SPRING FRAMEWORK I
SPRING BOOT

Korišćeni izvori:
https://www.petrikainulainen.net/software-development/design/understanding-spring-web-application-architecture-the-classic-way/
https://www.javatpoint.com/spring-boot-tutorial

SPRING FRAMEWORK

- okvir za razvoj Java enterprise aplikacija
- nastao još 2003 kao odgovor na kompleksnost rane J2EE (Enterprise Edition - Java sa podrškom za razvoj aplikacija u kompleksnim distribuiranim sistemima) specifikacije
- Java EE i Spring nisu suprotstavljeni - Spring se može smatrati komplementom, nadgradnjom J2EE.
  Iako ne usvaja sve J2EE specifikacije, Spring koristi pojedine delove EE specifikacije:
  - Servlet API
  - WebSocket API
  - Concurrency Utilities
  - JSON Binding API
  - Bean Validation
  - JPA
  - JMS
  - JTA/JCA.

SPRING FRAMEWORK

- Spring Framework podržava (i to se često navodi kao jedno od njegovih ključnih obeležja) Dependency Injection specifikaciju
- Koristi se i Common Annotations specifikacija.

COMMON ANNOTATIONS

- specifikacija koja omogućava da se u Java klasama koriste **anotacije**
- informacije o samom programu, ali koje ne čine deo programa. Anotacije obezbeđuju dodatne informacije o samom programu.
  - Anotacija počinje znakom ‘@’.
  - Anotacija ne menja ono što programski kod radi.
  - Anotacija pomaže da se povežu dodatni metapodaci (dodatne informacije) sa elementima programa.
  - Anotacije NISU obični komentari jer mogu da utiču na to kako kompajler tretira kod.

DEPENDENCY INJECTION

- Implementiran je pomoću **Inversion of Control** kontejner modula Spring frameworka
- Omogućava konzistentno upravljanje životnim ciklusom objekata - sam kontejner obavlja: inicijalizaciju objekata kada su potrebni, konfiguriše ih i povezuje.
- Šta ovo znači - programer ne mora više sam da poziva inicijalizaciju objekata, već to preuzima na sebe sam kontejner
- Objekti koje je kreirao i kontroliše kontejner nazivaju se ***managed objects*** ili ***beans***.
- Smanjuje se broj grešaka zbog nepostojanja pravovremene inicijalizacije i konfiguracije objekat (npr. Null pointer exceptions)
- Koje objekte kontroliše kontejner se inicijalno podešavalo (i sad može) XML fajlovima, ali je jednostavnije pomoću odgovarajućih anotacija

SPRING FRAMEWORK MODULI

- Sam Spring framework je organizovan po modulima koji obezbeđuju određene servise.

- Aplikacije koje se razvijaju koristeći Spring framework su takođe modularne, sa modulima koji (bi trebalo da) imaju jasne funkcije u celokupnoj aplikaciji

SPRING FRAMEWORK MODULI

- Spring Core Container - osnovni modul Springa koji obezbeđuje i ključne kontejnere (BeanFactory and ApplicationContext)
- Aspect-oriented programming - podrška za aspektno orijentisano programiranje
- Authentication and authorization - konfigurabilni modul koji obezbeđuje niz stadnarda, protkola, alata za bezbednost aplikacija.
- Convention over configuration - omogućava brz razvoj enterprise aplikacija primenom principa *kodiranja u skladu sa konvencijom.* Ovi pristupom se za određene stvari u aplikaciji podrazumevaju određene stvari, a programer treba samo da iskodira one stvari koje odstupaju od podrazumevanog stanja (i naravno specifičnu logiku svoje aplikacije).
- Data access - modul koji omogućava pristup relacionim bazama korišćenjem JDBC i objektno-relacionog mapiranja, kao i pristup NoSQL bazama.
- Inversion of control container - konfigurisanje komponenti aplikacije tako da kontejner upravlja životnim ciklusom odgovarajućih komponenti
- Messaging - *message listener-i* koji se konfigurišu za korišćenje poruka iz redova za poruke, nadogradnja nad standardni Java JMS:
- Model–view–controller: - obezbeđuje HTTP i servlet-bazirani okvir za razvoj web aplikacija i RESTful servisa
- Remote access framework - omogućava korišćenje RPC (remote procedure call) bazirano pozivanje udaljenih objekata
- …

OSNOVNI PRINCIPI SPRING FRAMEWORK-A

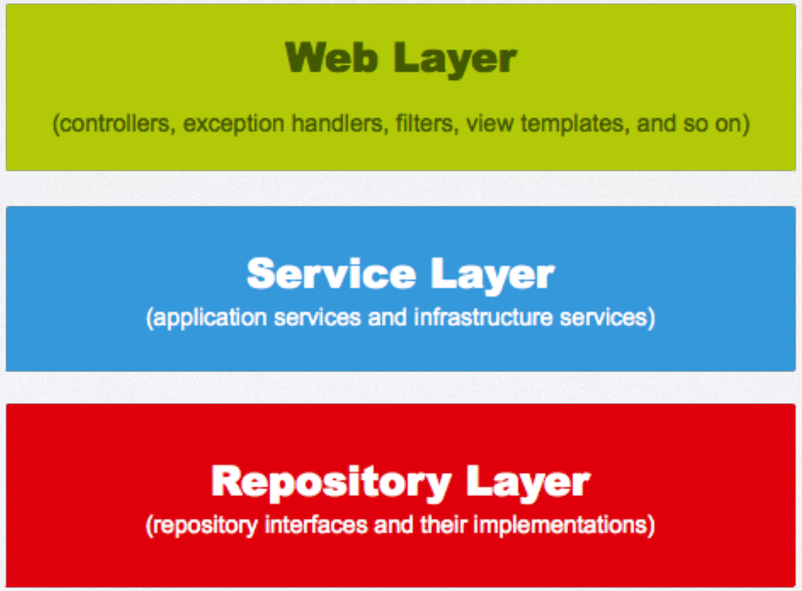
- Obezbeđivanje mogućnosti izbora na svakom nivou. Spring omogućava da se određene odluke odlože, i time omogući fleksibilniji razvoj aplikacije. Npr. moguće je zameniti sloj koji obezbeđuje perzistiranje podataka, a da to ne utiče na kod vaše aplikacije (jer svaki *persistence provider* mora da implementira isti interfejs koji obezbeđuje ovu funkcionalnost).
- Omogućava različite perspektive - Spring omogućava veliku fleksibilnost - ne pravi unapred zahteve kako ćete nešto da rešavate, rešenje zavisi od perspektive iz koje se sagledava problem.
- Zadržava kompatibilnost sa prethodnim verzijama - evolucija frameworka je pažljivo rzazvijana tako da se izbegavaju promene koje „razbijaju“ aplikaciju pisanu u prethodnu verziju. Obezbeđena je i podrška za različite verzije JDK-a i eksternih biblioteka.
- Vodi se računa o dizajnu API-ja kako bi on bio jednostavan i intuitivan za korišćenje i stabilan tokom vremena i nekoliko verzija Spring-a.
- Postavljeni su visoki standardi za kvalitet koda - održava se precizan, ažuran i razumljiv javadoc. Struktura koda je čista sa izbegnutim cirkularnim referencama između paketa.
- Upravljanje transakcijama ujedinjuje nekoliko API-ja za upravljanje transakcijama.
- Moguće je koristiti Java Management Extensions (JMX) za udaljeno upravljanje konfiguracijom
- Testiranje - podržava pisanje unit i integracionih testova

OSNOVNI PRINCIPI PRI RAZVOJU SOFTVERA

- Dizajn arhitekture softverskog rešenja je neophodan. Razvojni okviri (*frameworks)* mogu da pomognu u formiranju dobre arhitekture.
- Dva principa dobre arhitekture softverskih sistema:
  - **Princip razdvajanja nadležnosti (Separation of Concerns - SoC)** - dekomponavanje programa u distinktne komponente od kojih svaka ima jasnu nadležnost (izvršava jasnu funkcionalnost)
    1. treba utvrditi šta su to „nadležnosti“ - delovi problema koje treba rešiti
    2. odrediti koji deo koda će biti nadležan za to
  - **Težiti jednostavnosti rešenja (Keep It Simple Stupid! - KISS)**
    Većina sistema radi najbolje kada su implementirana na najjednostavniji mogući način.
- Dakle treba voditi računa o razdvajanju nadležnosti, ali istovremeno voditi računa o jednostavnosti celokupne arhitekture - izbegavati da na kraju arhitektura ima nebrojeno mnogo slojeva.

ARHITEKTURA WEB APLIKACIJA

- U načelu tri distinktna sloja zadovoljavaju osnovne potrebe arhitekture većine softverskih rešenja
- Ako govorimo o web aplikacijama:



Slika preuzeta sa https://www.petrikainulainen.net/wp-content/uploads/spring-web-application-layers.png

ARHITEKTURA WEB APLIKACIJA

- **web sloj (**web layer) - predstavlja najviši sloj web aplikacije. Njegova namena je da procesira ulazne podatke (zahteve) koje generiše korisnik (ili klijentska aplikacija) i da mu šalje odgovor. Web sloj takođe mora na neki prihvatljiv način obraditi i izuzetke koji dolaze iz ostalih slojeva aplikacije i preneti te infromacije korisniku. Pošto je ovaj sloj ulazna tačka - on mora obraditi i autentikaciju i biti prvi sloj „odbrane“ od neautorizovanog pristupa.
- **servisni sloj** (service layer) - sloj komponenti aplikacije neposredno ispod web sloja. Sadrži komponenete koje obezbeđuju servise same aplikacije i infrastrukturne servise. Aplikativni servisi obezbeđuju javno dostupni API servisnog sloja. Infrastrukturni servisi zapravo predstavljaju „povezujući kod“ koji omogućava pristup eksternim sistemima kao što su fajl sistemi, baze podataka, email serveri…
- **sloj čuvanja podataka (**repository layer) - odgovoran je za odgovarajuću kommunikaciju između aplikacije i izabranog sistema za čuvanje podataka.

MVC ARHITEKTURA APLIKACIJA

- MVC - **M**odel - **V**iew - **C**ontroller
- dizajn šablon za razvoj aplikacija - vrlo čest kod web aoplikacija, jer je to jedan način upotrebe principa razdvajanja nadležnosti. Spring aplikacije često slede ovaj šablon.
- **model** - komponente sistema koje reprezentuje podatke i ništa više - klase koje formiraju domenski model podataka, dakle nad kojima aplikacija radi. Nezavisne su od view i control komponenti
- **view** - komponente sistema koje su zadužene za vizuelizaciju podataka iz modela
- **controller -** komponente sistema koje interpretiraju korisničke akcije, kontrolišu tok podataka, dobavljaju odgovarajući model za view komponente.

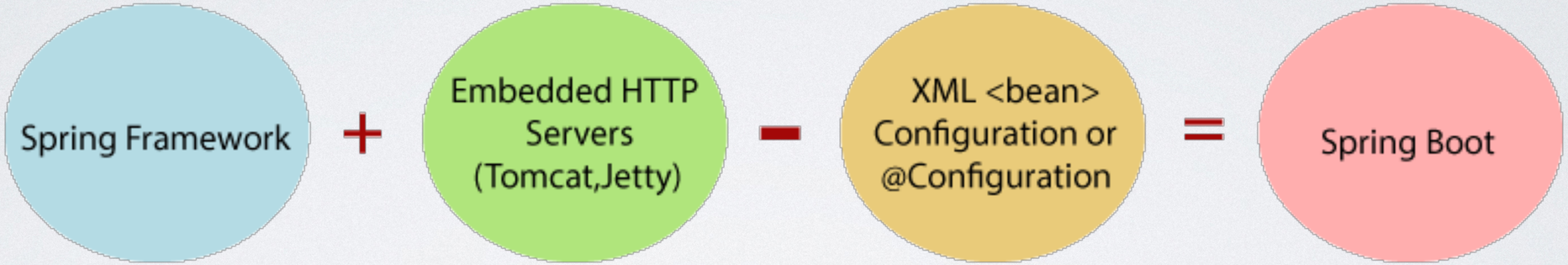
ŠTA JE SPRING BOOT?

- Spring Boot je projekat koji je izgrađen „iznad“ Spring Framework-a.
- Primarni cilj Spring Boot-a jeste da obezbedi jednostavniji i brži način za inicijalno kreiranje, kreiranje i konfigurisanje Spring projekata (koji mogu da zahtevaju jako mnogo podešavanja).
- Realizovan je kao Spring module koji obezbeđuje RAD (Rapid Application Development) za Spring Framework.
- Omogućava kreiranje samostalnih Spring baziranih aplikacija, koje mogu samostalno da se pokreću - jer zahtevaju minimalnu Spring konfiguraciju.
- Pojednostavljeno - Spring Boot je kombinacija Spring Framework, Embedded Servera i pojednostavljenog konfigurisanja projekata.

KAKO JE POJEDNOSTAVLJENA KONFIGURACIJA?

- Spring projekat zahteva relativno veliki broj konfiguracionih XML fajlova.
- Kod Spring Boot projekta XML configuration (deployment descriptor) nije neophodan.
- Koristi princip se *convention over configuration* princip.
- Moguće je pomoću Spring STS IDE-a ili Spring Initializr-a zadati neophodnu konfiguraciju za Spring Boot Java aplikacije.
- Realizovanu aplikaciju nije više potrebno exportovati isključivo u .war i *deploy*-ovati pod npr. Tomcat i konfigurisati server, već je moguće eksportovati .jar koji u sebi sadrži već ugrađen i prekonfigurisan Tomcat server.

ŠTA JE SPRING BOOT?



Slika preuzeta sa https://www.javatpoint.com/spring-boot-tutorial

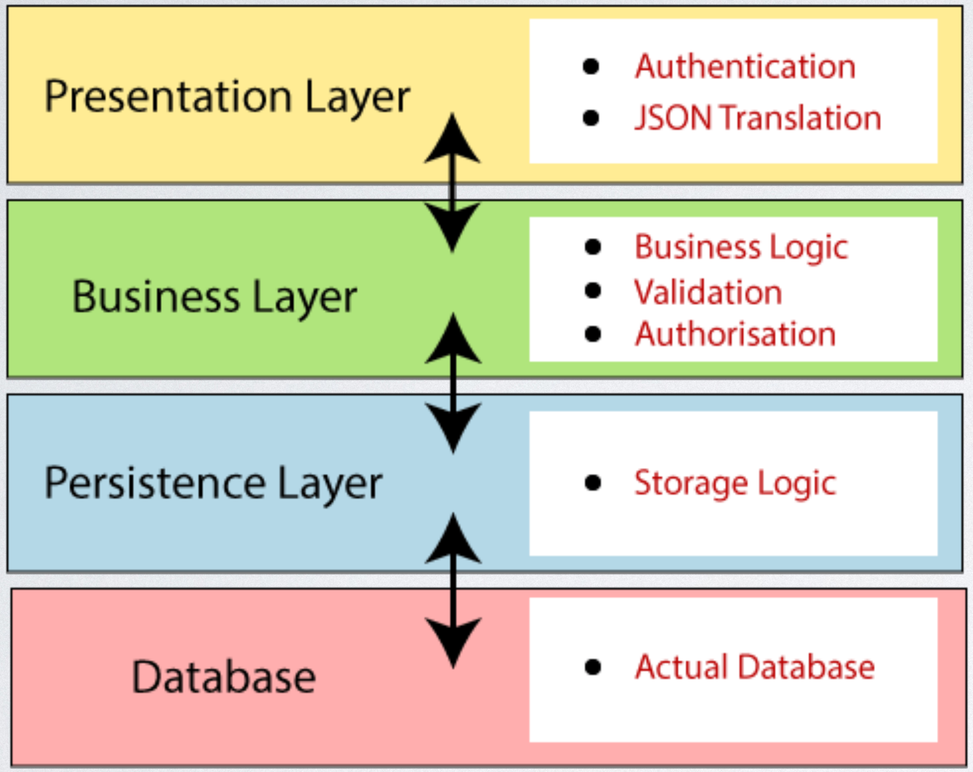
PREDNOSTI SPRING BOOT-A

- Kreira se samostalana Spring aplikacija koja se startuje sa Java -jar.
- Omogućava jednostavno testiranje web applikacija koristeći ugrađene HTTP servere kao što su Tomcat, Jetty, etc.
- Obezbeđuje 'startnu' POM konfiguraciju kako bi se pojednostavila Maven konfiguracija.
- Obezbeđuje i dodatne osobine kao što su metrika aplikacije, provera komponenti, eksternalizacija konfiguracije.
- Nema više potrebe za XML konfiguracionim fajlovima.
- Obezbeđuje veliki broj plug-ina.
- Minimizuje pisanje *boilerplate* koda (koda koji treba da se doda na dosta mesta zarad obezbeđenja nekih osnovnih funkcionlanosti, praktično istovetno bez izmena).
- Dokazano povećava produktivnosti i smanjuje vreme potrebno za razvoj.

ARHITEKTURA SPRING BOOT PROJEKATA

- Spring Boot je modul Spring Framework-a.
- Spring Boot sledi slojevitu arhitekturu u kojem svaki layer komunicira samo sa slojem ispod i iznad sebe (hijerarhijska organizacija).

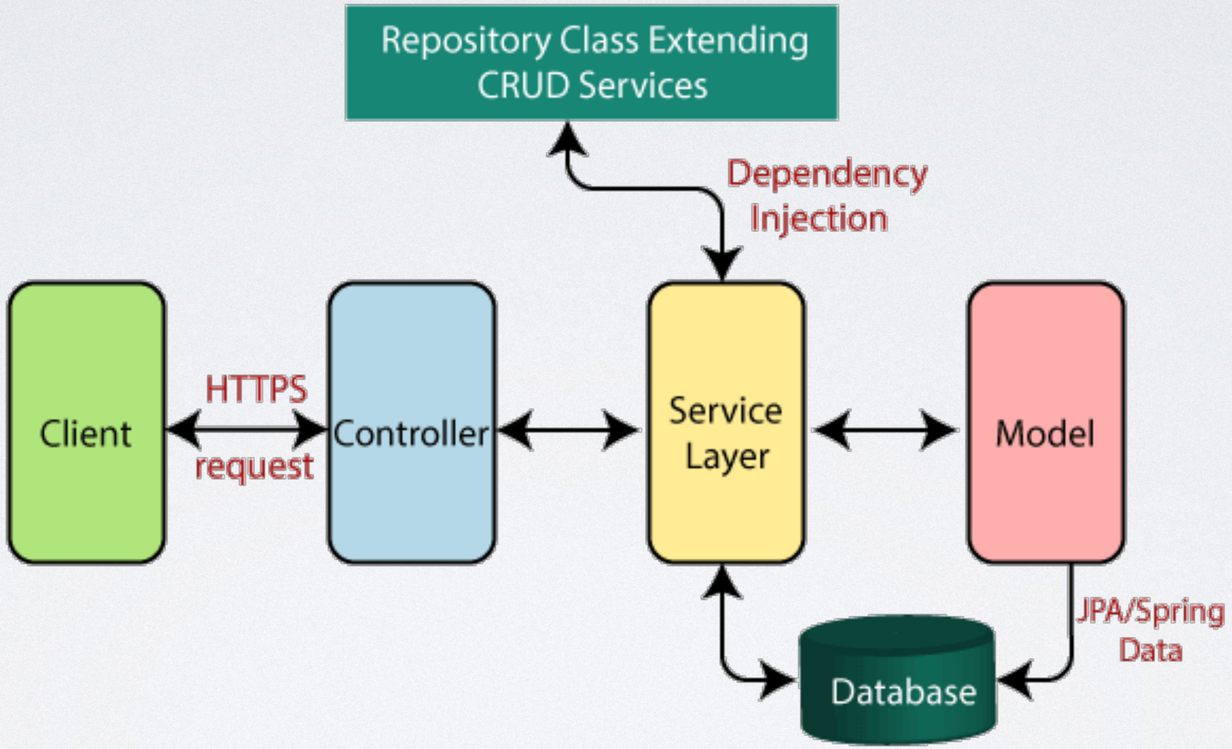
ARHITEKTURA SPRING BOOT APLIKACIJE



- **Prezentacioni sloj**: Obrađuje HTTP zahteve, transformiše JSON parametre u Java objekte, autentifikuje zahteve i prosleđuje ih odgovarajućoj komponenti u biznisl sloju. Sadrži komponente za prikaz sadržaja aplikacije - frontend deo.
- **Sloj poslovne logike** (Business Layer): Ovaj sloj odrađuje svu poslovnu logiku aplikacije. Sastoji se od **servisnih** klasa, a koristi servise koje obezbeđuje sloj za pristup podacima (data access - persistance). Takođe obavlja autorizaciju i validaciju.
- **Sloj za pristup i čuvanje podataka (Persistence Layer)**: U njemu se nalazi sva logika neophodna za čuvanje i transformaciju objektne reprezentacije podataka na reprezentaciju koju koristi baza podataka.
- **Sloj baze podataka** (Database Layer): U ovom sloju se obavalja samo čuvanje podataka kao i CRUD (create, retrieve, update, delete) operacije nad njima.

Slika preuzeta sa https://static.javatpoint.com/springboot/images/spring-boot-architecture.png

TOK PODATAKA IZMEĐU OSNOVNIH KOMPONENTI SPRING BOOT WEB APLIKACIJE



Slika preuzeta sa https://static.javatpoint.com/springboot/images/spring-boot-architecture2.png

OSNOVNE KOMPONENTE SPRING (BOOT) WEB APLIKACIJE

- **Component -** bilo koja *managed* komponenta - Spring Bean
- **Controller** - komponenta koja obavlja funkciju obrade web zahteva
- **Service** - komponenta koja obavlja određenu poslovnu logiku
- **Repository** - reprezentuje komponentu koja predstavlja DAO (Data Access Object) - preko ove komponente se obavlja sva komunikacija sa bazom podataka
- **Entity** - predstavlja komponente koje reprezentuju model podataka (entitete), koristeći Repository oni se mapiraju na bazu podataka

SPRING BOOT ANOTACIJE

- Spring Boot anotacije predstavljaju metapodatke koji obezeđuju dodatne informacije o programu i ulozi koju određeni deo koda ima u programu.
- Osnovne anotacije - Spring Framework Stereotype Annotations - anotacije koje se koriste da odrede koju ulogu u Spring frameworku određena klasa vrši

SPRING BOOT ANOTACIJE

- Anotacije (na nivou klase) za komponente
  - @Component
  - @Controller (ili @RestController)
  - @Service
  - @Repository
  - @Entity
- U Controllerima se koriste i anotacije iz Spring MVC
  - @RequestMapping - mapira URL na odgovarajuću klasu/metodu kontrolera

SPRING INITIALIZR

- Spring Initializr je veb alat koji je kreirao Pivotal Web Service.
- Koristeći ovaj alat i birajući komponente koje su nam potrebne za našu aplikaciju jednostavno se kreira osnovna struktura Spring Boot projekta.
- Omogućava da se u zavisnosti od potreba projekta izaberu i u konfiguraciju projekta automatski povežu odgovarajuće zavisnosti (dependancy).
- https://start.spring.io

Slika preuzeta sa https://static.javatpoint.com/springboot/images/spring-boot-architecture2.png