Spring Framework

Uvod

- Spring je najpopularniji radni okvir za razvoj Java web aplikacija[1]

Java

Framework	Score
Spring	89
JSF	81
Google Web Toolkit	77
Dropwizard	66
Vert.x	66
Struts	64
Vaadin	63
Wicket	63
Restlet	55
Tapestry	53
Ninja	53
ZK	50

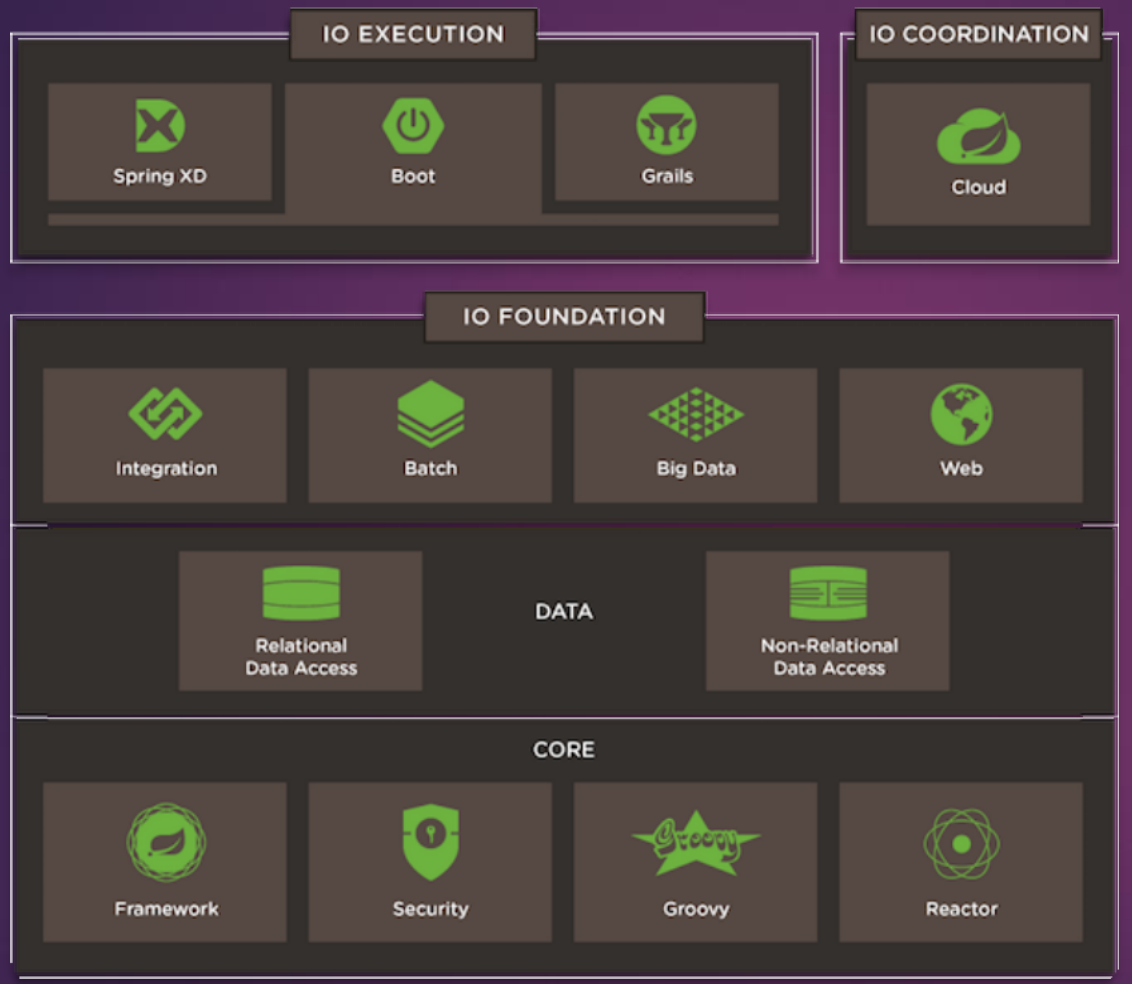
[1] https://hotframeworks.com/languages/java

Literatura

- Dokumentacija
  - https://spring.io
  - https://docs.spring.io/spring/docs/5.1.1.RELEASE/spring-framework-reference/
  - https://docs.spring.io/spring-boot/docs/2.0.6.RELEASE/reference/htmlsingle/
- Knjige
  - C. Walls, *Spring in Action*, Manning
  - C. Ho, R. Harrop, C. Schaefer, *Pro Spring*, Apress

Prednosti

- Spring radni okvir omogućava programerima da razvijaju enterprise aplikacije korišćenjem *POJO* klasa
- Organizovan je kroz module
- Koristi poznate koncepte uvedene mnogo ranije
- Testiranje aplikacija pisanih u Springu je jednostavno jer je celo okruženje potrebno za testiranje integrisano u radni okvir
- *Inversion of Control* (IoC) kontejner je lagan u odnosu na npr. EJB Java kontejner
- Spring pruža mogućnost korišćenja dobro razvijenog transakcionog menadžmenta



Spring IO platforma

Spring IO platforma - Core

Spring Framework	Nudi osnovnu podršku za dependency injection, transakcioni menadžment, web aplikacije, rad sa podacima iz baze podataka, razmenu poruka, itd.
Spring Security	Pruža podršku za zaštitu aplikacije različitim mehanizmima za autentifikaciju i auzorizaciju
Groovy	Donosi napredna svojstva dinamičkog jezika na JVM
Reactor	Nudi osnovu za razvoj reaktivnih aplikacija na JVM

Spring IO platforma - Data

- Spring IO nudi rešenja za rad sa različitim tipovima baza podataka i datotekama (dokument, graf, ključ-vrednost, relacione baze ili nestrukturirani fajlovi)

Spring IO platforma - Workloads

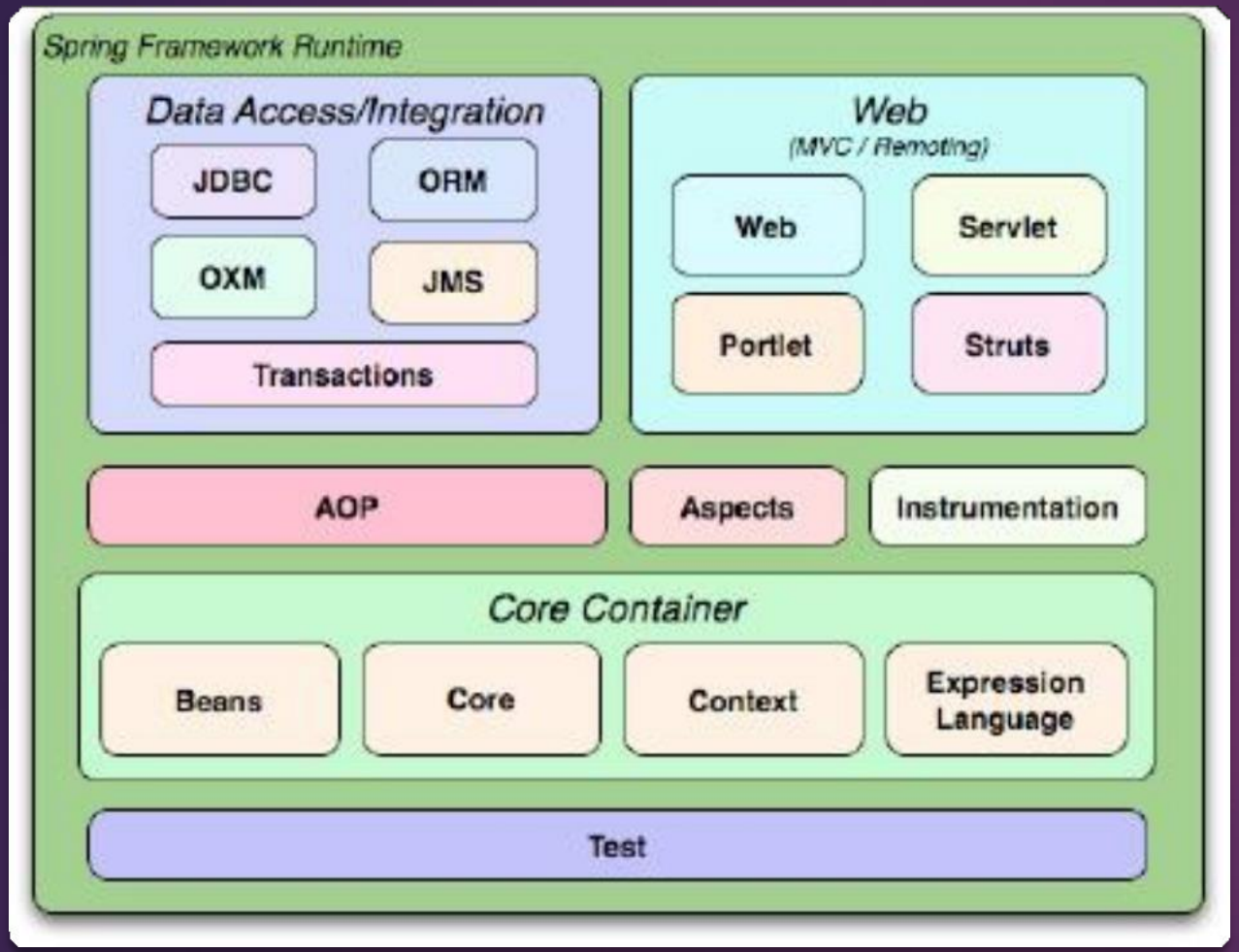
Integration	Channels, Adapters, Filters, Transformers
Batch	Jobs, Steps, Readers, Writers
Big data	Ingestion, Export, Orchestration, Hadoop
Web	Controllers, REST, WebSocket

Spring IO platforma - Boot

- *Spring Boot* olakšava konfiguraciju Spring projekata sa dosta funkcionalnosti koje su ugrađene, a koje bi se inače zasebno konfigurisale
  - Ugrađeni *Tomcat* ili *Jetty* serveri bez potrebe za zasebnim WAR fajlom
  - Mogućnost korišćenja "starter" biblioteka za olakšanu inicijalnu konfiguraciju
  - Automatska konfiguracija Spring i stranih (3rd party) biblioteka
  - Ugrađene su biblioteke za različite metrike, zdravlje aplikacije, tid.
  - Nema potrebe za XML konfiguracijom
- Starter projekti se mogu kreirati na https://start.spring.io
- Za kreiranje Spring starter projekata i njihov dalji razvoj mogu se koristiti Eclipse sa STS pluginom ili Spring Tool Suite IDE

Spring radni okvir

- Zamišljen je kao “lagan” radni okvir što podrazumeva da se od obične Java aplikacije vrlo lako može napraviti Spring Java aplikacija i obrnuto
- Osnovna svojstva Spring radnog okvira mogu se iskoristiti za razvoj bilo kakve Java aplikacije, ali najveća podrška postoji za izradu web aplikacija na Java EE platformi
- Omogućava izgradnju aplikacija iz POJO (*Plain Old Java Object*) objekata i neinvazivnu primenu enterprise servisa nad njima



Komponente Spring radnog okvira

Spring radni okvir – Core container

- **Core** modul sadrži osnovne delove radnog okvira uključujući IoC i upravlja životnim ciklusom objekata
- **Bean** modul nudi *BeanFactory* što predstavlja implementaciju *factory* šablona
- **Context** modul izgrađen je na osnovama prethodna dva modula i služi za čitanje konfiguracije aplikacije i pristup svakom konfigurisanom objektu

- **Expression Language** modul nudi moćan DSL (*Domain Specific Language*) za upite i manipulaciju objektima kojim kontejner upravlja prilikom izvršavanja aplikacije

Spring radni okvir – Data access

- **JDBC** modul nudi viši sloj apstrakcije koji zamenjuje mukotrpnu programsku komunikaciju sa bazom podataka
- **ORM** modul nudi integraciju sa popularnim objektno-relacionim maperima (*JPA, JDO, Hibernate*, itd)
- **OXM** modul nudi viši sloj apstrakcije koji podržava objektno-XML mapiranje za *JAXB, Castor, XMLBeans, XStream*, itd.
- **JMS** modul sadrži svojstva za kreiranje i razmenu poruka
- **Transaction** modul podržava programsko i deklarativno upravljanje transakcijama za klase koje implementiraju jasno definisane interfejse za sve POJO objekte

Spring radni okvir – Web

- **Web** modul nudi podršku za različite aspekte razvoja web aplikacija poput inicijalizacije IoC kontejnera pomoću servleta, kreiranja web aplikativnog koneksta, postavljanja fajlova na server, itd.
- **Web-Servlet** modul sadrži Spring MVC (*model-view-controller*) implementaciju za web aplikacije
- **Web-Portlet** modul sadrži MVC implementaciju koja se koristi u portlet okruženju
- **Web-Struts** modul sadrži klase za podršku pri integraciji *Struts* web sloja u Spring aplikaciju

Spring radni okvir – Ostali moduli

- **AOP** modul sadrži implementacije koncepata aspektno orijentisanog programiranja
- **Aspects** modul nudi integraciju sa *AspectJ*
- **Instrumentation** modul nudi podršku za različite *classloadere* koji se koriste u okviru različitih aplikativnih servera
- **Testing** modul omogućava jedinično i integraciono testiranje zasnovano na *JUnit* i *TestNG* bibliotekama

Inversion of Control (IoC)

- *Inversion of Control* (IoC) je tehnika kojom se povezivanje objekata vrši u toku izvršavanja (*runtime*) a ta veza nije poznata u toku kompaliranja (*compile time*)
- Da bi kontejner mogao da izvrši povezivanje, objekti moraju imati odgovarajuće uputstvo
- Jedna forma IoC je dependency injection (DI)

Inversion of Control (IoC)

- Spring kontejner je srž Spring radnog okvira
- Spring kontejner koristi *dependency injection* (DI) da upravlja komponentama koje čine aplikaciju
- Kontejner će kreirati objekte, uvezati ih, konfigurisati i upravljati njihovim životnim ciklusom od kreiranja do uništenja
- Kontejner na osnovu konfiguracije koja može biti iskazana XML fajlovima, Java anotacijama ili Java kodom zna šta da radi

DI kontejneri

- Spring *BeanFactory* kontejner
  - Najjednostavniji kontejner koji pruža osnovnu podršku za DI
  - Postoji nekoliko implementacija BeanFactory interfejsa od kojih je najčešće korišćena XmlBeanFactory klasa
- Spring *ApplicationContext* kontejner
  - Uključuje sve funkcionalnosti koje pruža BeanFactory i preporučuje se za korišćenje
  - Podržava više funkcionalnosti kao npr. čitanje tekstualnih poruka iz *properties* fajlova, mogućnost objave događaja zainteresovanom osluškivačima (*event listeners*)

Spring beans

- Objekti koji formiraju aplikaciju, a koji se nalaze u nadležnosti Spring kontejnera zovu se zrna (*beans*)
- Klasa postaje bean dodavanjem konfiguracije u vidu Java anotacija ili XML metapodataka koje su potrebne Spring kontejneru da bi znao:
  - Kako da kreira bean
  - Detalje o životnom ciklusu beana
  - Zavisnosti koje bean poseduje

Spring beans metapodaci

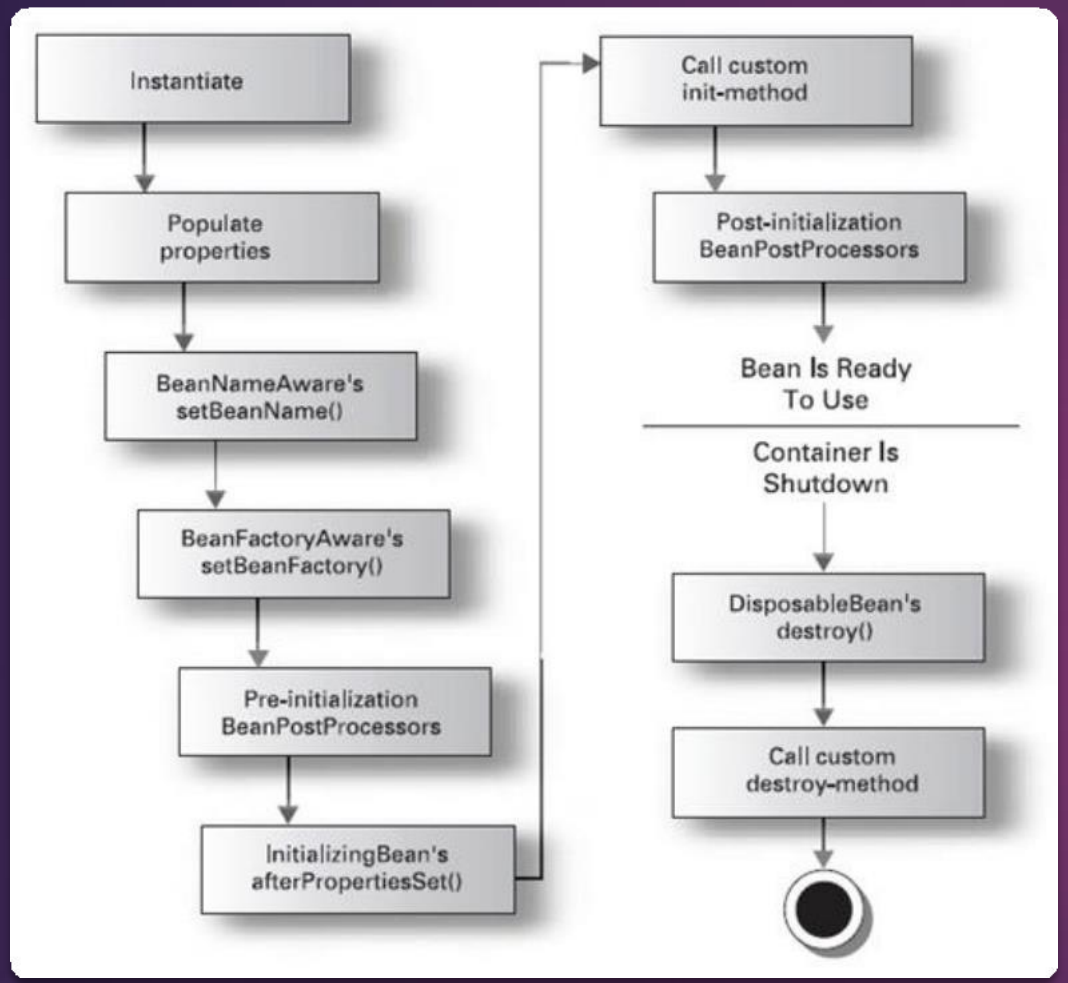
Svojstvo	Opis
class	Klasa koja se koristi za kreiranje beana
name	Jedinstveni identifikator beana
scope	Opseg objekta nastalog od bean definicije
autowiring mode	Režim uvezivanja objekata u bean
lazy-initialization mode	Davanje instrukcija IoC kontejneru da kreira instancu beana kada se zatraži, a ne pri pokretanju aplikacije
constructor-arg	Služi za povezivanje zavisnosti kroz konstruktor
properties	Služi za povezivanje zavisnosti kroz set metodu
initialization method	Callback metoda koja se poziva nakon što je kontejner odradio setup beana i povezivanje sa zavisnim objektima
destruction method	Callback metoda koja se poziva pre nego što se bean uništi

Opseg Spring beana

- Podrazumevano ponašanje beana je *singleton* kada se uvek kreira samo jedna instanca i ona se injektuje kada je tražena
- Ako to nije željeno ponašanje, opseg se može eksplicitno definisati

Opseg Spring beana

Svojstvo	Opis
singleton	Jedna instanca po Spring kontejneru (podrazumevani opseg)
prototype	Nova instanca se kreira svaki put kada se objekat injektuje ili kada se traži iz konteksta aplikacije
request	Nova instanca za svaki HTTP zahtev
session	Nova instanca za svaku HTTP sesiju



Životni ciklus Spring beana

Životni ciklus Spring beana

- Kada se bean instancira, potrebno je izvršiti inicijalizaciju kako bi postao upotrebljiv
- Kada bean više nije potreban, potrebno je očistiti tragove njegovog prisustva

Životni ciklus Spring beana

- Postoje tri mehanizma konfiguracije inicijalizacije i uništavanja beana
- Inicijalizaciju može izvršiti
  - Metoda anotirana @PostConstruct anotacijom
  - Metoda afterPropertiesSet() koju propisuje interfejs *InitializingBean* koji komponenta implementira
  - Proizvoljna init() metoda uvezana preko XML konfiguracije
- Uništavanje može izvršiti
  - Metoda anotirana @PreDestroy anotacijom
  - Metoda destroy() koju propisuje interfejs *DisposableBean* koji komponenta implementira
  - Proizvoljna destroy() metoda uvezana preko XML konfiguracije

Tipovi DI

Tip	Opis
Kroz konstruktor	DI se ostvaruje kada kontejner pozove konstruktor beana koji kao parametre ima zavisne komponente
Kroz set metodu	DI se ostvaruje kada kontejner pozove set metodu beana posle pozivanja konstruktora bez parametara
Kroz atribut	Ostvarivo isključivo pomoću anotacija za razliku od preostalih načina

Konfiguracija bazirana na anotacijama

- Od verzije Spring 2.5 postoji mogućnost lakše i čistije konfiguracije korišćenjem anotacija umesto XML fajlova
- Anotacije se dodaju na postojeći Java kod
- Anotiranjem Java klase anotacijom @Configuration ta klasa postaje konfiguraciona klasa iz koje Spring kontejner čita uputstva
- @Bean anotacija govori Springu da u konfiguracionoj klasi metoda koja je anotirana tom anotacijom treba da bude registrovana kao bean

Konfiguracija bazirana na anotacijama

- Konkretne klase mogu biti anotirane posebnim anotacijama i tako registrovane kao komponente umesto korišćenja @Bean anotacije u Java konfiguracionoj klasi

Anotacija	Opis
@Component	Stereotip za generalnu upotrebu (sve u Spring kontejneru je bean tj. komponenta)
@Controller	Označava da je klasa Spring MVC kontroler
@Repository	Označava da je klasa repozitorijum podataka
@Service	Označava da klasa sadrži neku biznis logiku
Bilo koja proizvoljna anotacija izvedena iz @Component	/

DI anotacije

- @Autowired, @Inject, @Resource – rade DI povezivanje i mogu se primeniti na konstruktor, set metode ili atribute
- @Autowired je Spring specifična i najčešće korišćena anotacija za DI
- @Qualifier - u kombinaciji sa @Autowired može se koristiti da jednoznačno naglasi koja komponenta se injektuje (ukoliko ima više komponenti koje služe istoj svrsi)