Ranga Rao Karanam

Naučite Spring 5

Sveobuhvatni vodič koji će vam pomoći da postanete stručnjak za radni okvir Spring

kompjuter biblioteka

Packt›

Ranga Rao Karanam

Naučite **Spring 5**

Izdavač:

kompjuter biblioteka

Obalskih radnika 15, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:
Mihailo J. Šolajić

Autori: Ranga Rao Karanam

Prevod: Slavica Prudkov

Lektura: Miloš Jevtović

Slog: Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: Apollo Plus D.O.O.
Beograd

Tiraž: 500

Godina izdanja: 2017.

Broj knjige: 495

Izdanje: Prvo

ISBN: 978-86-7310-5xx-x

Mastering Spring 5.0
by Ranga Rao Karanam

ISBN 978-1-78712-317-5

June 2017

O AUTORIMA

Ranga Rao Karanam je programer, trener i arhitekta. Njegova interesovanja uključuju Cloud Native Applications, mikroservise, evolucijski dizajn, visokokvalitetni kod, DevOps, BDD, TDD i preradu koda. Voli da savetuje početnike u razvoju skalabilnih Cloud Native aplikacija koje su zasnovane na komponenti i prati modernu praksu razvoja, kao, na primer,BDD, Continuous Delivery i DevOps. Voli slobodu koju radni okvir Spring pruža u razvoju poslovnih Java aplikacija.

Ranga je pokrenuo sajt *in28minutes* sa vizijom da kreira visokokvalitetne kurseve za razvoj Cloud Native Java aplikacija. Teži unapređenju svog već značajnog uspeha – na sajtu Udemy prati ga 75.000 učenika, a na YouTubeu 35.000 korisnika.

Ranga voli da igra kriket i da učestvuje na pešačkim turama. Njegov san je da provede godinu dana u pešačenju na Himalajima.

Mojoj porodici i najboljim prijateljima – hvala za sve!

O RECENZENTU

Jarosław Krochmalski je strastveni dizajner softvera i programer koji se specijalizovao u domenu finansijskog poslovanja. Ima više od 12 godina iskustva u razvoju softvera. Ljubitelj je čistog koda i razvoja softvera. On je sertifikovani ScrumMaster i veliki ljubitelj Agile. Njegova profesionalna interesovanja uključuju nove tehnologije u razvoju veb aplikacija, obrasce projektovanja, poslovnu arhitekturu i obrasce integracije. Voli da eksperimentiše sa NoSQL-om i računarstvom „u oblaku“. Jaroslaw koristi IDE od njegovog prvog izdanja i pratio je njegov rast i „sazrevanje“. Dizajnira i razvija softver profesionalno od 2000. godine i koristi Javu kao glavni programski jezik od 2002. godine. Ranije je radio za kompanije „Kredyt Bank“ (KBC) i „Bank BPS“ na velikim projektima, kao što su međunarodni novčani nalozi, ekspresna isplata i sistemi sakupljanja. Trenutno radi kao konsultant za dansku kompaniju „7N“ kao arhitekta infrastrukture za Nykredit banku. Možete da kontaktirate sa njim pomoću Twittera na @jkroch ili e-maila jarek@finsys.pl.

On je autor knjiga „*IntelliJ Idea Essentials*“, „*Developing with Docker*“ i „*Docker and Kubernetes for Java Developers*“, čiji je izdavač „Packt“, i recenzent je „Packtove“ knjige „*Spring Essentials*“.

UVOD

Spring 5.0 sadrži ogroman broj novih i uzbudljivih funkcija koje će promeniti način na koji ste do sada koristili ovaj radni okvir. Ova knjiga će vam prikazati razvoj Springa – od rešavanja problema u aplikacijama koje mogu da se testiraju, do izgradnje distribuiranih aplikacija u Cloudu.

Knjiga započinje pregledom novih funkcija u verziji Spring 5.0 i prikazom kako se gradi aplikacija pomoću Spring MVC-a. Zatim ćete naučiti kako da izgradite i proširite mikroservise pomoću radnog okvira Spring i kako da izgradite i primenite Cloud aplikacije. Videćete kako su se arhitekture aplikacije razvijale, od monolita, do onih koje su izgrađene oko mikroservisa. Napredne funkcije Spring Boota će biti ilustrovane kroz moćne primere.

Do kraja ove knjige ćete steći znanje i najbolju praksu koji su potrebni za razvoj aplikacija pomoću radnog okvira Spring.

ŠTA OBUHVATA OVA KNJIGA?

Poglavlje 1, „Evolucija radnog okvira Spring 5.0“, vodi vas kroz evoluciju radnog okvira Srping, od njegovih početnih verzija, do verzije Spring 5.0. Na početku je Spring korišćen za razvoj aplikacija koje se mogu testirati pomoću dependency injectiona i osnovnih modula. Noviji Spring projekti, kao što su Spring Boot, Spring Cloud i Spring Cloud Data Flow, posvećeni su infrastrukturi aplikacije i prebacivanju aplikacija u Cloud. Pregledaćemo različite Spring module i projekte.

U Poglavlju 2, „*Dependency injection*“, detaljno je opisan metod dependency injection – pregledaćemo njegove različite vrste dependency injectiona koje su dostupne u Springu i načine automatskog spajanja koji će nam olakšati posao. Takođe ćemo govoriti o testiranju koda.

Poglavlje 3, „*Izgradnja veb aplikacije pomoću Spring MVC-a*“, sadrži pregled izgradnje veb aplikacije pomoću Spring MVC-a.

U Poglavlju 4, „*Evolucija mikroservisa i Cloud-Native aplikacija*“, objašnjena je evolucija arhitektura aplikacije u poslednjoj deceniji. Naučićete zašto su potrebni mikroservisi i Cloud-Native aplikacije. Pregledaćemo različite Spring projekte koji pomažu izgradnju Cloud-Native aplikacija.

U Poglavlju 5, „*Izgradnja mikroservisa pomoću Spring Boota*“, opisano je kako Spring Boot uklanja složenost u kreiranju proizvodnih aplikacija zasnovanih na Springu. Ovo poglavlje olakšava da započnemo projekte zasnovane na Springu i obezbeđuje laku integraciju sa nezavisnim bibliotekama. Osim toga, povešćemo korisnike na „putovanje“ sa Spring Bootom. Započećemo poglavlje implementiranjem osnovnog veb servisa, a nastavićemo dodavanjem keširanja, obrade izuzetaka, HATEOAS-a i internacionalizacije, dok koristimo različite funkcije iz radnog okvira Spring.

Poglavlje 6, „*Proširenje mikroservisa*“, fokusirano je na dodavanje naprednijih funkcija u mikroservise koje smo izgradili u Poglavlju 4, „*Evolucija mikroservisa i Cloud-Native aplikacija*“.

U Poglavlju 7, „*Napredne funkcije Spring Boota*“, opisane su napredne funkcije u Spring Bootu. Naučićete kako da nadgledate mikroservis pomoću Spring Boot Actuatora. Zatim ćemo primeniti mikroservis na Cloud. Takođe ćete naučiti kako da efikasnije razvijate kod pomoću programerskih alatki koje obezbeđuje Spring Boot.

U Poglavlju 8, „*Spring Data*“, upoznaćete modul Spring Data. Kreiraćemo jednostavne aplikacije da bismo integrisali Spring, koristeći tehnologije JPA i Big Data.

U Poglavlju 9, „*Spring Cloud*“, opisani su distribuirani sistemi u Cloudu koji imaju zajedničke probleme, upravljanje konfiguracijom, otkrivanje servisa, prekidače i inteligenentno usmeravanje. Naučićete kako Spring Cloud pomaže da se kreiraju rešenja za ove uobičajene obrasce, koja treba dobro da funkcionišu na Cloudu i na lokalnim sistemima programera.

Poglavlje 10, „*Spring Cloud Data Flow*“, posvećeno je projektu Spring Cloud Data Flow, koji pruža kolekciju obrazaca i najbolju praksu za striming sisteme i grupno usmeravanje podataka zasnovanih na mikroservisima. Naučićete osnove Spring Cloud Data Flowa, koji ćemo upotrebiti za izgradnju osnovnih primera upotrebe toka podataka.

U Poglavlju 11, „*Reaktivno programiranje*“, istražićemo programiranje sa asinhronim tokovima podataka. U ovom poglavlju ćemo predstaviti Reactive programiranje i pregledaćemo funkcije koje obezbeđuje radni okvir Spring.

Poglavlje 12, „*Najbolja praksa za Spring*“, pomoći će vam da razumete najbolju praksu u razvoju poslovnih aplikacija korišćenjem testiranja koda, testiranja integracije, održavanja konfiguracije Springa i tako dalje.

U Poglavlju 13, „*Upotreba Kotlina u Springu*“, predstavićemo Kotlin, JVM jezik koji postaje sve popularniji. Opisaćemo kako se podešava Kotlin projekat u Eclipseu. Kreiraćemo novi Spring Boot projekat pomoću Kotlina i implementiraćemo nekoliko osnovnih servisa, koristeći testiranje koda i integracije.

ŠTA VAM JE POTREBNO ZA OVU KNJIGU?

Da biste mogli da pokrenete primere iz ove knjige, potrebni su vam sledeći alati:

- Java 8
- Eclipse IDE
- Postman

Upotrebićemo Maven koji je ugrađen u Eclipse IDE za preuzimanje svih zavisnosti koje su nam potrebne.

ZA KOGA JE OVA KNJIGA?

Ova knjiga je namenjena iskusnim Java programerima koji znaju osnove Springa i žele da nauče kako da upotrebe Spring Boot za izgradnju aplikacija i primene na Cloud.

KONVENCIJE

U ovoj knjizi pronaći ćete više različitih stilova za tekst koje sam upotrebio za različite vrste informacija. Evo nekih primera ovih stilova i objašnjenja njihovog značenja.

Reči koda u tekstu, nazivi tabela baze podataka, nazivi direktorijuma, nazivi fajlova, ekstenzije fajla, nazivi putanja, kratki URL-ovi i korisnički unos su prikazani na sledeći način: „Konfigurišite spring-bootstarter-parent u fajlu pom.xml“.

Blok koda je postavljen na sledeći način:

```
<properties>
  <mockito.version>1.10.20</mockito.version>
</properties>
```

Svaki unos komandne linije ili ispis napisan je ovako:

```
mvn clean install
```

Novi termini i **važne reči** su napisani masnim slovima. Reči koje vidite na ekranu - na primer, u menijima ili okvirima za dijalog, biće prikazane u tekstu na sledeći način: „Obezbedite detalje i kliknite na **General Project**“.

Upozorenja ili važne napomene će biti prikazani u ovakvom okviru.

Saveti i trikovi prikazani su ovako.

POVRATNE INFORMACIJE OD ČITALACA

Povratne informacije od naših čitalaca su uvek dobrodošle. Obavestite nas šta mislite o ovoj knjizi – šta vam se dopalo ili šta vam se možda nije dopalo. Povratne informacije čitalaca su nam važne da bismo kreirali naslove od kojih ćete dobiti maksimum.

Da biste nam poslali povratne informacije, jednostavno nam pošaljite e-mail na adresu informatori@kombib.rs i u naslovu poruke napišite naslov knjige.

PREUZIMANJE PRIMERA KODA

Možete da preuzmete fajlove sa primerima koda prateći sledeće korake:

1. Posetite veb stranicu knjige Naučite Spring 5 : https://goo.gl/8jviTk
2. Kliknite Preuzmite kod: https://goo.gl/Nv3osd

Kada su fajlovi preuzeti, ekstrahujte direktorijum, koristeći najnoviju verziju:

- WinRAR / 7-Zip za Windows
- Zipeg / iZip / UnRarX za Mac
- 7-Zip / PeaZip za Linux

Kod za ovu knjigu možete da pronađete i na GitHubu na adresi https://github.com/PacktPublishing/Learning-jQuery-3

ŠTAMPARSKE GREŠKE

Iako smo preduzeli sve mere da bismo obezbedili tačnost sadržaja, greške su moguće. Ako pronađete grešku u nekoj od naših knjiga (u tekstu ili u kodu), bili bismo zahvalni ako biste nam to prijavili. Na taj način možete da poštedite čitaoce od frustracije i nama da pomognete da poboljšamo naredne verzije ove knjige. Ako pronađete neku štamparsku grešku, molimo vas da nas obavestite, tako što ćete posetiti stranicu knjige: https://goo.gl/N3Ze3M i ostaviti komentar.

PIRATERIJA

Piraterija autorskog materijala na Internetu je aktuelan problem na svim medijima. Mi u „Packtu“ zaštitu autorskih prava i licenci shvatamo veoma ozbiljno. Ako pronađete ilegalnu kopiju naših knjiga, u bilo kojoj formi, na Internetu, molimo vas da nas o tome obavestite i pošaljete nam adresu lokacije ili naziv web sajta da bismo mogli da podnesemo tužbu.

Kontaktirajte sa nama na adresi copyright@packtpub.com i informatori@kombib.rs pošaljite nam link ka sumnjivom materijalu.

1

Evolucija radnog okvira Spring 5.0

Prva verzija radnog okvira Spring 1.0 izdata je u martu 2004. godine. Više od jedne i po decenije on je ostao radni okvir po izboru za izgradnju Java aplikacija. A u relativno mladom i dinamičkom svetu Java radnih okvira decenija je dug period.

Ovo poglavlje započećemo opisom osnovnih funkcija radnog okvira Spring. Pregledaćemo zašto je radni okvir Spring postao popularan, kako se prilagođavao i kako je ostao omiljen. Nakon brzog pregleda važnih modula u radnom okviru Spring, preći ćemo na svet Spring projekata. Poglavlje ćemo završiti pregledom novih funkcija u radnom okviru Spring 5.0.

Ovo poglavlje pružiće vam odgovore na sledeća pitanja:

- Zašto je radni okvir Spring popularan?
- Kako se radni okvir Spring prilagodio evoluciji arhitektura aplikacije?
- Koji su važni moduli u radnom okviru Spring?
- Gde se radni okvir Spring uklapa u Spring projektima?
- Koje su nove funkcije u radnom okviru Spring 5.0?

RADNI OKVIR SPRING

Veb sajt Springa (https://projects.spring.io/spring-framework/) definiše radni okvir Spring ovako: radni okvir Spring obezbeđuje sveobuhvatni model programiranja i konfiguracije za moderne poslovne aplikacije zasnovane na jeziku Java.

Radni okvir Spring se koristi za spajanje poslovnih Java aplikacija. Njegova glavna svrha je da vodi računa o svim tehničkim vezama koje su potrebne da bi se povezali različiti delovi aplikacije. To omogućava programerima da se fokusiraju na srž svog posla – na pisanje poslovne logike.

Problemi sa EJB-om

Radni okvir Spring je izdat u martu 2004. godine. Kada je izdata njegova prva verzija, popularan način razvijanja poslovne aplikacije je bila upotreba **Enterprise Java Beansa (EJB-a) 2.1**.

Razvijanje i primena EJB-a predstavljali su veoma komplikovan proces. Iako je EJB olakšavao distribuciju komponenata, razvijanje, testiranje koda i primena uopšte nisu bili jednostavni. Početne verzije EJB-a (1.0, 2.0, 2.1) imale su veoma složen **Application Programmer Interface (API)**, što dovodi do percepcije da uvedena složenost daleko prevazilazi prednosti:

- Teško je testiranje koda. U stvari, teško je za testiranje van EJB Containera.
- Potrebno je implementirati više interfejsa upotrebom velikog broja nepotrebnih metoda.
- Komplikovana je obrada izuzetaka.
- Opisi rasporeda nisu odgovarajući.

Radni okvir Spring je predstavio lak radni okvir namenjen olakšavanju razvoja Java EE aplikacija.

ZAŠTO JE RADNI OKVIR SPRING POPULARAN?

Prva verzija radnog okvira Spring izdata je u martu 2004. godine. U narednih 13 godina njegova upotreba i popularnost su samo rasle.

Veoma važni razlozi popularnosti radnog okvira Spring su sledeći:

- pojednostavljeno testiranje koda – zbog dependency injectiona
- smanjivanje upravljačkog koda
- arhitekturna fleksibilnost
- praćenje promena

Pojednostavljeno testiranje koda

U ranijim verzijama EJB-a testiranje koda je bilo veoma teško. U stvari, bilo je teško pokrenuti EJB van kontejnera (od verzije 2.1). Jedini način da testiramo kod bio je da ga primenimo u kontejneru.

Radni okvir Spring je uveo koncept **Dependency Injection (DI)**. Dependency injection ćemo detaljno opisati u Poglavlju 2.

Dependency injection omogućava testiranje koda, olakšavajući zamenu zavisnosti lažnim zavisnostima. Ne treba da primenimo celu aplikaciju da bismo testirali kod.

Pojednostavljenje testiranja koda ima više prednosti:

- Programeri su produktivniji.
- Greške se pronalaze ranije, pa njihovo ispravljanje nije skupo.
- Aplikacije imaju automatizovano testiranje koda, koje može da se pokrene kada se gradi **Continuous Integration**, što će ubuduće sprečiti pojavu grešaka.

Smanjenje upravljačkog koda

Pre pojave radnog okvira Spring, tipične J2EE (ili Java EE, kako se sada nazivaju) aplikacije sadržale su mnogo upravljačkog koda, kao što su, na primer, dobijanje konekcije baze podataka, kod za obradu izuzetaka, kod za upravljanje transakcijama, evidentiranje koda i tako dalje.

Pogledajte jednostavan primer izvršavanja upita pomoću unapred definisanog iskaza:

```
PreparedStatement st = null;
try {
      st = conn.prepareStatement(INSERT_TODO_QUERY);
      st.setString(1, bean.getDescription());
      st.setBoolean(2, bean.isDone());
      st.execute();
    }
                catch (SQLException e) {
      logger.error("Failed : " + INSERT_TODO_QUERY, e);
                      } finally {
                         if (st != null) {
       try {
       st.close();
      } catch (SQLException e) {
       // Ignore - nothing to do..
      }
   }
 }
```

U prethodnom primeru postoje četiri linije poslovne logike i više od 10 linija upravljačkog koda.

Koristeći radni okvir Spring, ista logika može da bude primenjena u dve linije:

```
jdbcTemplate.update(INSERT_TODO_QUERY,
bean.getDescription(), bean.isDone());
```

Kako radni okvir Spring izvršava ovu magiju?

U prethodnom primeru Spring JDBC (i Spring, generalno) konvertuje većinu potvrđenih izuzetaka u nepotvrđene izuzetke. Obično, kada je upit neuspešan, nemamo mnogo šta da uradimo, osim da zatvorimo iskaz i potvrdimo da je transakcija neuspešna. Umesto da implementiramo funkciju za obradu izuzetaka u svaki metod, možemo da imamo centralizovanu obradu izuzetaka i da je injektiramo pomoću **Spring Aspect-Oriented Programminga (AOP-a)**.

Spring JDBC uklanja potrebu za kreiranje upravljačkog koda koji je uključen u dobijanje konekcije, kreiranje unapred definisanog iskaza i tako dalje. Klasa jdbcTemplate može da bude kreirana u Spring kontekstu i injektirana u klasu **Data Access Object (DAO)** kad god je potrebna.

Slično prethodnom primeru, Spring JMS, Spring AOP i drugi Spring moduli pomažu nam u smanjenju upravljačkog koda.

Radni okvir Spring omogućava programeru da se fokusira na primarni posao programera – na pisanje poslovne logike.

Izostavljanje upravljačkog koda ima još jednu prednost – smanjenje dupliranja u kodu. Pošto je sav kod za upravljanje transakcijom, obradu izuzetaka i tako dalje implementiran na jednom mestu, mnogo ga je lakše održavati.

Arhitekturna fleksibilnost

Radni okvir Spring je modularan. Izgrađen je kao skup nezavisnih modula koji su izgrađeni povrh osnovnih Spring modula. Većina Spring modula je nezavisna – možete da upotrebite jedan modul bez potrebe da upotrebite druge.

Pogledajte nekoliko primera:

- - U veb sloju Spring pruža sopstveni radni okvir - Spring MVC. Međutim, Spring ima odličnu podršku za Struts, Vaadin, JSF ili bilo koji drugi veb radni okvir.
- - Spring Beans može da obezbedi jednostavnu implementaciju poslovne logike. Međutim, Spring može da bude integrisan i pomoću EJB-a.
- - U sloju podataka Spring pojednostavljuje JDBC pomoću svog Spring JDBC modula. Međutim, Spring ima odličnu podršku za bilo koji drugi radni okvir sloja podataka - JPA, Hibernate (sa ili bez JPA-a) ili iBatis.
- - Imate na raspolaganju opciju implementiranja unakrsnih pitanja (evidentiranje, upravljanje transakcijom, bezbednost i tako dalje) pomoću Spring AOP-a ili možete da integrišete punopravne AOP implementacije, kao što je AspectJ.

Radni okvir Spring „ne želi“ da bude „sveznalica“. Iako se fokusira na svoj osnovni posao smanjenja povezanosti između različitih delova aplikacije i omogućava njihovo testiranje, Spring obezbeđuje odličnu integraciju sa radnim okvirima po vašem izboru. To znači da imamo fleksibilnost u arhitekturi – ako ne želimo da upotrebimo specifičan radni okvir, možemo da ga zamenimo drugim.

Budite „u toku“ sa promenama

Prva verzija radnog okvira Spring bila je fokusirana na omogućavanje testiranja aplikacije. Međutim, vremenom, nastali su i drugi izazovi. Radni okvir Spring je napredovao i uspeo da ostane popularan zbog svoje fleksibilnosti i modula koje pruža. Sledi lista nekih prednosti Springa:

- Anotacije su predstavljene u radnom okviru Java 5. Spring (verzija 2.5 – u novembru 2007. godine) i bile su ispred Java EE-a u predstavljanju modela kontrolera zasnovanog na anotacijama za Spring MVC. Programeri koji su koristili Java EE morali su da čekaju do verzije Java EE 6 (koja je izdata u decembru 2009. godine) pre nego što su dobili sličnu funkciju.
- Radni okvir Spring predstavio je veliki broj apstrakcija pre Java EE-a da bi aplikacije održao razdvojene od specifične implementacije. API keširanja obezbeđuje takav slučaj. Java EE je izdao JSR-107 za JCache (2014) – podrška za ovu verziju je obezbeđena u verziji Spring 4.1.

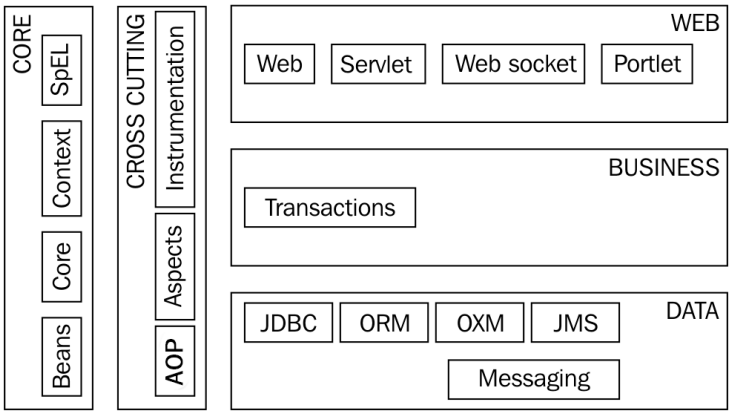
Još jedna važna stvar koju Spring uvodi je paleta Spring Projects. Radni okvir Spring je samo jedan od mnogih projekata pod Spring Projectsom. Mi ćemo opisati različite Spring projekte u posebnom odeljku. U sledećim primerima ilustrovano je kako Spring uspeva da ostane popularan uz nove Spring projekte.

- **Spring Batch** definiše novi pristup za izgradnju Java Batch aplikacija. Morali smo da čekamo do verzije Java EE 7 (u junu 2013. godine) da bismo imali sličnu specifikaciju aplikacije za paketnu obradu u Java EE-u.
- Kako se arhitektura razvijala u pravcu Clouda i mikroservisa, Spring je uveo i nove Spring projekte koji su orijentisani ka Cloudu. Spring Cloud nam pomaže u pojednostavljenju razvoja i primene mikroservisa.

SPRING MODULI

Modularnost radnog okvira Spring je jedan od najvažnijih razloga za njegovu popularnost. Radni okvir Spring je visoko modularan, sa više od 20 različitih modula koji imaju jasno definisane granice.

Na sledećoj slici prikazani su različiti Spring moduli organizovani po sloju aplikacije za koju se obično koriste:



Započećemo odeljak opisom Spring Core Containera, pre nego što pređemo na druge module koji su grupisani prema sloju aplikacije u kojem se obično koriste.

Spring Core Container

Spring Core Container obezbeđuje osnovne funkcije radnog okvira Spring – dependency injection, **IoC (Inversion of Control)** kontejner i kontekst aplikacije. Naučićemo više o DI-u i IoC Containeru u Poglavlju 2, „*Dependency injection*“.

Važni osnovni Spring moduli su izlistani u sledećoj tabeli:

MODUL/ARTEFAKT	UPOTREBA
spring-core	pomoćni programi koje koriste drugi Spring moduli
spring-beans	podrška za Spring Beans; u kombinaciji sa spring-coreom, obezbeđuje osnovnu funkciju radnog okvira Spring - dependency injection. Uključuje implementaciju za BeanFactory.
spring-context	Implementira ApplicationContext, koji proširuje BeanFactory i obezbeđuje podršku za učitavanje resursa i internacionalizacije, između ostalog.
spring-expression	Proširuje **EL (Expression Language iz JSP-a)** i obezbeđuje jezik za pristup i manipulaciju svojstva bean (uključujući nizove i kolekcije).

Unakrsne odgovornosti

Unakrsne odgovornosti su primenljive na sve slojeve aplikacije – na primer, na evidentiranje i bezbednost. AOP se, obično, koristi za implementiranje unakrsnih odgovornosti.

Testovi koda i testovi integracije spadaju u ovu kategoriju, jer su primenljivi na sve slojeve.

Važni Spring moduli koji se odnose na unakrsne odgovornosti su izlistani u narednoj tabeli:

MODUL/ARTEFAKT	UPOTREBA
spring-aop	Obezbeđuje osnovnu podršku za programiranje koje je vođeno aspektom – upotrebom presretača i pointcuta metoda.
spring-aspects	Obezbeđuje integraciju sa najpopularnijim i potpuno opremljenim AOP radnim okvirom AspectJ.
spring-instrument	Obezbeđuje osnovnu podršku za instrumentaciju.
spring-test	Obezbeđuje osnovnu podršku za testiranje koda i testiranje integracije.

Veb

Spring obezbeđuje sopstveni MVC radni okvir Spring MVC i odličnu integraciju sa popularnim veb radnim okvirima, kao što je Struts.

Važni artefakti/moduli su sledeći:

- **spring-web -** Obezbeđuje osnovne veb funkcije, kao što je slanje fajla koji se sastoji od više delova. Obezbeđuje podršku za integraciju sa drugim veb radnim okvirima, kao što je Struts.
- **spring-webmvc -** Obezbeđuje potpuno opremljeni veb MVC radni okvir Spring MVC, koji uključuje funkcije za implementiranje REST servisa.

Mi ćemo opisati Spring MVC i pomoću njega kreirati veb aplikacije i rest servise u Poglavlju 3, „*Izgradnja veb aplikacije pomoću Spring MVC-a*“, i u Poglavlju 5, „*Izgradnja mikroservisa pomoću Spring Boota*“.

Posao

Poslovni sloj je fokusiran na izvršavanje poslovne logike aplikacije. Pomoću Springa poslovna logika je obično implementirana u Plain Old Java Objectu (POJO-u).

Spring Transactions (spring-tx) obezbeđuje deklarativno upravljanje transakcijom za POJO i druge klase.

Podaci

Sloj podataka u aplikacijama obično komunicira sa bazom podataka i/ili eksternim interfejsima. Neki od važnih Spring modula koji se odnose na sloj podataka su izlistani u sledećoj tabeli:

MODUL/ARTEFAKT	UPOTREBA
spring-jdbc	Obezbeđuje apstrakciju oko JDBC-a za izbegavanje šablonskog koda.
spring-orm	Obezbeđuje integraciju sa ORM radnim okvirima i specifikacijama - na primer, sa radnim okvirima JPA i Hibernate.
spring-oxm	Obezbeđuje objekat za integraciju XML mapiranja. Podržava radne okvire, kao što su JAXB, Castor i tako dalje.
spring-jms	Obezbeđuje apstrakciju oko JMS-a za izbegavanje šablonskog koda.

SPRING PROJEKTI

Iako radni okvir Spring obezbeđuje osnovu za osnovne funkcije poslovnih aplikacija (DI, Veb, podatke), ostali Spring projekti istražuju integraciju i rešenja za druge probleme u poslovnom prostoru – kao što su primena i projekti Cloud, Big Data, Batch i Security.

Neki važni Spring projekti su sledeći:

- Spring Boot
- Spring Cloud
- Spring Data
- Spring Batch
- Spring Security
- Spring HATEOAS

Spring Boot

Neki od izazova prilikom izgradnje mikroservisa i veb aplikacija su sledeći:

- donošenje odluka o izboru radnog okvira i verzijama kompatibilnog radnog okvira
- obezbeđivanje mehanizma za eksternalizaciju konfiguracije – svojstava koji mogu da se menjaju od jednog okruženja do drugog
- provere i praćenje – obezbeđivanje upozorenja ako su specifični delovi aplikacije pali
- odlučivanje o okruženju primene mikroservisa i konfigurisanje aplikacije za to okruženje

Spring Boot rešava sve ove probleme, uzimajući u obzir mišljenje kako aplikacije treba da budu razvijene.

Spring Boot ćemo detaljno opisati u Poglavlju 5, „*Izgradnja mikroservisa pomoću Spring Boota*“, i u Poglavlju 7, „*Napredne funkcije Spring Boota*“.

Spring Cloud

Nije preterano ako kažemo da se svet pomera u Cloud (oblak).

Cloud Native mikroservisi i aplikacije su veoma popularni. Opisaćemo ih detaljno u Poglavlju 4, „*Evolucija mikroservisa i Cloud-Native aplikacija*“.

Spring brzo napreduje ka olakšavanju razvoja aplikacija za Cloud pomoću projekta Spring Cloud.

Spring Cloud obezbeđuje rešenja za uobičajene šablone u distribuiranim sistemima. On omogućava programerima da brzo kreiraju aplikacije koje implementiraju uobičajene šablone. Neki od uobičajenih šablona koji su implementirani u Spring Cloudu su sledeći:

- upravljanje konfiguracijom
- otkrivanje servisa
- prekidači
- inteligentno usmeravanje

Opisaćemo Spring Cloud i njegov različit raspon funkcija detaljnije u Poglavlju 9, „Spring Cloud“.

Spring Data

Postoji više izvora podataka u današnjem svetu - SQL (relacioni) i različite NOSQL baze podataka. Spring Data pokušava da obezbedi dosledan pristup podacima za sve vrste baza podataka.

Spring Data obezbeđuje integraciju sa različitim specifikacijama i/ili skladištima podataka:

- JPA
- MongoDB
- Redis
- Solr
- Gemfire
- Apache Cassandra

Neke od važnih funkcija Data projekta su sledeće:

- obezbeđena apstrakcija oko skladišta i mapiranja objekta određivanjem upita iz naziva metoda
- jednostavna Spring integracija
- integracija sa Spring MVC kontrolerima
- napredne automatske funkcije za nadgledanje - created by, created date, last changed by i last changed date

Spring Data ćemo opisati detaljnije u Poglavlju 8, „*Spring Data*“.

Spring Batch

Poslovne aplikacije danas obrađuju mnogo podataka, koristeći programe za paketnu obradu. Potrebe ovih aplikacija za njihovo funkcionisanje su veoma slične. Spring Batch obezbeđuje rešenja za velike programe za paketnu obradu sa zahtevima visoke performanse.

Važne funkcije u Spring Batchu su sledeće:

- mogućnost pokretanja, zaustavljanja i restartovanja poslova, uključujući i mogućnost restartovanja neuspešnih poslova od tačke gde se greška javila
- mogućnost obrade podataka u grupi
- mogućnost ponovnog izvršavanja koraka ili izostavljanja koraka prilikom pojave greške
- interfejs za administraciju, zasnovan na Vebu

Spring Security

Autentifikacija je proces identifikacije korisnika. **Autorizacija** je proces potvrđivanja da korisnik ima pristup za izvršavanje identifikovane akcije u resursu.

Autentifikacija i autorizacija su važni delovi poslovnih aplikacija i veb aplikacija i veb servisa. Spring Security obezbeđuje deklarativnu autentifikaciju i autorizaciju za aplikacije zasnovane na Javi.

Važne funkcije u Spring Securityu su sledeće:

- - autentifikacija i autorizacija pojednostavljene
- - odlična integracija sa Spring MVC-om i Servlet API-jima
- - podrška za sprečavanje uobičajenih bezbednosnih napada - **cross-site forgery request (CSRF) i Session Fixation**
- - moduli dostupni za integraciju sa SAML-om i LDAP-om

Opisaćemo kako se obezbeđuju veb aplikacije pomoću Spring Securitya u Poglavlju 3, „*Izgradnja veb aplikacija pomoću Spring MVC-a*“.

Opisaćemo u Poglavlju 6, „*Proširenje mikroservisa*“, kako možete da obezbedite REST Services pomoću mehanizama autentifikacije Basic i Oauth, koristeći Spring Security.

Spring HATEOAS

HATEOAS je skraćenica za **Hypermedia as The Engine of Application State**. Iako mu naziv zvuči komplikovano, koncept je prilično jednostavan. Glavni cilj je da se razdvoji server (provajder servisa) od klijenta (korisnika servisa).

Provajder servisa obezbeđuje korisniku servisa informacije koje akcije mogu da se izvrše u resursu.

Spring HATEOAS obezbeđuje HATEOAS implementaciju – posebno za REST servise, koji su implementirani pomoću Spring MVC-a.

Važne funkcije Spring HATEOAS-a su sledeće:

- pojednostavljena definicija linkova koji pokazuju ka metodima servisa, čineći linkove manje krhkim
- podrška za JAXB (zasnovan na XML-u) i JSON integraciju
- podrška za korisnika servisa (na strani klijenta)

Opisaćemo detaljno u Poglavlju 6, „**Proširenje mikroservisa**“, kako se upotrebljava HATEOAS.

NOVE FUNKCIJE U RADNOM OKVIRU SPRING 5.0

Radni okvir Spring 5.0 je prva glavna nadgradnja radnog okvira Spring, koja se pojavila skoro četiri godine nakon verzije Spring 4.0. U toku te četiri godine glavni napredak bila je evolucija Spring Boot projekta. Opisaćemo nove funkcije u Spring Bootu 2.0 u sledećem odeljku.

Jedna od najvažnijih funkcija radnog okvira Spring 5.0 je **reaktivno programiranje**. Osnovne funkcije reaktivnog programiranja i podrška reaktivnih krajnjih tačaka dostupne su u radnom okviru Spring 5.0. Lista važnih promena uključuje sledeće:

- osnovne nadgradnje
- kompatibilnost pokretanja JDK-a 9
- upotreba JDK 8 funkcija u kodu radnog okvira Spring
- podrška za reaktivno programiranje
- funkcionalni veb radni okvir
- Java modularnost sa Jigsawom
- podrška za Kotlin
- otpuštene funkcije

Osnovne nadgradnje

Radni okvir Spring 5.0 ima JDK 8 i Java EE 7 osnovu. U suštini, to znači da prethodne verzije JDK-a i Java EE-a više nisu podržane.

Neke važne osnove Java EE 7 specifikacija za radni okvir Spring 5.0 su sledeće:

- Servlet 3.1
- JMS 2.0
- JPA 2.1
- JAX-RS 2.0
- Bean Validation 1.1

Postoji mnogo promena u minimalno podržanim verzijama nekoliko Java radnih okvira. Sledeća lista sadrži neke minimalno podržane verzije istaknutih radnih okvira:

- Hibernate 5
- Jackson 2.6
- EhCache 2.10
- JUnit 5
- Tiles 3

Sledeća lista sadrži podržane verzije servera:

- Tomcat 8.5+
- Jetty 9.4+
- WildFly 10+
- Netty 4.1+ (za veb reaktivno programiranje pomoću Spring Web Fluxa)
- Undertow 1.4+ (za veb reaktivno programiranje pomoću Spring Web Fluxa)

Aplikacije koje koriste ranije verzije bilo kojih navedenih specifikacija/radnih okvira treba da budu nadgrađene najmanje na prethodno izlistane verzije pre nego što mogu da upotrebe radni okvir Spring 5.0.

Kompatibilnost pokretanja JDK-a 9

JDK 9 je izdat sredinom ove godine. Očekuje se da će radni okvir Spring 5.0 imati kompatibilnost pokretanja JDK-a 9.

Upotreba JDK 8 funkcija u kodu radnog okvira Spring

Verzija osnove radnog okvira Spring 4.x je Java SE 6. To znači da Spring podržava Java 6, 7 i 8. Potreba za podrškom za Java SE 6 i 7 postavlja neka ograničenja u kod radnog okvira Spring. Kod radnog okvira ne može da upotrebi ni jednu od novih funkcija u verziji Java 8. Dakle, iako je ostatak sveta nadgradio verziju na Java 8, kod u radnom okviru Spring (bar glavni delovi) bio je ograničen na upotrebu ranijih verzija Jave.

U radnom okviru Spring 5.0 verzija osnove je Java 8. Kod radnog okvira Spring je sada nadgrađen da koristi nove funkcije u Javi 8, što će rezultirati čitkijim i bržim kodom radnog okvira. Neke od funkcija Jave 8 koje su upotrebljene su sledeće:

- standardni metodi Java 8 u osnovnim interfejsima Springa
- poboljšanja internog koda na osnovu poboljšanja Java 8 refleksije
- upotreba funkcionalnog programiranja u kodu radnog okvira – lambda i tokovi

Podrška za reaktivno programiranje

Reaktivno programiranje je jedna od najvažnijih funkcija radnog okvira Spring 5.0.

Arhitekture mikroservisa su, obično, izgrađene na osnovu komunikacije zasnovane na događajima. Aplikacije su građene tako da reaguju na događaje (ili poruke).

Reaktivno programiranje obezbeđuje alternativni stil programiranja fokusiranog na izgradnju aplikacija koje reaguju na događaje.

Iako Java 8 nema ugrađenu podršku za reaktivno programiranje, postoji veliki broj radnih okvira koji obezbeđuju podršku za to programiranje:

- **Reactive Streams** - jezički neutralan pokušaj da se definišu reaktivni API-ji
- **Reactor** - Java implementacija Reactive Streamsa, koju obezbeđuje Spring Pivotal tim
- **Spring WebFlux** - Omogućava razvoj veb aplikacija na osnovu reaktivnog programiranja. Obezbeđuje programski model sličan Spring MVC-u.

Opisaćemo reaktivno programiranje i način na koji možete da ga implementirate pomoću Spring Web Fluxa u Poglavlju 11, „*Reaktivno programiranje*“.

Funkcionalni veb radni okvir

Oslanjajući se na reaktivne funkcije, Spring 5 takođe obezbeđuje funkcionalni veb radni okvir.

Funkcionalni veb radni okvir obezbeđuje funkcije za definisanje krajnjih tačaka pomoću funkcionalnosti stila programiranja. Jednostavan „hello world“ primer je prikazan ovde:

```
RouterFunction<String> route =
route(GET("/hello-world"),
request -> Response.ok().body(fromObject("Hello World")));
```

Funkcionalni veb radni okvir takođe može da se upotrebi za definisanje složenijih ruta, kao što je prikazano u sledećem primeru:

```
RouterFunction<?> route = route(GET("/todos/{id}"),
request -> {
   Mono<Todo> todo = Mono.justOrEmpty(request.
   pathVariable("id"))
   .map(Integer::valueOf)
   .then(repository::getTodo);
   return Response.ok().body(fromPublisher(todo, Todo.
   class));
  })
 .and(route(GET("/todos"),
 request -> {
   Flux<Todo> people = repository.allTodos();
   return Response.ok().body(fromPublisher(people, Todo.
   class));
 }))
.and(route(POST("/todos"),
request -> {
  Mono<Todo> todo = request.body(toMono(Todo.class));
  return Response.ok().build(repository.saveTodo(todo));
}));
```

Važno je da napomenemo:

- RouterFunction procenjuje odgovarajući uslov za zahteve rutiranja za odgovarajuću funkciju za obradu.
- Definisali smo tri krajnje tačke, dve GET i jednu POST, i mapirali smo ih u različite funkcije za obradu.

Mono i Flux ćemo opisati detaljnije u Poglavlju 11, „Reaktivno programiranje“.

Java modularnost sa Jigsawom

Do verzije Java 8, Java platforma nije bila modularna. Iz toga je proizašlo nekoliko problema:

- **prenatrpanost platforme** - Java modularnost nije bila razlog za brigu u poslednje dve decenije. Međutim, hitno je potrebno upotrebom Internet of Thingsa (IOT) i novih lakih platformi, kao što je Node.js, rešiti problem prenatrpanosti Java platforme (početna verzija JDK-a je bila manja od 10 MB, a najnovije verzije zahtevaju više od 200 MB).

- **JAR Hell** - Veliki problem je i JAR Hell. Kada Java ClassLoader pronađe klasu, neće videti da li za nju postoje druge dostupne definicije. Odmah učitava prvu klasu koju pronađe. Ako dva različita dela aplikacije zahtevaju istu klasu iz različitih arhiva, ne postoji način da specifikuju arhivu iz koje treba da bude učitana klasa.

Open System Gateway initiative (OSGi) je jedna od inicijativa za uvođenje modularnosti u Java aplikacije, koja je započeta 1999. godine.

Svaki modul (koji se zove komplet) definiše sledeće:

- **uvoz** - drugi kompleti koje modul koristi
- **izvoz** - paketi koje ovaj paket izvozi

Svaki modul može da ima svoj „životni ciklus“. Može da bude instaliran, pokrenut i zaustavljen nezavisno.

Jigsaw je inicijativa pod projektom **Java Community Process (JCP)**, pokrenuta od verzije Java 7, za uvođenje modularnosti u Javu. Jigsaw ima dva glavna cilja:

- definisanje i implementiranje modularne strukture za JDK
- definisanje sistema modula za aplikacije koje su izgrađene na Java platformi

Očekuje se da će Jigsaw biti deo verzije Java 9 i da će radni okvir Spring 5.0 uključiti osnovnu podršku za Jigsaw module.

Kotlin podrška

Kotlin je statični tip JVM jezika. On omogućava kod koji je ekspresivan, kratak i čitak. Radni okvir Spring 5.0 ima odličnu podršku za Kotlin.

Pogledajte jednostavan Kotlin program koji ilustruje klasu podataka:

```
import java.util.*
data class Todo(var description: String, var name: String,
var
targetDate : Date)
fun main(args: Array<String>) {
  var todo = Todo("Learn Spring Boot", "Jack", Date())
  println(todo)
        //Todo(description=Learn Spring Boot, name=Jack,
    //targetDate=Mon May 22 04:26:22 UTC 2017)
  var todo2 = todo.copy(name = "Jill")
  println(todo2)
     //Todo(description=Learn Spring Boot, name=Jill,
     //targetDate=Mon May 22 04:26:22 UTC 2017)
```

```
    var todo3 = todo.copy()
    println(todo3.equals(todo)) //true
}
```

U manje od 10 linija koda smo kreirali i testirali podatke koji imaju tri svojstva i sledeće funkcije:

- `equals()`
- `hashCode()`
- `toString()`
- `copy()`

Kotlin je strogo tipiziran. Međutim, nema potrebe da eksplicitno specifikujemo tip svake promenljive:

```
val arrayList = arrayListOf("Item1", "Item2", "Item3")
// Type is ArrayList
```

Imenovani argumenti omogućavaju da specifikujemo nazive argumenata kada pozivamo metode, što rezultira još čitkijim kodom:

```
var todo = Todo(description = "Learn Spring Boot",
name = "Jack", targetDate = Date())
```

Kotlin pojednostavljuje funkcionalno programiranje obezbeđivanjem standardnih promenljivih (`it`) i metoda, kao što su `take`, `drop` i tako dalje:

```
var first3TodosOfJack = students.filter { it.name == "Jack"
 }.take(3)
```

Takođe možemo da specifikujemo standardne vrednosti za argumente u Kotlinu:

```
import java.util.*
data class Todo(var description: String, var name: String, var
targetDate : Date = Date())
fun main(args: Array<String>) {
  var todo = Todo(description = "Learn Spring Boot", name =
  "Jack")
}
```

Sa svim ovim funkcijama koje kod čine konciznim i ekspresivnim, očekujemo da će Kotlin biti jezik koji će se učiti

Više reči o Kotlinu biće u Poglavlju 13, „*Upotreba Kotlina u Springu*“.

Uklonjene funkcije

Radni okvir Spring 5.0 je glavno izdanje Springa sa značajnim povećanjem u osnovi. Osim što je povećan u verzijama Java, JavaEE i nekoliko drugih radnih okvira, radni okvir Spring 5.0 je takođe uklonio podršku za nekoliko radnih okvira:

- Portlet
- Velocity
- JasperReports
- XMLBeans
- JDO
- Guava

Ako koristite bilo koji od navedenih radnih okvira, preporučujemo da planirate promenu i ostanete na verziji radnog okvira Spring 4.3, koja je podržana do 2019. godine.

NOVE FUNKCIJE SPRING BOOTA 2.0

Prva verzija Spring Boota je izdata 2014. godine. Sledi lista nekih važnih ažuriranja koja se očekuju u verziji Spring Boot 2.0:

- Osnovna JDK verzija je Java 8.
- Osnovna Spring verzija je Spring 5.0.
- Spring Boot 2.0 ima podršku za reaktivno veb programiranje pomoću WebFluxa.

Minimalna podržana verzija nekih važnih radnih okvira je izlistana ovde:

- Jetty 9.4
- Tomcat 8.5
- Hibernate 5.2
- Gradle 3.4

Detaljno ćemo opisati Spring Boot u Poglavlju 5, „*Izgradnja mikroservisa pomoću Spring Boota*“, i u Poglavlju 7, „*Napredne funkcije Spring Boota*“.

REZIME

Tokom poslednje decenije i po radni okvir Spring je dramatično poboljšao iskustvo programerima za razvijanje poslovnih Java aplikacija. Pomoću radnog okvira Spring 5.0 uvedeno je mnoštvo novih funkcija, dok se osnova značajno smanjila.

U sledećim poglavljima ćemo opisati dependency injection i način kreiranja veb aplikacija pomoću Spring MVC-a. Nakon toga ćemo preći u svet mikroservisa. U Poglavlju 5, „*Izgradnja mikroservisa pomoću Spring Boota*“, Poglavlju 6, „*Proširenje mikroservisa*“, i Poglavlju 7, „*Napredne funkcije Spring Boota*“, opisaćemo kako Spring Boot pojednostavljuje kreiranje mikroservisa. Zatim ćemo usmeriti pažnju na izgradnju aplikacija u Cloudu pomoću modula Spring Cloud i Spring Cloud Data Flow.