Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.41:[004.771:339
DOI: https://doi.org/10.24867/02BE00Gere

RAZVOJ WEB APLIKACIJA SA SPRING DATA REST NA PRIMERU APLIKACIJE ZA PRAĆENJE ZADATAKA

DEVELOPMENT OF WEB APPLICATIONS WITH SPRING DATA REST WITH AN EXAMPLE OF ISSUE TRACKER APPLICATION

Igor Gere, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu prikazan je razvoj demonstrativne web aplikacije korišćenjem Spring Data REST modula popularnog framework-a Spring. Ukazane su neke prednosti i mane korišćenja modula koji je u fokusu ovog rada.*

Ključne reči: *Java, Spring, JPA, REST*

Abstract – *This paper presents development of web application using Spring Data REST module of the popular framework Spring. Some advantages and drawbacks of module which is in the focus of this paper are shown.*

Keywords: *Java, Spring, JPA, REST*

1. UVOD

Tema ovog rada jeste modul Spring Data REST, jedan od modula Spring framework-a (radni okvir). Pomenuti modul će biti iskorišćen prilikom implementacije demonstrativnog projekta. Iskustva stečena prilikom implementacije će biti navedena u okviru ovog rada.

Funkcionalnosti koje su implementirane u demonstrativnoj aplikaciji korišćenjem Spring framework-a su mogle biti implementirane i na druge načine bez korišćenja Spring Data REST modula, ali bi to zahtevalo daleko složeniji kod. Ovaj rad služi i kao primer kako posmatrani modul ubrzava razvoj aplikacija.

Pored Spring Data REST koji predstavlja glavnu temu rada korišćen je i modul Spring Data JPA. Cilj je da se istraži na koji način se koristi Spring Data REST kroz implementaciju aplikacije koja predstavlja issue tracker (sistem za evidenciju zadataka).

Svrha primer aplikacije jeste implementacija funkcionalnosti korišćenjem Spring Data REST modula radi njegove analize i demonstracije, dok sam Spring famework kao takav nije u fokusu.

1.1. REST

REST (Representational State Transfer) prestavlja arhitektonski stil mrežno baziranih aplikacija, definisao ga je Roy Thomas Fielding u svojoj doktorskoj disertaciji [1].

Predstavlja set pravila koja se primenjuju za izgradnju web baziranih aplikacija. Nastao je na osnovu izbora pogodnih arhitektonskih stilova uzimajući u obzir njihove pojedinačne prednosti i mane, a sa ciljem kreiranja stila koji je optimalan u opštem slučaju.

Od postavljenih pravila REST arhitekturom naglasio bih da je to klijent-server arhitektura kod koje je postavljen zahtev za unificiranim interfejsom. Unificirani interfejs olakšava izgradnju složenog sistema olakšavajući interakcije izeđu njegovih komponenti. Kao negativnu osobinu unificirani interfejs smanjuje efikasnost sistema jer podaci nisu prilagodjeni konkretnom slučaju korišćenja.

Svaku informaciju REST smatra resursom. Resursi ne moraju direktno predstavljati entitete u sistemu već njihovo značenje. Moguće je da isti entitet bude predstavljen kao dva različita resursa ili da više entiteta bude spregnuto u jedan resurs. Interakcija izmedju komponenti sistema odvija se razmenom resursa koji u toj interakciji mogu biti predstavljeni svojim identifikatorom. U ovom slučaju identifikatorom se smatra URL (uniform resource locator), a ne primarni ključ entitata koji je predstavljen resursom.

REST je stateless što znači da svaki zahtev ka REST sistemu treba da sadrži sve potrebne informacije za njegovu obradu i da se ne može računati na neko sačuvano stanje na serveru tzv. sesiju.

HATEOAS (Hypermedia as the Engine of Application State) je princip REST-a po kome je potrebno obezbediti način za otkrivanje i navigaciju interfejsom. Jednostavno objašnjenje se može naći na [2].

1.2. Spring i Spring Boot

Spring framework je open source (otvorenog koda) framework nastao 2002. godine i veoma je popularan u Java svetu. Namena mu je kreiranje kompleksnih aplikacija gde je jaka konkurencija EJB-u (Enterprise JavaBeans) [3]. Tokom godina evoluirao je uporedo sa programskim jezikom Java i trenutno se nalazi u verziji 5 koja kao minimalnu verziju Jave zahteva verziju 8. Sastavljen je od mnoštva modula koje pokrivaju celokupni spektar potreba kompleksnih aplikacija.

Osnovni koncepti zastupljeni u Spring-u (Spring framework) su dependency injection / inversion of control (inverzija kontrole) i convention over configuration (konvencija pre konfiguracije). Inverzija kontrole je postignuta time što Spring aplikacija predstavlja kontejner za komponente koje mogu biti zavisne jedne od drugih. Svaka komponenta ume da kreira sebe, a komponente od kojih zavisi će joj biti pružene automatski od strane Spring kontejnera. Komponente se u terminologiji Springa nazivaju bean-ovi i one mogu biti različitog tipa.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Branko Milosavljević, red. prof.

One dolaze konfigurisane po konvenciji, a njihovom korisniku je ostavljena mogućnost da konfiguraciju izmeni ako je potrebno. Na navedenim principima zasnovani su moduli od kojih se framework sastoji.
Ranije navedene glavne koncepte Spring framework-a pogotovu zapažamo pri radu sa Spring Boot modulom koji je korišćen kao osnova za razvoj primer aplikacije u okviru ovog rada. Spring Boot je osmišljen tako da ubrza razvoj aplikacija nudeći default (uobičajnu/podrazumevanu) konfiguraciju modula koji se uključuju u sastav razvijane aplikacije.
Takođe Spring Boot vodi računa o verzijama modula tako da je početak razvoja dodatno olakšan. Korišćenjem Spring Boot modula vrlo brzo se moze preći na implementaciju biznis logike jer je dodatna konfiguracija od one sa kojom Spring Boot već dolazi često minimalna.
Ovo je veoma olakšalo razvoj i dodatno podiglo popularnost Spring framework-a.
Spring framework je veoma fleksibilan tako da kada je potrebno uobičajna konfiguracija se lako može izmeniti i prilagoditi potrebama aplikacije.

1.3. Spring Data JPA

Spring Data predstavlja skup modula Spring framework-a zaduženih za rad sa skladištima podataka. Nudi veći broj podmodula orjentisanih ka konkretnom tipu skladišta podataka. Korišćeni podmodul Spring Data JPA [4] služi za rad sa relacionim bazama podataka. Pruža generički sloj za manipulaciju podacima i time znatno olakšava razvoj aplikacija.
Osnovna abstrakcija koju Spring Data JPA koristi je repository (repozitorijum). U okviru ovog modula postoje vise vrsta repository interfejsa koji nude različite funkcionalnosti za pristup podacima.
Kreiranje konkretnog repozitorijuma vrši se nasleđivanjem interfejsa koji sadrži željene funkcionalnosti.
Interfejsi koji se nasleđuju su template (šablonski) sa prvim parametrom koji predstavlja tip entiteta koji repozitorijum obrađuje i drugim parametrom koji predstavlja tip primarnog ključa obrađivanog entiteta.
Repozitorijumi ponuđeni za nasleđivanje nude mnoštvo već postojećih metoda za rad sa entitetima kao što su standardne CRUD metode (create, read, update, delete - kreiranje, čiranje, ažuriranje i brisanje) kao i razne metode za osnovne pretrage kao što su metode za pronalaženje svih entiteta i metode za pronalaženje entiteta na osnovu primarnog ključa. Nije potrebno pisati implementaciju ovih metoda već će ona biti automatski pružena.
Pored nasleđenih metoda moguće je deklarisati i nove metode čije ime treba da prati specifikaciju koja je propisana u dokumentaciji i modul će i za njih automatski pružiti implementaciju.
Ovo predstavlja veoma moćan koncept koji znatno ubrzava razvoj sloja za manipulaciju podacima. Korisnicima modula je dodatno na raspolaganje stavljen i mehanizam gde oni sami mogu da odrede šta će definisane metode raditi tako da je pružena mogućnost i za izvršavanje veoma specifičnih funkcionalnosti.
Funkcionalnosti koje modul dodatno i automatski podržava su sortiranje i paginacija, a da bi ih modul za nas implementirao sve što treba da se uradi je da se navedu kao dodatni parametri deklarisanih funkcija. Treba naglasiti da parametar za paginaciju u sebi već sadrži podatke za sortiranje pa kada se koristi nije potrebno dodatno navoditi parametar za sortiranje.

1.4. Spring Data REST

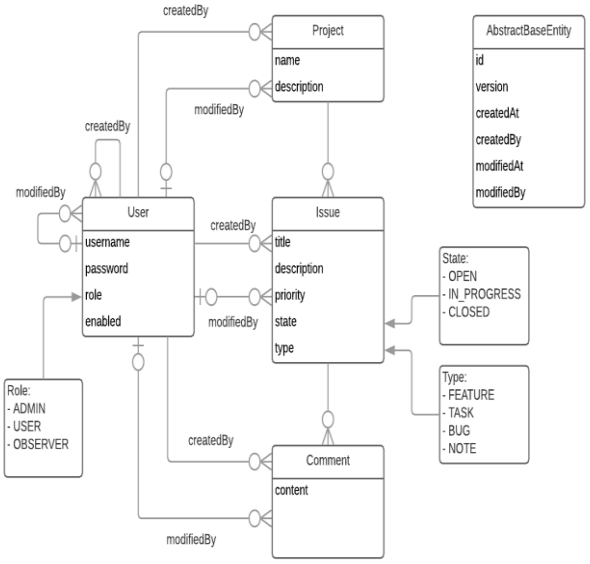
Spring Data REST [5] je modul za ekstenziju sloja za manipulaciju podacima pružajući mogućnost da se definisani repozitorijumi otvore i izlože kao REST API. On zapravo predstavlja sam po sebi Spring aplikaciju i kao takav može da stoji uz već postojeću aplikaciju ili može da predstavlja osnovu pored koje bi se izgradila nova aplikacija.

2. SPECIFIKACIJA

Kao demonstrativnu aplikaciju mogućnosti Spring Data REST modula treba kreirati applikaciju za issue tracking (praćenje zadataka i problema).
Funkcionalnosti aplikacije treba da su prilagođene demonstraciji mogućnosti posmatranog modula Spring-a i ne treba da pokrivaju potpuno domen problema issue trackinga.

2.1. Model podataka

Domenski model sadrži 3 tipa entiteta i to su po hijerarhiji poređani: project (projekat), issue (zadatak) i comment (komentar). Project sadrži kolekciju svojih issue-a, a oni imaju kolekciju svojih comment-a.
Svaki od tipova entiteta sadrži spopstvene informacije vezane za svoju namenu, a zajedničko za sve je da sadrže audit (revizione) podatke. Nije potrebno čuvati istoriju vrednosti već samo podatke o kreaciji i poslednjoj izmeni. Podaci od značaja za izmene su vreme i korisnik koji ih je učinio kao i redni broj verzije entiteta.
Pored entiteta direktno vezanih za domen problema aplikacija sadrži i korisnike. Postoje 3 tipa korisnika, to su admin (administrator), user (korisnik) i observer (posmatrač). Podrazumeva se da administrator ima veća prava od korinika i da može da vrši iste stvari kao korisnik, dok korisnik ima veća prava od posmatrača i da može da vrši iste stvari kao i posmatrač. Dijagram modela dat je na slici 1.



Slika 1. *Dijagram modela*

2.2. Slučajevi korišćenja

Samo ulogovani korisnici smeju da pristupaju resursima sistema. Posmatrači imaju prava da vide sve podatke, ali ih ne mogu menjati. Korisnici mogu da kreiraju issue i comment-e, a mogu da vrše izmene nad entitetom tipa issue bez ograničenja dok comment entitet mogu da menjaju samo ako su ga oni i kreirali. Administator kreira korisnike, ali ih ne može brisati. Brisanje korisnika nije moguće. Administrator kreira, ažurira i briše projekte, a može i da briše entitet tipa issue.

Tabelarni prikaz mogućnosti korisnika po entitetu dat je u tabeli 1.

Legenda:

- C - Pravo kreiranja (create).
- R - Pravo pregleda (read).
- U - Pravo izmene (update).
- U! - Uslovno pravo izmene (update).
- D - Pravo brisanja (delete).
- D! - Uslovno pravo brisanja (delete).

Tip korisnika entiteta	User	Project	Issue	Comment
Admin	C,R,U	C,R,U,D	C,R,U,D	C,R,U!,D!
User	R	R	C,R,U	C,R,U!,D!
Observer	R	R	R	R

Tabela 1. *Mogu'nosti korisnika*

2.3. Dodatne napomene

S obzirom da entitet korisnika sadrži password (lozinku) ona se ne sme čuvati u otvorenom obliku. Lozinka ne sme da bude dostupna preko REST API za čitanje, ali je dostupna za unos i izmenu.

3. IMPLEMENTACIJA

U ovom poglavlju biće opisana implementacija demonstrativne aplikacije sa fokusom na implementaciju njenog backend dela. Pregled implementacije će biti izložen prolazom kroz izabrane funkcionalnosti.

3.1 Osnovna podešavanja

U okviru osnovnih podešavanja moguće je podesiti baznu putanju na kojoj su izloženi REST repozitorijumi, strategiju izlaganja rapozitorijuma preko REST interfejsa. Inicijalno identifikatori entiteta nisu uključeni u JSON odgovora, ali ih je moguće uključiti pojedinačno navodeći klase koje modeluju entitete.

U radu je korišćena zajednička bazna klasa za sve JPA entitete, ali identifikatore nije moguće uključiti navođenjem bazne klase.

3.2 Sigurnosna podešavanja

S obzirom da se repozitorijumi izlažu automatski kao REST API na osnovu izabrane strategije potrebno je posebnu pažnju posvetiti zaštiti.

Zaštita se ostvaruje na standardne načine koji su dostupni sa Spring Security, ali je kao novina moguće zaštitne anotacije koristiti i na metodama koje reaguju na događaje nad repozitorijumom. Ovim mehanizmom je moguće presresti i sprečiti neželjene akcije.

U radu sa REST repozitorijumima je uočeno da se u slučaju kada je kao autentifikacioni mehanizam uključena autentifikaciona forma za neautentifikovane pristupe dobija HTTP odgovor 302 umesto očekivanije 401 i 403.

Autentifikaciona forma je iz navedenih razloga isključena i za autentifikaciju se koristi drugi mehanizam (Basic Auth). Pošto je REST API namenjen konzumiranju od strane drugih aplikacija za slučaj kada su one izvršavane u web pretraživačima potrebno je posesiti CORS (Cross-origin resource sharing) u okviru sigurnosnih podešavanja.

3.3 Model

Model je sačinjen od standardnih JPA entiteta izvođenjem bazne klase koja u sebi satrži identifikator i revizijske podatke. Pored klasa koje modeluju entitete korišćene su i klase za modelovanje projekcija koje služe za kontrolu izlaznog JSON-a time što se pojedini atributi ili asocijacije mogu uključiti ili isključiti. Uz pomoć projekcija moguće je dodati vrednosti koje se isračunavaju evaluacijom SpEL izraza.

Na žalost, u radu sa REST repozitorijumima nije moguće navesti više od jedne projekcije koja bi se primenila na JSON odgovor. Nedostatak oko broja upotrebljenih projekcija donekle može da se reši nasleđivanjem s obzirom da se projekcije definišu kao interfejsi.

Za olakšavanje razvoja korišćena je biblioteka Lombok koja generiše set/get metode za pristup atributima entiteta. Pored JPA/Hibernate anotacija u modelu su korišćene i anotacije specifične za Jackson, komponentu koja vrši serijalizaciju između Java objekata i JSON-a. Korisnička lozinka je ovim mehanizmom izbačena iz JSON odgovora, ali je korišćena anotacija koja dozvoljava izmenu kroz REST repozitorijume.

3.4 Repozitorijumi

Ključnu tačku u okviru modula Spring Data REST predstavljaju JPA repozitorijumi kojima se kreiraju REST resursi. Moguća je kontrola omogućenih HTTP metoda koje odgovaraju metodi ili metodama repozitorijuma. U slučaju entiteta korisnika u odgovarajućem repozitorijumu je onemogućeno brisanje tako što su sve metode brisanja bile označene anotacijom koja ograničava izlaganje.

Metode pretrage koje su definisane u okviru repozitorijuma je moguće pozvati preko REST-a jer su izložene na */search/* podputanjama resursa pod istim nazivom. Na sve CRUD (create, update, delete) metode repozitorijuma moguće reagovati implementacijom *event handlera*. Ovaj princip je iskorišćen za transformisanje lozinke korisnika.

4. ZAKLJUČAK

Spring Data REST predstavlja dobro rešenje za generisanje REST API-ja aplikacija. Posebna prednost mu je to što lako može da koegzistira sa već postojećim ručno napisanim REST API-jem.

Nudi mogućnost prilagođavanja potrebama aplikacije, a nudi i klase za ručno generisanje REST interfejsa.

Pri radu je bitno dobro upoznati ponašanje modula jer nije baš sve onako kako bi se očekivalo, o ovome je najviše bilo reči u radu u sekciji implementacije demonstrativnog projekta.

S obzirom da Spring Data REST otvara generički REST API potrebno je značajnije posvetiti pažnju bezbednosnim aspektima jer vrlo lako i neželjeno mogu da se otvore funkcionalnosti i podaci koji ne bi smeli biti dostupni.

Kao što ni REST nije dobro rešenje za sve vrste aplikacije tako ni Spring Data REST nije rešenje za sve. Domen primene bi mu mogle biti aplikacije sa formama gde postoji obrada podataka jednostavnih struktura.

Aplikacije koje imaju složene izlazne podatke možda nisu pogodne za implementaciju korišćenjem Spring Data REST modula, ali i kod njih bismo savetovali da se deo aplikacije koji je pogodan za REST implementira korišćenjem pomenutog modula.

5. LITERATURA

[1] https://www.ics.uci.edu/~fielding/pubs-/dissertation/top.htm (pristupljeno u avgustu 2018. godine)

[2] https://martinfowler.com/articles/-richardsonMaturityModel.html (pristupljeno u avgustu 2018. godine)

[3] https://en.wikipedia.org/wiki/Spring_Framework (pristupljeno u avgustu 2018. godine)

[4] https://docs.spring.io/spring-data/jpa/docs/current/reference/html/ (pristupljeno u avgustu 2018. godine)

[5] https://docs.spring.io/spring-data/rest/docs/current/reference/html/ (pristupljeno u avgustu 2018. godine)

Kratka biografija:

Igor Gere rođen je u Novom Sadu 1982. godine. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva - Računarske nauke i informatika odbranio je 2018. godine.