna JSP stranica
    - Gubi se kod povratka
    - Gubi se prilikom prosljeđivanja na drugu stranicu
    - Koriste se set/getAttribute metode
  - **Poziv** (*request*) – trenutni HttpServletRequest objekt
    - Ostaje vidljiv kod prosljeđivanja ili uvlačenja stranice
    - Gubi se kod povratnog odgovora
  - **Sjednica** (*session*) – trenutni HttpSession objekt
    - Vidljivost za vrijeme trajanja sjednice
  - **Aplikacija** (*application*) – trenutni ServletContext objekt
    - Vidljivost kroz cijelu aplikaciju

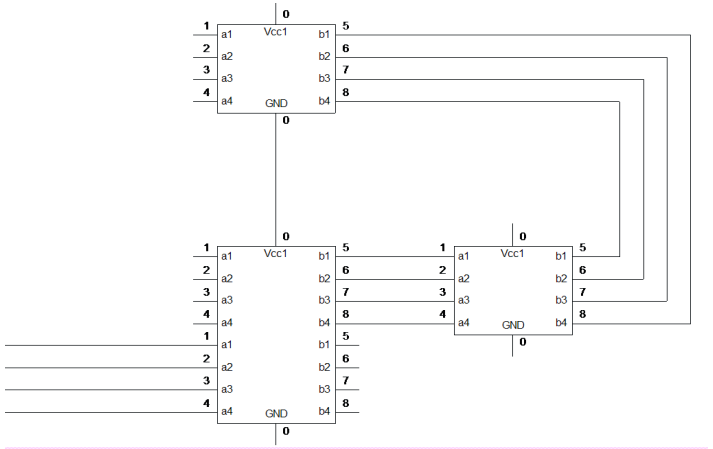
JSP – dobre strane i mogući problemi

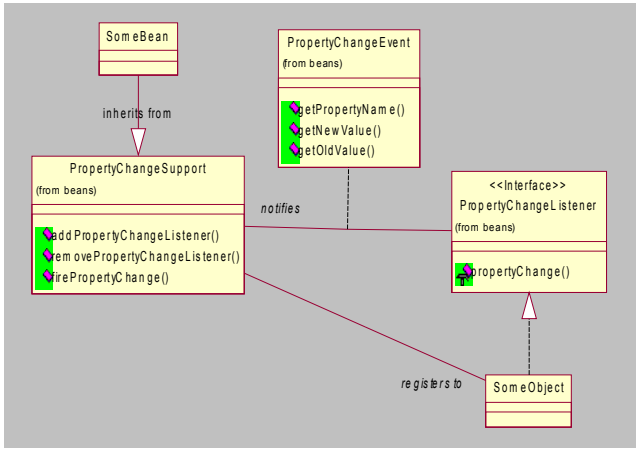
- Dobre strane:
  - Uporaba dizajnerskih vizualnih alata za dizajn stranice
  - Uporaba programerskih alata za programsku logiku
  - Izrada sadržaja u skladu s MVC uzorkom dizajna
  - Nema miješanja Java kôda i HTML kôda
- Mogući problemi:
  - N(en)amjerno stavljanje programske logike u JSP stranicu
  - Moguće dodati mnogo Java kôda u JSP stranicu koji služi za promjenu izgleda stranice (straničenje, filtari, sort)
    - JSP stranice više ne služe samo za prezentaciju (Pogled)
- Rješenje:
  - Korištenje drugih tehnologija i korisničkih oznaka
    - npr. JavaBeans i Custom Tags

JavaBeans

Komponentna arhitektura

- Definicija:
  - *A Java Bean is a reusable software component that can be visually manipulated in builder tools. To understand the precise meaning of this definition of a Bean, clarification is required for the following terms:*
    - *Software component*
    - *Builder Tool*
    - *Visual manipulation*

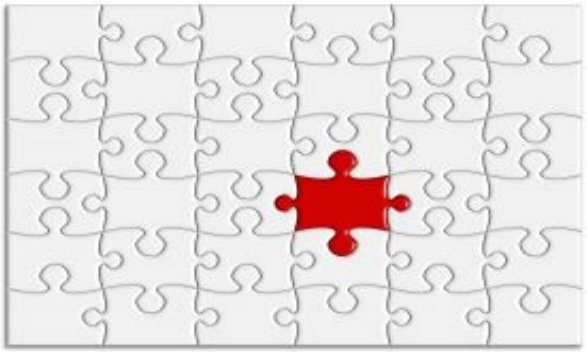




Komponentna arhitektura

- **Komponenta:**
  - **Samostalni** (*self-contained*), **ponovno iskoristiv** (*reusable*) dio programske podrške koja se može u i koristiti u složenim modulima
  - **Pakirani programski objekt** s **normiranim** (poznatim) **sučeljem**

- **Komponentna arhitektura:**
  - Skup specifikacija o detaljima **suradnje** komponenata
    - Međusobne i s drugim programima



JavaBeans

- **JavaBeans** označava:
  - Specifikaciju i implementaciju programskog komponentnog modela
  - Naziv tehnologije
- **JavaBean (ili bean)**
  - Referenca na objekt izgrađen po JavaBeans specifikaciji
  - Napisani (opet) u Javi – platformska neovisnost
- JavaBeans arhitektura
  - Temeljena na komponentnom modelu
  - Omogućava programerima izradnju komponenata
- JavaBeanovi
  - Korištenje u aplikacijama, Servletima, appletima...

JavaBean – koncepti

- Alati za izgradnju otkrivaju **značajke** *Beana*:
  - **Svojstva, metode, događaje**
  - Metoda **introspekcije**
    - Istraživanje pridržavanjem određenih pravila (**uzorci dizajna**)
    - Eksplicitnim uvidom u značajke preko informacija u BeanInfo
- **Svojstva** su karakteristike događaja i ponašanja
  - Mogu se mijenjati i **dorađivati** u trenutku dizajna:
    - Pomoću uređivača svojstava (*properties editor*)
    - Korištenjem dorađivača (*bean customizer*)
- *Beanovi* međusobno komuniciraju **događajima**
  - Osluškivač (*listener*) Bean i izvorišni (*source*) *Beana*
    - Alati istražuju Bean i uočavaju koje događaje Bean može poslati i primiti
- **Postojanost** omogućava pohranu **stanja** *Beana*
  - Pohranjuje se i restaurira se – **serijalizacija** (*Serialization*)
- **Metode** *Beanova* - standardne javne Java metode

JavaBean – karakteristike

- *JavaBean* je (samo) instanca (objekt) Java razreda
- Sadrži skup javnih svojstava:
  - **Svojstva** (atribute, varijable), **metode**, **događaje**
- Tipovi:
  - **Vizualni:**
    - **Grafički** (GUI) – npr. graf, animacija, ...
  - **Nevizualni**:
    - **Poslovna logika** – npr. sadržaj košarice u Internet trgovini
    - **Programski kod** – npr. zbroj prometa, provjera pravopisa
    - **Podaci** (baza) – npr. dohvat ranga na OR

- *JavaBeans API*
  - API opisuje komponentni model za programski jezik
    - Specifikacija – *JavaBeans* komponentnu arhitekturu

JavaBean – značajke i API

- Značajke *Beanova*:
  - Dinamički, mogu se mijenjati i prilagođavati
    - Vizualni alati (npr. NetBeans)mogu mijenjati *Bean*
    - Na Properties karticama se podešavaju značajke *Beana*
  - Niz promjenjivih svojstava
    - Vizualna manipulacija (*Drag'n'Drop*), promjena izgleda
    - Promjena izgleda i ponašanja
    - Definiranje interakcije s drugim komponentama
  - **Sintaksa i konvencije imenovanja:**
    - **Konstruktor bez argumenata**
    - **Svojstva** (varijable) dostupne **samo** putem pristupnih metoda
      - Korištenje **getter i setter metoda**
    - Implementira sučelje ***Serializable***

JavaBean – tipovi značajki

- Tipovi značajki:
  - **Jednostavne** – jedna jednostavna značajka
  - **Logičke** – istinosna vrijednost (istina/laž)
    - Metode:
      - public boolean **is**Naziv()
      - public void **set**Naziv(**boolean** vrijednost)
      - **Ne sadrže get**Naziv metodu
  - **Indeksirane** – nizovi vrijednosti
    - Korisne kad treba pohraniti niz značajki istog tipa
    - Kao nizovi u Javi, cjelobrojni indeksi
    - Metode:
      - public String **get**Nazivi(int indeks)
      - public void **set**Nazivi(int indeks, String naziv)
      - public String**[] get**Nazivi()
      - public void **set**Nazivi(String**[]** nazivi)

JavaBean i MVC

- MVC:
  - **Model** – jedan, poslovna logika, rješava problem
    - **JavaBean**
  - **Pogled** – višestruk, "pogled" na model, prezentacija, interakcija s korisnikom
    - Web stranica – **Servlet**, **JSP**
    - Klijent – **GUI JavaBean**
  - **Kontroler** – jedan ili više, obrađuje zahtjeve i parametre
- Ideja: enkapsulacija dinamičkog sadržaja u JavaBean
- JavaBeanovi se održavaju neovisno od stranice
- JSP stranice pristupaju *JavaBeanu* putem <jsp:useBean> oznake

JavaBean – primjer – student

```
public class Student implements java.io.Serializable {
   private String firstName;
   private String lastName;
   private boolean active;

   public Student() {
   }

   public String getFirstName() {
      return this.firstName;
   }
   public void setFirstName(String name) {
      this.firstName = name;
   }

   public String getLastName() {
      return this.lastName;
   }
   public void setLastName(String name) {
      this.lastName = name;
   }
   public boolean isActive() {
      return this.active;
   }
   public void setActive(boolean active) {
      this.active = active;
   }
}

public class TestBean {
   public static void main(String[] args) {

      Student student = new Student();
      student.setFirstName("Ivo");
      student.setLastName("Ivić");
      student.setActive(true);

      System.out.print(student.getFirstName() + " je ");
      System.out.println(student.isActive() ? "neaktivan"
      : "aktivan");
   }
}
```

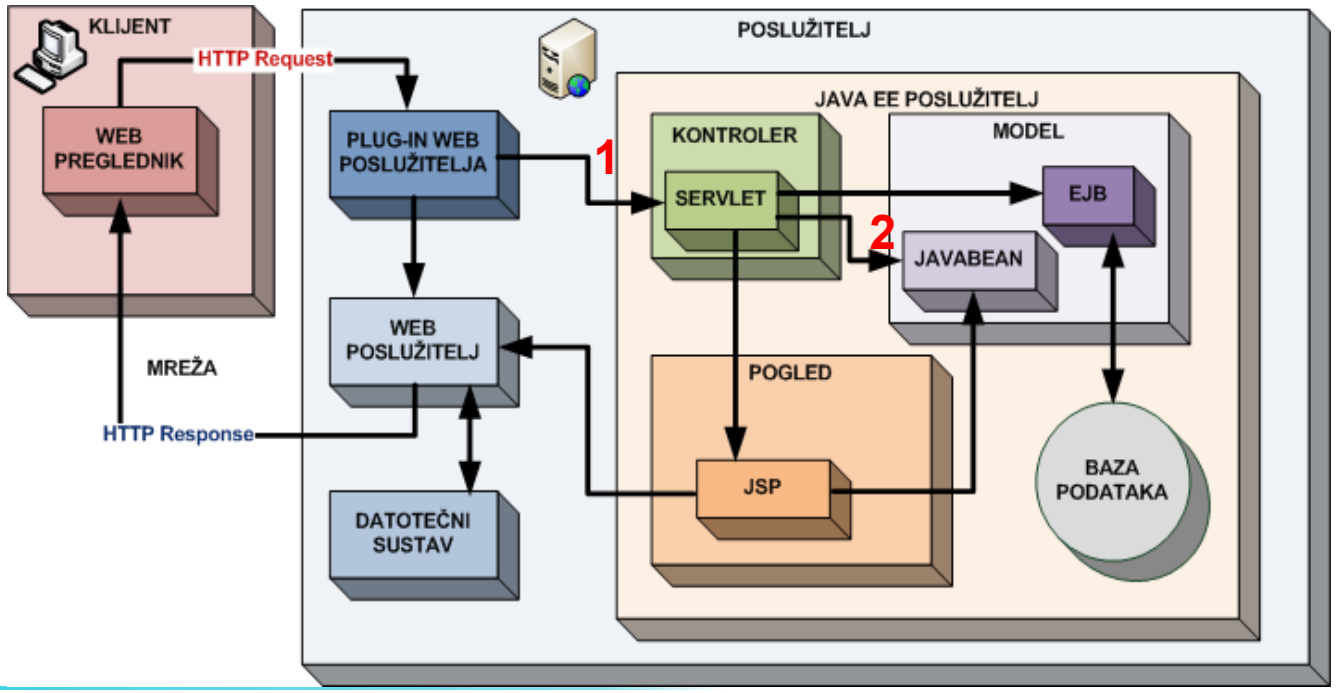
JavaBean - koristi

- Zašto koristiti *JavaBeanove*?
  - Programiranje korištenjem vizualnih programskih alata
  - Pouzdan model učahurivanja
  - Standardiziran u Java EE Connector Architecture (JCA)

Napomena: Ne miješati *JavaBean* s *Enterprise JavaBeans* (EJB)!

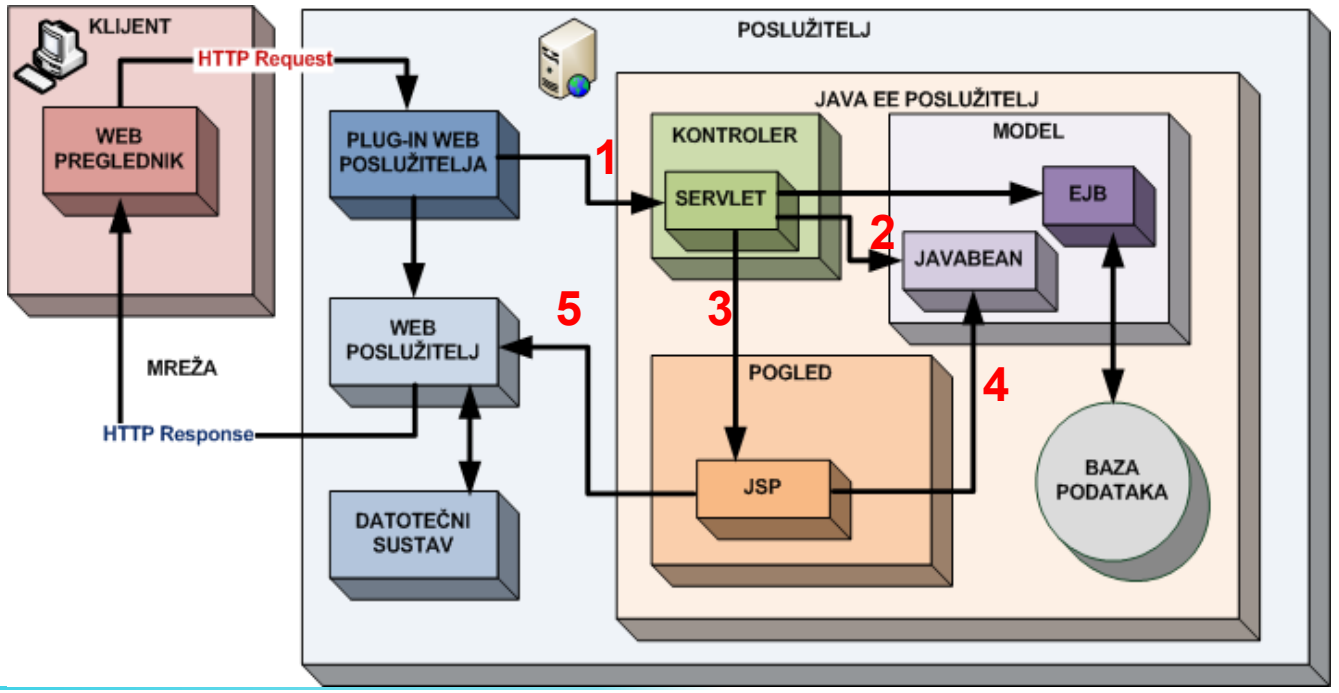
Servlet-JSP-*JavaBean* cjelina

1) Klijent šalje HTTP Request Web poslužitelju koji ga prosljeđuje aplikacijskom poslužitelju
   - Poziva se **service**() metoda odgovarajućeg Servleta
2) Servlet stvara / pokreće odgovarajuće JavaBeans komponente i prosljeđuje poziv odgovarajućoj JSP stranici.
   - JavaBeans komponente mogu pristupati drugim sustavima ili bazi podataka i posčkedično sadrže podatke za izradu dinamičkog sadržaja u JSP stranici



Servlet-JSP-*JavaBean* cjelina

3) Servlet prosljeđuje poziv JSP stranici.
   Pri prvom pozivu JSP stranica se prevodi, inače ne
   Poziva se **service**() metoda prevedenog JSP Servleta
4) JSP Servlet čita sadržaj *JavaBean* komponente
5) JSP Servlet izrađuje sadržaj Web stranice te ga prosljeđuje Web poslužitelju koji ga prosljeđuje Web pregledniku (klijentu)



Pitanja ?