Spring Framework

Univerzitet u Beogradu
Fakultet organizacionih nauka
Miloš Milić

Sadržaj

- Uvod u Java EE
- Spring Framework
- Zaključak
- Literatura

Uvod u Java EE

- Java – softverska tehnologija
- Platforma i OO programski jezik
- Java tehnologije su definisane specifikacijama
- Java Community Proces JCP – http://jcp.org
- Java platforma uključuje:
  1. Java Platform, Standard Edition (Java SE)
  2. Java Platform, Enterprise Edition (Java EE)
  3. Java Platform, Micro Edition (Java ME)

Uvod u Java EE

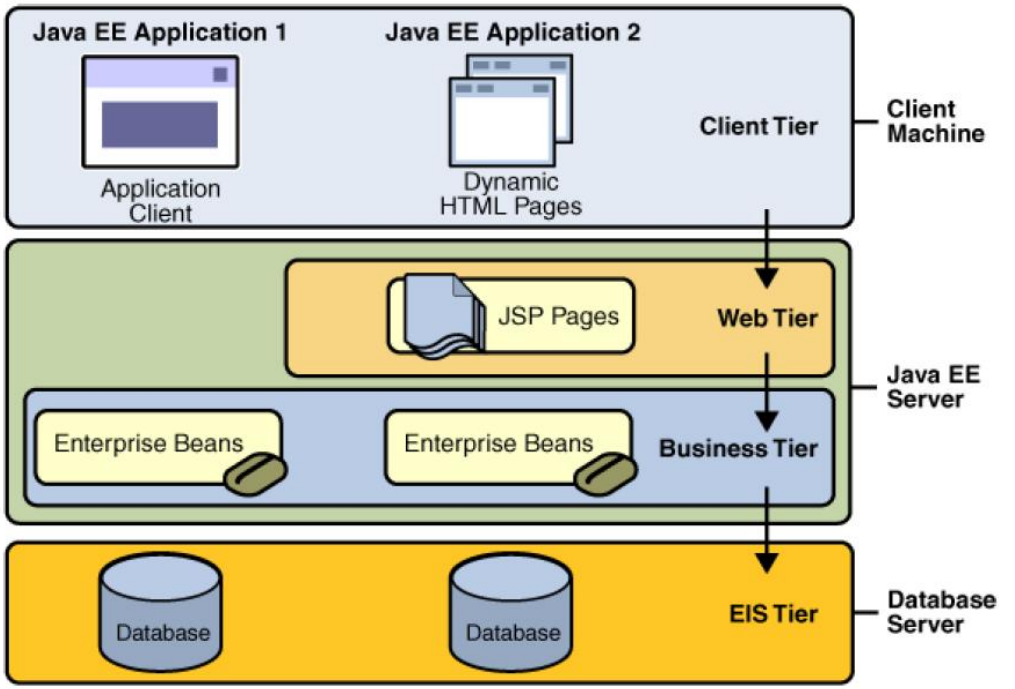
- JEE – obezbeđuje Java EE API i alate za građenje, testiranje i izvršavanje JEE aplikacija
- Java EE je zasnovana na Java SE
- Sastoji se od velikog broja tehnologija:
  a) Java Web tehnologija
  b) Enterprise JavaBeans (EJB) tehnologija
  c) Java XML tehnologija

Uvod u Java EE

- Karakteristike:
  - Višenivojski distribuirani aplikacioni model
  - Komponente koje se mogu ponovo koristiti
  - Jedinstveni model zaštite
  - Fleksibilnu kontrolu transakcija
  - Web servisi zasnovani na XML standardima i protokolima
- Verzije: J2EE 1.3, J2EE 1.4, Java EE 5.0, Java EE 6.0...
- Svaka verzija podržana je odgovarajućim tehnologijama

Uvod u Java EE

- Višenivojske aplikacije
- Klijent, Java EE server, Database server

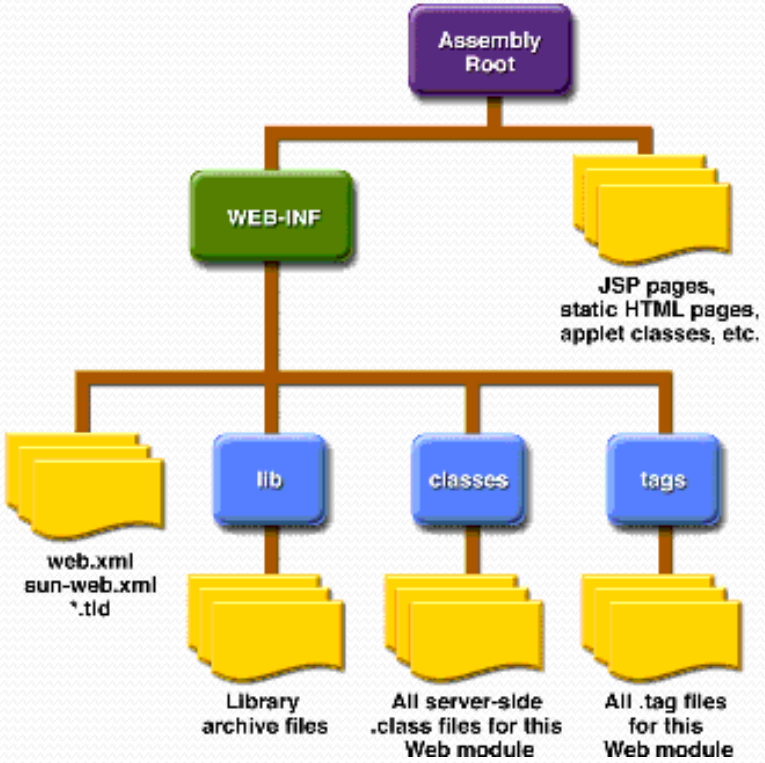


Uvod u Java EE

- Java EE komponente:
  - aplikacioni klijenti i apleti (klijent)
  - Java Servlet, JavaServer Pages (server)
  - Enterprise JavaBeans (server)
- Ugrađuju se u Java EE aplikaciju ukoliko zadovoljavaju Java EE specifikaciju
- Njihovim izvršenjem upravlja Java EE server

Uvod u Java EE

- Struktura web modula



Uvod u Java EE

- WEB-INF direktorijum je privatni direktorijum aplikacije
- Resursi dostupni samo servletima i Java klasama koje sačinjavaju web aplikaciju
- Organizacioni okvir za web aplikacije
- WAR (WEB archive)
- Opisivač rasporeda u vreme izvršenja (runtime deployment descriptor) - informacije o kontekstu Web aplikacije (gde je početak – koren Web aplikacije preko koga se pristupa Web komponentama)

Uvod u Java EE

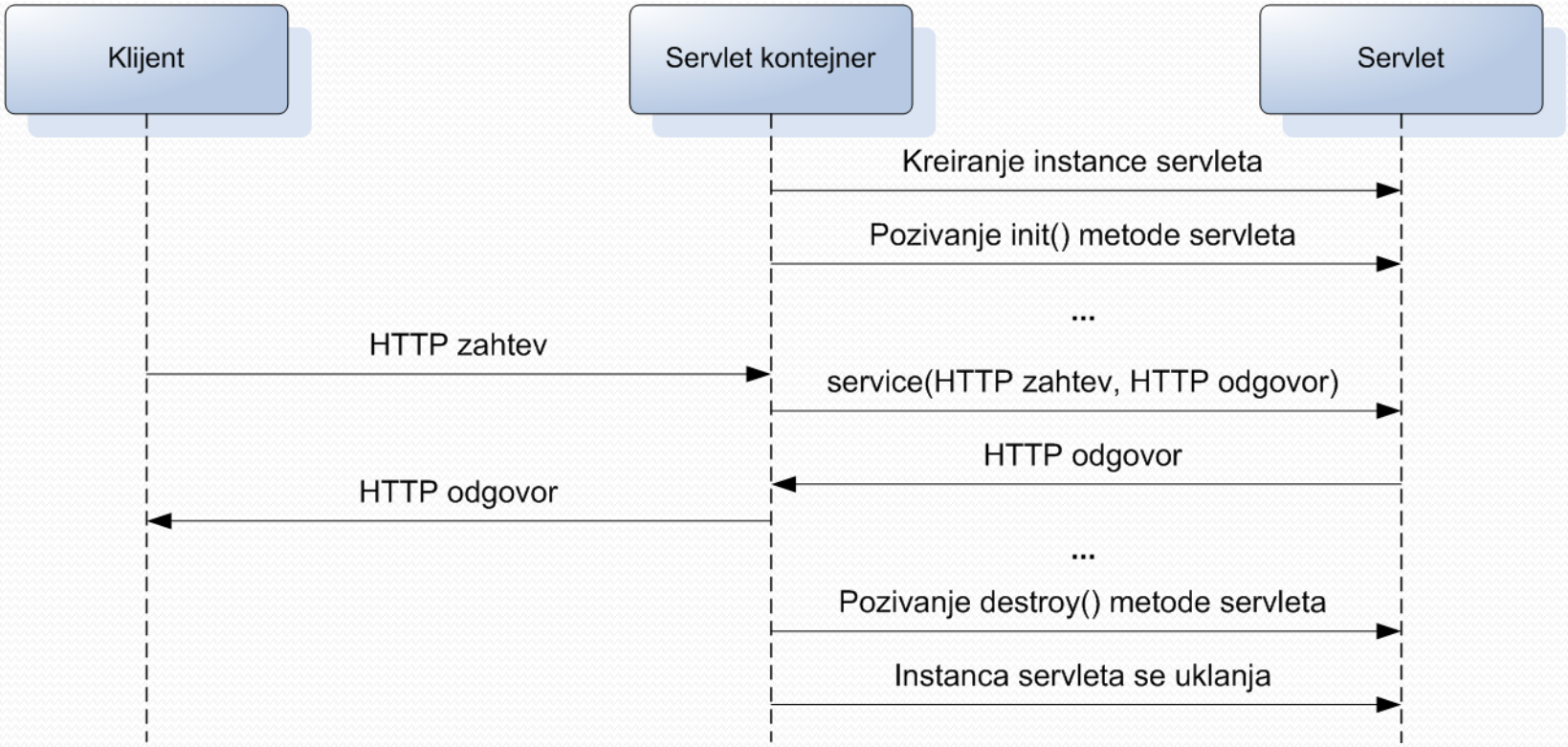
- Java servleti – dinamički procesiraju zahteve i kreiraju odgovarajući odgovor
- Napisani su u Javi
- Komponentno zasnovan i platformski nezavistan mehanizam
- Zahtevaju postojanje servlet kontejnera
- Trenutno je aktuelna Servlet 3.1 specifikacija
- Ograničenje servleta: korišćenje HTML-a, internacionalizacija, raspodela odgovornosti...

Uvod u Java EE

- Mapiran pomoću jedne ili više URL adresa
- Kada server primi korisnički zahtev (request) – pobuđuje se metoda service() i procesira zahtev generišući određeni odgovor (response)
- Za svaki korisnički zahtev kreira se odvojena nit (thread) koja je povezana sa tim zahtevom
- Više niti ili korisnika može da aktivira service() metodu datog servleta istovremeno

Uvod u Java EE

- Životni ciklus servleta



Uvod u Java EE

- Definisanje u web.xml*
- Ograničenja servleta: korišćenje HTML-a, internacionalizacija, raspodela odgovornosti...

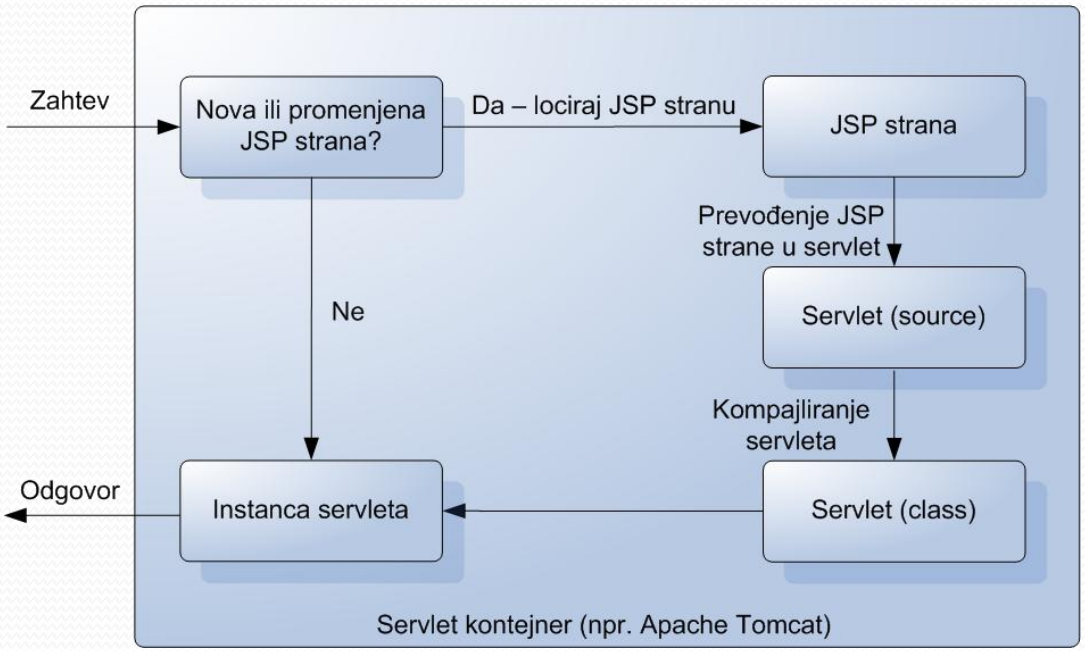
- *Odnosi se na servlet 2.5 specifijaciju. Od Servlet 3.0 specifikacije ne zahteva se korišćenje web.xml deskriptora (koriste se anotacije)

Uvod u Java EE

- JavaServer Pages - proširenje Servlet tehnologije
- Tekstualni dokumenti sa jsp ekstenzijom
- Sadrži kombinaciju HTML i XML tagova i skriptleta
- Skriptleti, direktive, deklaracije, predefinisane varijable, biblioteke tagova, EL (kreiranje izraza, provera uslova, iteracija nad elementima liste...)
- Nakon predprocesiranja iz JSP strana se generišu servleti
- Trenutno je aktuelna JSP 2.3 specifikacija

Uvod u Java EE

- Posle određenog predprocesiranja koje obavlja servlet kontejner iz JSP strana se generišu njima odgovarajući servleti
- Životni ciklus JSP strane



Spring Framework

- Tehnologija za razvoj složenih aplikacija
- Interface21/SpringSource /VMWare/Pivotal: http://spring.io
- Open source projekat
- Rod Johnson
  - Expert One-On-One J2EE Design and Development, Wrox, 2002.
  - Expert One-On-One J2EE Development without EJB, Wrox, 2004.

Spring Framework

- Spring 1.0 u proleće 2003. godine
- Mnogi nisu shvatali da prilikom razvoja aplikacija ima mnogo problema
- Drugačiji i inovativni pristup razvoju aplikacija
- Uticao na razmišljanja mnogih
- Dobre ideje su ugrađene u mnoge tehnologije
- Nije rešenje za sve probleme ali olakšava razvoj aplikacija

Spring Framework

- "The EJB architecture will make it easy to write applications: Application developers will not have to understand low-level transaction and state management details, multi-threading, connection pooling, and other complex low-level APIs.“ [EJB2]

Spring Framework

Problemi u procesu razvoja Java EE aplikacija [Johnson03]:

- Testiranje aplikacija
- Narušavanje principa objektno-orijentisanog razvoja softvera
- Složenost
- Deploy
- Aplikacioni serveri

Spring Framework

Ciljevi Spring framework-a [Johnson]:

- Jednostavnija izrada Java EE aplikacija
- Rešavanje kompletnih zahteva (eng. end-to-end requirements) a ne samo zahteva u jednom sloju
- Najbolje rešenje za inverziju kontrole (IoC)
- AOP implementacija sa fokusom na rešavanje najčešćih problema u razvoju Java EE aplikacija
- Portabilnost u odnosu na aplikacione servere

Spring Framework

Ciljevi Spring framework-a (nastavak):

- “Nenametljiv” (eng. non-invasive) framework
- Primena unit testiranja
- Primena najboljih praksi u procesu objektno-orijentisanog razvoja softvera
- Dobra alternativa EJB
- Povećanje produktivnosti u odnosu na “tradicionalne” Java EE aplikacije

Spring Framework

Stalno unapređenje Spring tehnologije

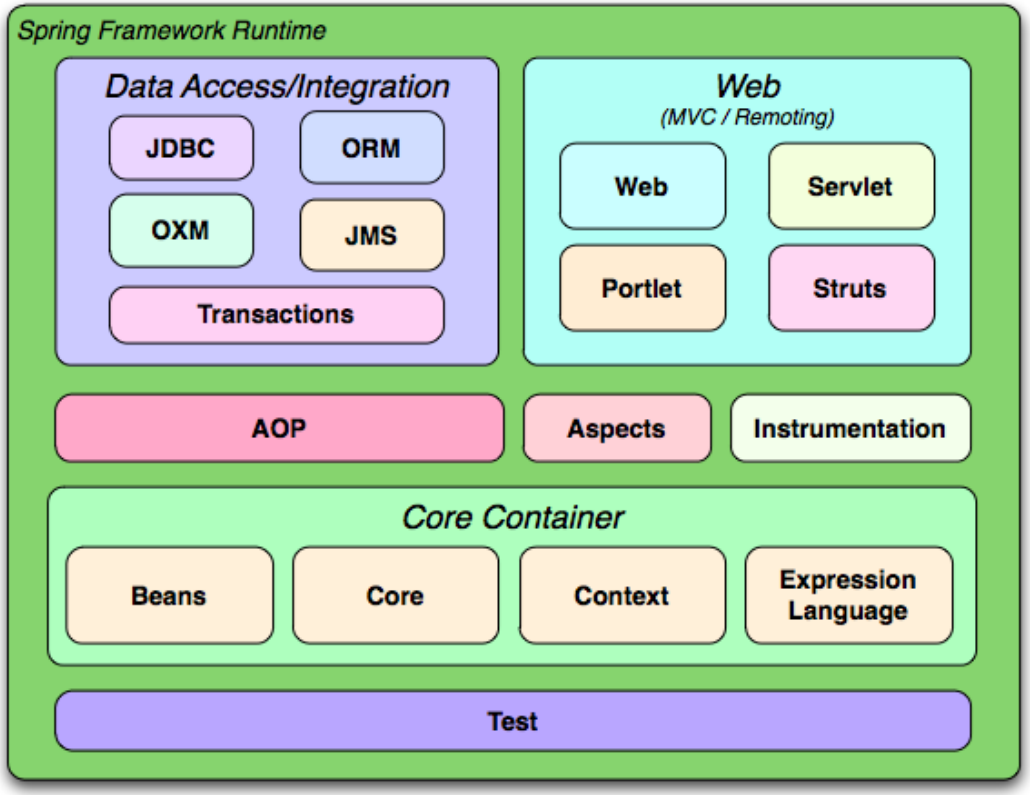
- Jednostavnija konfiguracija
- Korišćenje XML šeme
- Korišćenje anotacija
- Poboljšane biblioteke tagova
- Isključivanje klasa i metoda koje nije poželjno koristiti (Deprecated)
- Podrška za nove tehnologije, ali i uklanjanje podrške (pitanje kompatibilnosti i migracije)
- Poboljšana dokumentacija...

Spring Framework

- Spring kao aplikacioni okvir
  - Osnovne karakteristike, core funkcionalnosti
- Spring kao platforma za razvoj drugih projekata
  - Spring Security
  - Spring Data
  - Spring Social
  - Spring .NET
  - ...

Spring Framework

- Trenutno aktuelni Spring 3.2, Spring 4.x, Spring 5.x
- Organizacija Spring Framework-a (moduli):



Spring Framework

- Osnovni modul Spring Framework-a je Core
- Inverson of Control (IoC), odnosno Dependency Injection (DI)
- Martin Fowler: http://martinfowler.com/articles/injection.html
- IoC predstavlja generalni princip koji omogućava otklanjanje zavisnosti komponenti
- Zajednička k-ka, odnosno suština “laganih” (lightweight) kontejnera
- Kontejneri sa objektima rade na nenametljiv (non-invasive) način

Spring Framework

- Inversion of Control (IoC) je generalni princip kojim se otklanja međusobna zavisnost komponenti na nivou implementacije, čime se ostvaruje tzv. izolacija komponenti
- IoC princip je poznat po nazivu Holivudski princip (Hollywood principle - don't call us, we'll call you)
- Entiteti modela postaju “komponente” - celine koje se mogu konfigurisati, koje su razdvojene i nezavisne jedna od druge

Spring Framework

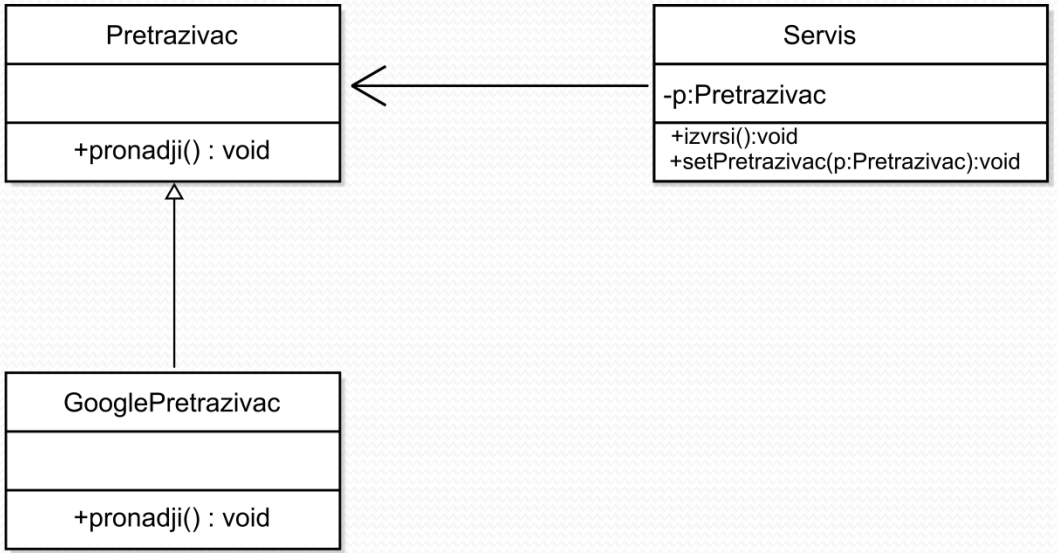
- Softverski sistem kao skup komponenti
- Komponente na određeni način sarađuju, zavise jedna od druge (u toku izvršenja aplikacije)
- Komponente se sastoje od klasa
- Klase su povezane
- Dobijanje reference na zavisne objekte
- Odgovornost?

Spring Framework

- Problem: čvrsta povezanost i direktna zavisnost
- Nemogućnost konfiguracije i zavisnost na nivou implementacije

Spring Framework

- IoC pristupa problemu inverzno
- Slaba zavisnost komponenti (npr. na nivou interfejsa)
- Komponente ne dobavljaju druge komponente koje su im potrebne
- Zavisnosti se deklarišu, konkretno se ostvaruju spolja



Spring Framework

- Minimalna zavisnost između aplikacije i kontejnera
- Razrešavanje zavisnosti između komponenti aplikacije
- Korisnička klasa koristi komponentu koji neko drugi prethodno podešava
- Nije odgovornost korisničke klase da dođe do komponente, već je to odgovornost IoC kontejnera

Spring Framework

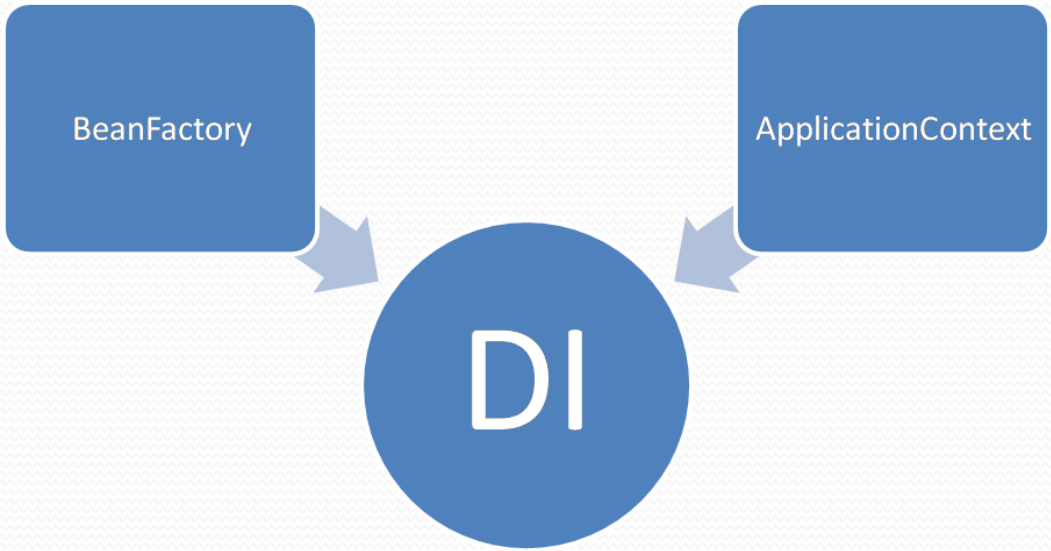
- Podela odgovornosti
- Putem IoC zavisnosti se **definišu**
- Putem DI zavisnosti se **realizuju**
- Spring je jedan od framework-a koji obezbeđuje ove funkcionalnosti
  - Java EE CDI (Context and Dependency Injection) Framework
  - Google Guice
  - Google Dagger
  - ...

Spring Framework

- Podešavanje projekta
  - Java JDK (preporučljivo 1.8 ili novija)
  - Java IDE, npr. NetBeans ili Eclipse (nije neophodno)
  - Spring
    - Biblioteke sping-context, sping-core, sping-beans, sping-asm, sping-expression
    - Biblioteka commons-logging
- Biblioteke je potrebno postaviti na classpath putanju
- Moguće je korišćenje pojedinih modula
- Moguće je korišćenje dodatka za razvojno okruženje

Spring Framework

- Implementacija DI je realizovana kroz dva interfejsa koja su srž IoC kontejnera:
  - org.springframework.beans.factory.BeanFactory
  - org.springframework.context.ApplicationContext



Spring Framework

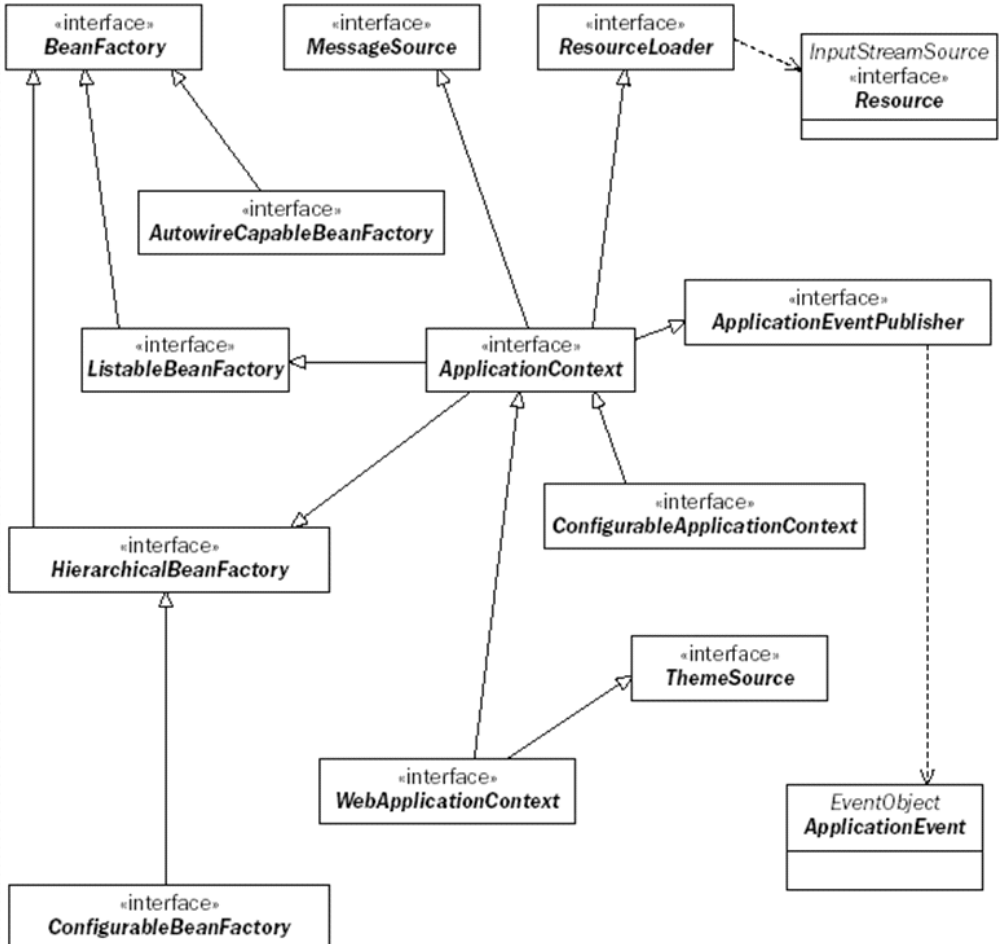
- Interfejs BeanFactory – konfiguracija okvira i osnovne funkcionalnosti
- Interfejs ApplicationContext – dodatne funkcionalnosti, izveden je iz prvog i predstavlja njegov nadskup
  - npr. kontekst aplikacionog sloja WebApplicationCotext koji se koristi u web aplikacijama

Spring Framework

```
package org.springframework.beans.factory;
import org.springframework.beans.BeansException;

public interface BeanFactory {
String FACTORY_BEAN_PREFIX = "&";
Object getBean(String name) throws BeansException;
    Object getBean(String name, Class requiredType) throws BeansException;
    Object getBean(String name, Object[] args) throws BeansException;
    boolean containsBean(String name);
boolean isSingleton(String name) throws NoSuchBeanDefinitionException;
    boolean isPrototype(String name) throws NoSuchBeanDefinitionException;
boolean isTypeMatch(String name, Class targetType) throws
   NoSuchBeanDefinitionException;
    Class getType(String name) throws NoSuchBeanDefinitionException;
    String[] getAliases(String name);
}
```

Spring Framework

- Odnos interfejsa BeanFactory i ApplicationContext



Spring Framework

- Termin bean
- Konfigurisanje i pokretanje kontejnera
- Konfiguracioni metapodaci najčešće u obliku XML datoteka ili anotacija
- applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:p="http://www.springframework.org/schema/p"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

   <bean id="helloWorld" class="hello.HelloWorld">
      <!-- konfiguracija bean-a -->
   </bean>

   <!-- konfiguracija ostalih bean-ova -->
</beans>
```

Spring Framework

- Definisanje bean-a korišćenjem DTD ili XSD šeme (preporuka)

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:p="http://www.springframework.org/schema/p"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
     http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

    <bean id="..." class="...">
       <property name="..." value="..."/>
    </bean>

</beans>
```

Spring Framework

- Pokretanje kontejnera – instanciranjem klase koja realizuje BeanFactory ili ApplicationContext interfejs
- Primer:

```
public class Main {
   public static void main (String[] args){
      ApplicationContext appContext = new
    FileSystemXmlApplicationContext("applicationContext.xml");

      //nastavak programa...
   }
}
```

Spring Framework

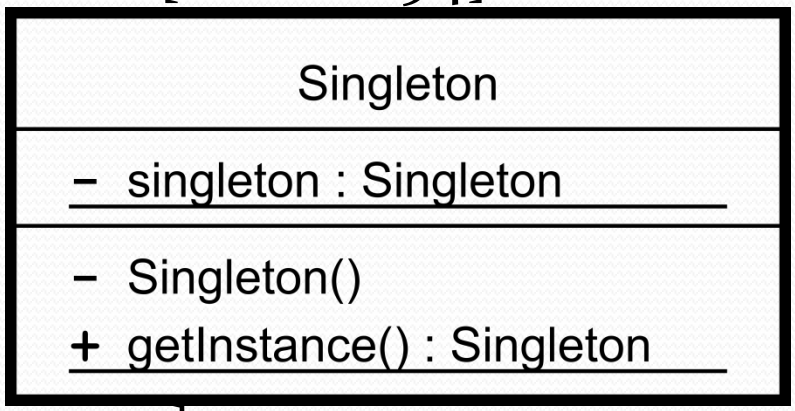
- Način inicijalizacije bean-ova:
  - Inicijalizacija pozivanjem konstruktora
    - Ukoliko se eksplicitno ne naglasi drugačije, poziva se podrazumevani konstruktor a nakon toga odgovarajuća set metoda
    - Ukoliko postoji jedan ili više contructor-arg elemenata, poziva se odgovarajući konstruktor
  - Inicijalizacija pozivanjem statičke factory metode (static factory method)
  - Inicijalizacija pozivanjem factory metode (Instance factory method)

Spring Framework

- Opseg:
  - singleton
  - prototype
  - request
  - session
  - global session
- U Spring 1.x postoje samo singleton i prototype (npr. singleton=”true”)
- Upravljanje kreiranjem i uništavanjem

Spring Framework

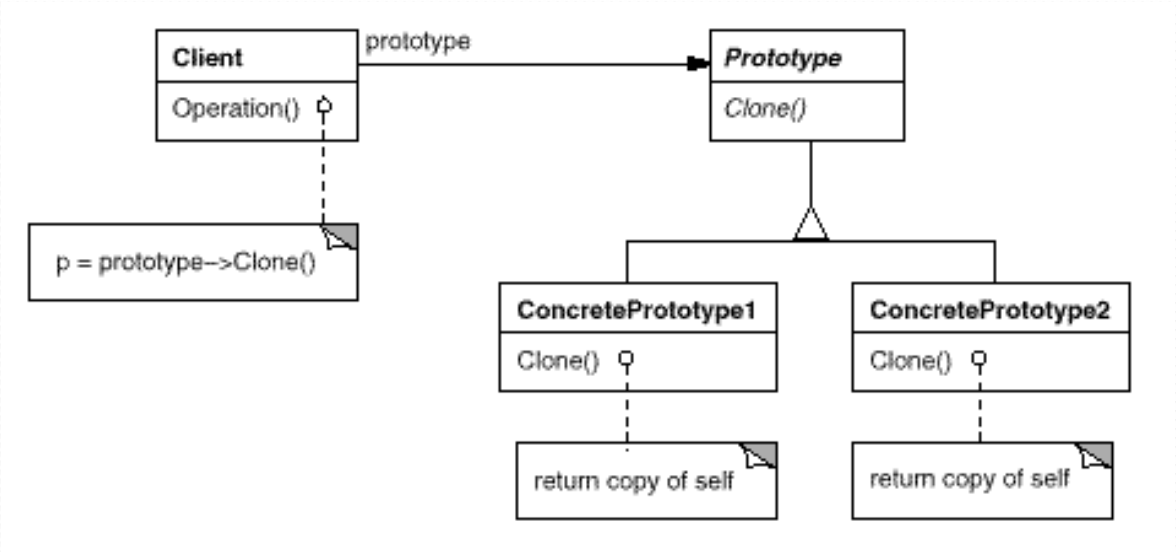
- Singleton - obezbeđuje klasi samo jedno pojavljivanje i globalni pristup do nje [Gamma94]
- Singleton struktura [Gamma94]



- Učesnici [Gamma94]
  - Singleton – definiše getInstance() operaciju koja omogućava klijentima pristup do njenog jedinstvenog pojavljivanja

Spring Framework

- Prototype - određuje (specificira) vrste objekata koje će biti kreirane korišćenjem prototipskog pojavljivanja i kreira nove objekte kopiranjem tog prototipa [Gamma94]
- Prototype struktura [Gamma94]



Spring Framework

- Prototype učesnici [Gamma94]
  - Prototype – deklariše interfejs za sopstveno kloniranje
  - ConcretePrototype – implementira operaciju za sopstveno kreiranje
  - Client – kreira novi <<objekat proizvod>> zahtevajući od prototipa da se klonira

Spring Framework

- Pored eksplicitnog definisanja zavisnosti bean-ova, Spring Framework nudi i mogućnost automatkog povezivanja korišćenjem autowire atributa bean elementa
- Povezivanje korišćenjem refleksije (npr. na osnovu tipa ili imena)
- Na globalnom nivou ili za pojedinačni bean
- Pored konfiguracije korišćenjem XML dokumenata moguće je podešavanja obaviti i korišćenjem anotacija u programskom kodu

Ključne karakteristike

- Navedene karakteristike predstavljaju sastavni deo principa objektno orijentisanog projektovanja softvera [Martin96]
  - Princip jedne odgovornosti (eng. The Single Responsibility Principle)
  - Princip otvoreno-zatvoreno (eng. The Open Closed Principle)
  - Liskov princip zamene (eng. The Liskov Substitution Principle)
  - Princip izdvajanja interfejsa (eng. The Interface Segregation Principle)
  - Princip inverzije zavisnosti (eng. The Dependency Inversion Principle)
  - Princip dodavanja zavisnosti (eng. Dependency injection principle)
- SOLID principi objektno-orijentisanog projektovanja softvera

Ključne karakteristike

Princip jedne odgovornosti (eng. Single responsibility principle)

- U procesu objektno-orijentisanog razvoja softvera ne bi trebao da postoji više od jednog razloga za izmenu posmatrane klase [Martin96]
- Klasa treba da ima jednu odgovornost u posmatranom softverskom sistemu

Ključne karakteristike

Princip otvoreno-zatvoreno (eng. Open-closed principle)

- Entiteti (npr. klase, moduli, funkcije itd.) u procesu objektno-orijentisanog razvoja softvera trebaju da budu otvoreni za proširenja ali i zatvoreni za modifikaciju [Martin96]

Ključne karakteristike

Liskov princip zamene (eng. Liskov substitution principle)

- Funkcije koje koriste pokazivače ili reference ka osnovnim klasama (tj. nadklasama) moraju biti u mogućnosti da koriste i objekte izvedenih klasa (tj. podklasa) [Martin96]

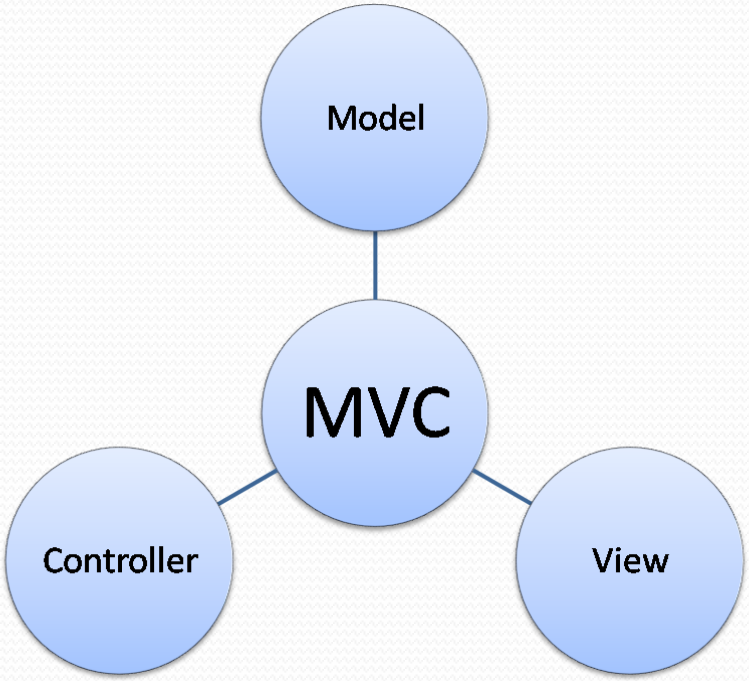
Ključne karakteristike

Princip izdvajanja interfejsa (eng. Interface segregation principle)

- Specifikacija funkcija posmatranog modula
- Klijenti ne trebaju biti zavisni od interfejsa koje ne koriste [Martin96]

Spring Framework

- Spring Web MVC okvir za razvoj web aplikacija
- MVC uzor makroarhitekture
- Tri ključne komponente

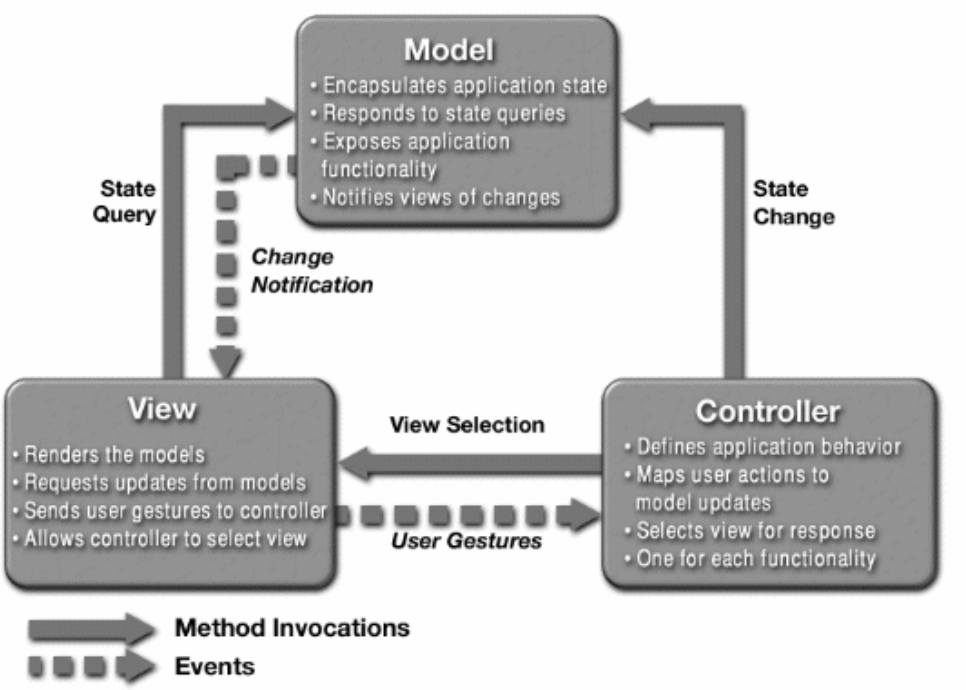


Spring Framework

- Model – sadrži strukturu i operacije poslovnog sistema (podatke i operacije za obradu podataka)
- Pogled (View) – obezbeđuje korisnički interfejs preko koga korisnik komunicira sa sistemom
- Kontroler (Controller) – zadužen za upravljanje izvršenjem sistemskih operacija
  - prihvata zahtev od klijenta, poziva operaciju definisanu u modelu i kontroliše njeno izvršenje

Spring Framework

- Komponentno orijentisan pristup
- Spring MVC je zasnovan na MVC uzoru makroarhitekture



Spring Framework

- Konfiguracija konteksta aplikacije (ApplicationContext) u web.xml
- Korišćenje klase u zavisnosti od podržane servlet specifikacije

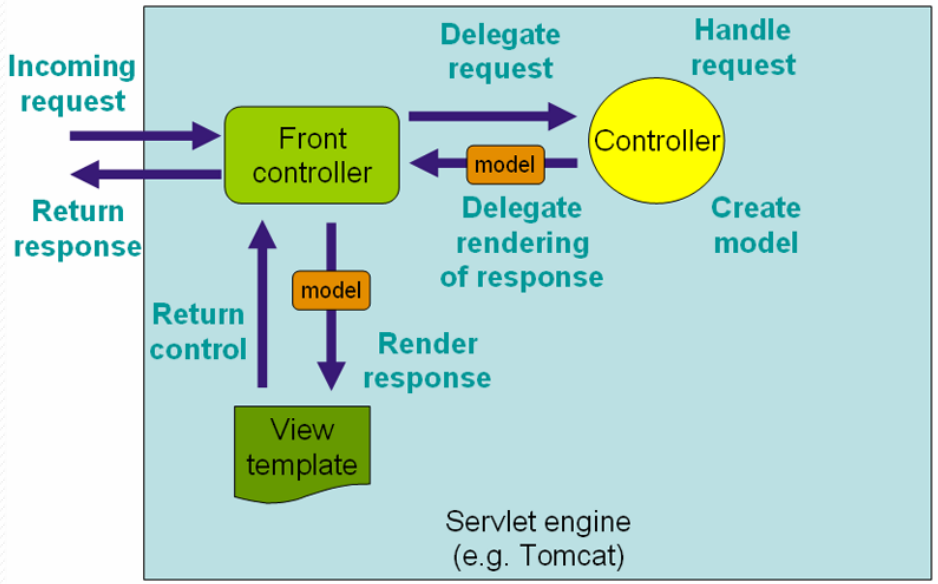
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
    app_2_5.xsd">
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/applicationContext.xml</param-value>
  </context-param>
  <listener>
    <listener-class>

    org.springframework.web.context.ContextLoaderListener
              </listener-class>
  </listener>
    <!-- ostatak konfiguracije... -->
  </web-app>
```

Spring Framework

- DispatcherServlet kao centralni kontroler
- Inicijalno procesiranje klijentskog zahteva
- Jedinstvena tačka ulaska u web aplikaciju
- Kompletno integrisan u IoC kontejner

Spring Framework

- Proces obrade zahteva [Spring]



Spring Framework

- Deklaracija u web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
  <servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>2</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>*.htm</url-pattern>
  </servlet-mapping>
    <!-- ostatak konfiguracije... -->
  </web-app>
```

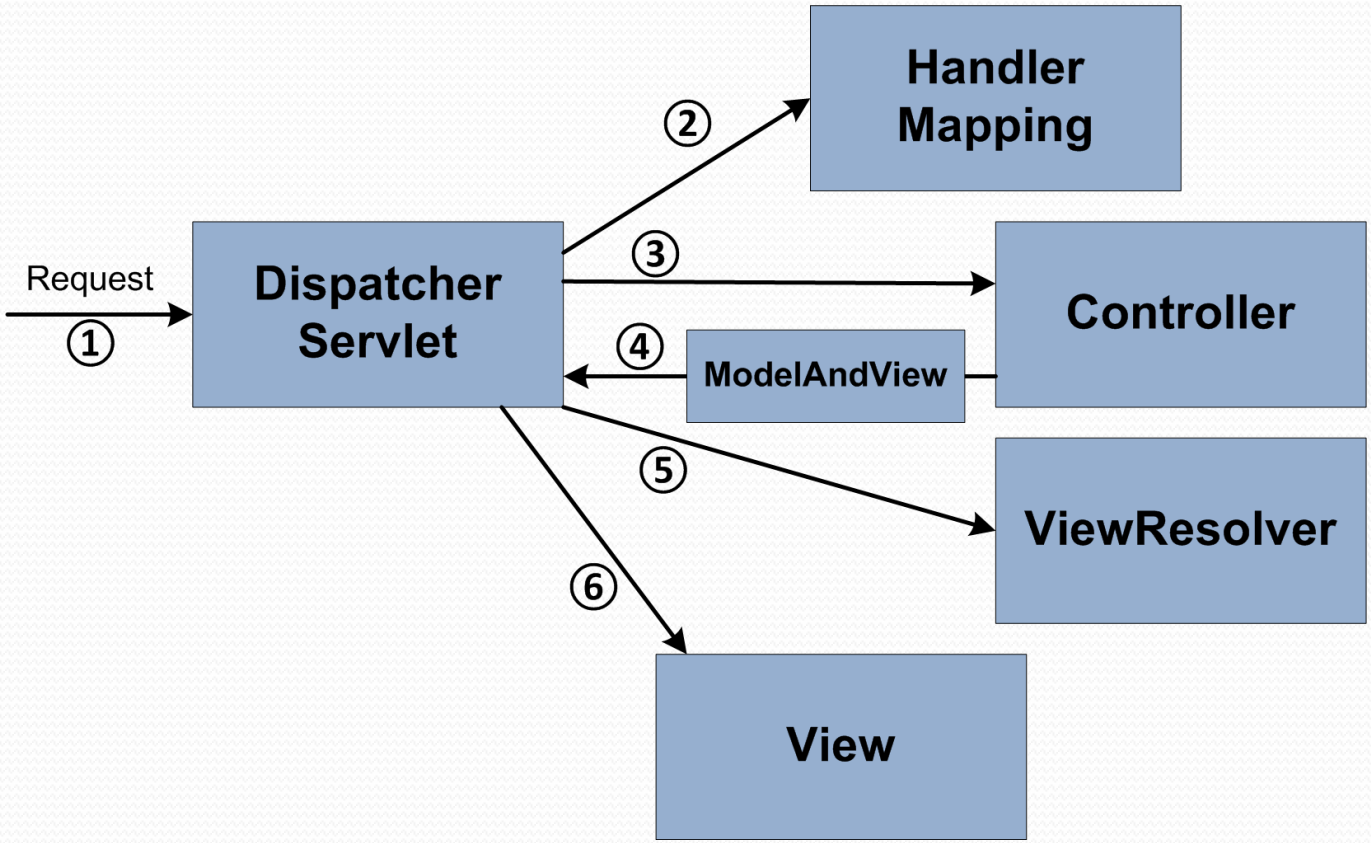
Spring Framework

- Ukoliko se koristi podrazumevana konfiguracija, okvir traži datoteku [ime-servleta]-servlet.xml u okviru WEB-INF dir.
- Definicija lokalnih bean-ova
- Prekrivanje bean-ova
- Primer: dispatcher-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:p="http://www.springframework.org/schema/p"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
     http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

    <!-- konfiguracija bean-ova -->
</beans>
```

Spring Framework

- Tok obrade zahteva [Walls14]



Spring Framework

- Nakon obrade zahteva, kao povratna vrednost vraća se instanca klase ModelAndView
- Model predstavlja mapu (java.util.Map) sačinjenu od imena objekata kao ključeva, i samih objekata kao vrednosti
- Pogled može da bude implementacija interfejsa org.springframework.web.servlet.View ili logičko ime

Spring Framework

- Moguće korišćenja više view resolver-a njihovim ulančavanjem
- Različite tehnologije kao realizacija pogleda: HTML, JSP...
- Preporuka je da se stranice nalaze unutar WEB-INF dir.
- Onemogućava se direktan pristup do stranica
- Pristup je moguć preko kontrolera

Spring Framework

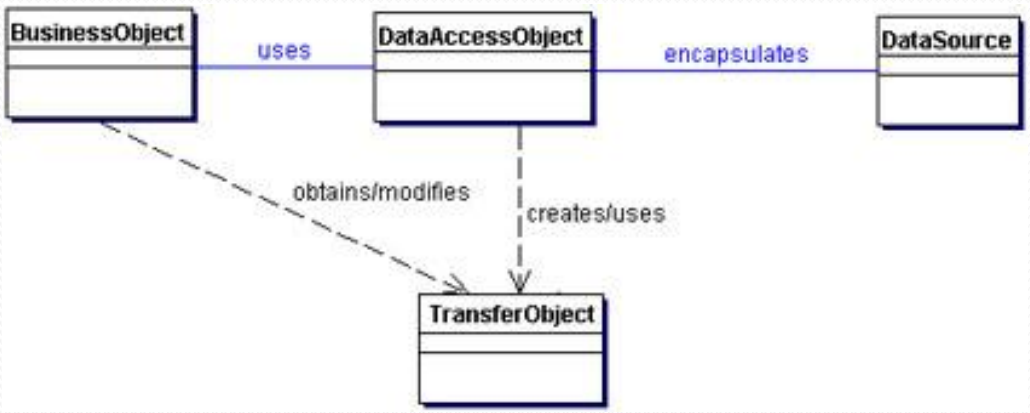
- JDBC – Java tehnologija za rad sa relacionim bazama podataka
- JDBC je nezavisan u odnosu na SUBP
- JDBC API i upravljački program (driver) za konkretni SUBP
- Jednom napisan program može se izvršavati nad različitim SUBP
  - Preduslov: postojanje drajvera
- Nedostatak: upravljanje izuzecima

Spring Framework

- Skup DAO (Data Access Object) klasa za jednostavan pristup tehnologijama
- Razdvajanje poslovne logike od dela zaduženog za perzistentnost
- Definisanje interfejsa sa operacijama koje perzistentni okvir nudi uz sakrivanje realizacije

Spring Framework

- DAO (Data Access Object) uzor
- Sakrivanje detalja implementacije izvora podataka (data source) od klijenata. Interfejs izložen klijentu se ne menja prilikom promene implementacije (različiti mehanizmi skladištenja)
- Struktura



Spring Framework

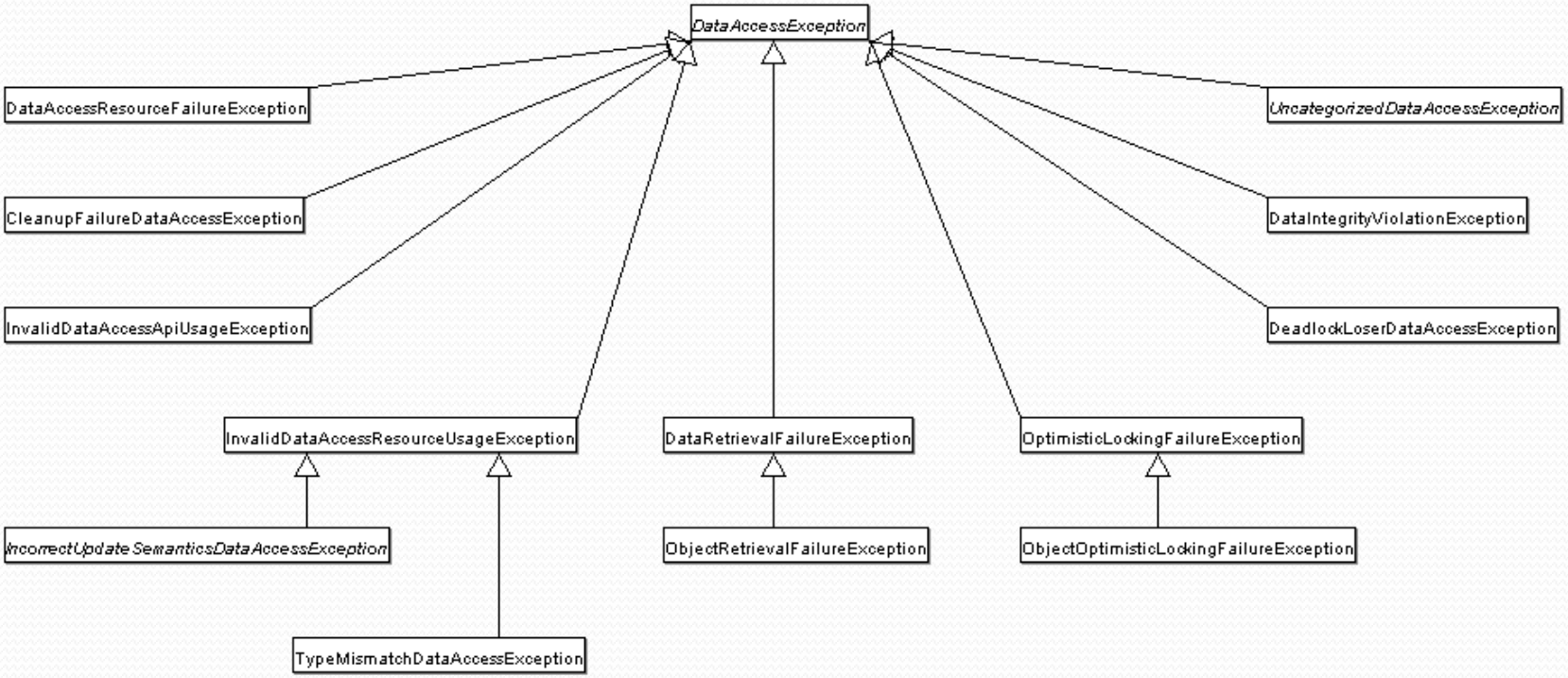
- DAO (Data Access Object) uzor učesnici
  - BusinessObject – objekat koji predstavlja klijenta koji zahteva pristup izvoru podataka
  - DataAcessObject (DAO) – centralni objekat ovog uzora. Operacije nad bazom podataka se delegiraju ovom objektu
  - DataSource – implementacija izvora podataka (npr. Hibernate sesija)
  - TransferObject – nosilac podataka

Spring Framework

- Obrada izuzetaka u Spring JDBC modulu
- JDBC API – metode najčešće generišu izuzetak java.sql.SQLException
- SQLException je **proveravan** izuzetak
- Spring JDBC modul – svi izuzeci su potklasa org.springframework.dao.DataAccessException
- Izveden je iz klase RuntimeException
- DataAccessException je **neproveravan** izuzetak

Spring Framework

- Hijerarhija izuzetaka [Spring]



Spring Framework

- Kako znati koji izuzetak treba generisati?
  - Ispituju se atributi errorCode i SQLState izuzetka SQLException (specifičnost svakog SUBP)
  - Preslikavanja se definišu u sql-error-codes.xml (paket org.springframework.jdbc.support)
- Izuzetak SQLException → podklasa izuzetka DataAccessException
- Pošto su neproveravani, ovi izuzeci se ne moraju obrađivati

Spring Framework

- Integracija sa ORM okvirima
- Preslikavanje iz relacione baze podataka u Java objekte i obrnuto
- Nezavisnost od SUBP
- Jednostavna migracija SUBP
- Podrška za većinu ORM okvira
  - Hibernate, JPA, iBatis, TopLink…
  - Uklanja se podrška za starije verzije tehnologija!

Spring Framework

- Koncepti u tehnologijama za pristup podacima

Koncept	JDBC	Hibernate	JPA
Resource	Connection	Session	EntityManager
Resource Factory	DataSource	SessionFactory	EntityManagerFactory
Exception	SQLException	HibernateException	PersistenceException

Zaključak

- Problemi u procesu razvoja Java EE aplikacija
- Ciljevi Spring framework-a
- Verzije i stalna unapređenja Spring framework-a
- Ključne karakteristike
  - Inversion of Control
  - Dependency Injection
  - SOLID principi objektno-orijentisanog projektovanja softvera
- Spring MVC

Reference

- [EJB2] Enterprise JavaBeans Specification, Version 2.0
- [Gamma94] E. Gamma et al, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, 1994.
- [Johnson] R. Johnson, The Spring Framework
- [Johnson03] R. Johnson, Expert One-On-One J2EE Design and Development, Wrox, 2002.
- [Martin96] R.C. Martin, The Principles of OOD, 1996.
- [Spring] R. Johnson et al, Spring Framework Reference Documentation, 2016.
- [Walls14] C. Walls, Spring in Action, Fourth Edition, Manning, 2014.

Spring Framework

Univerzitet u Beogradu
Fakultet organizacionih nauka
Miloš Milić