Tallinna Ülikool

Digitehnoloogiate instituut

Spring Boot raamistik

Seminaritöö

Autor: Karl Oha

Juhendaja: Jaagup Kippar

Tallinn 2017

Sisukord

Sissejuhatus 3

1 Mis on Spring Boot 4

1.1 Spring Boot raamistiku nõuded 4

1.2 Spring Boot installeerimine 5

2 Spring Boot võimalused 6

2.1 Koodi struktuur ja automaatne seadistus 6

2.2 Spring Boot starterid 7

2.3 Arendustööriistad 7

2.4 Andmebaasid 8

2.5 Testimine 10

2.5.1 Komponent testid 10

2.5.2 Integratsiooni testid 11

2.6 Turvalisus 15

2.6.1 Spring Security 15

2.6.2 OAuth 2 17

3 Spring Boot haldus 19

3.1 Seire 19

3.2 Logimine 19

3.3 Paigaldamine 20

Kokkuvõte 21

Kasutatud kirjandus 22

Sissejuhatus

Seminaritöö eesmärgiks on tutvustada Spring Boot raamistikku ning tuua välja selle võimalused. Töös sisalduv informatsioon ning näited põhinevad Spring Boot 1.4.1.RELEASE versioonil. Seminaritöö on teoreetiline ülevaade raamistiku võimalustest.

Seminaritöö on jaotatud kolmeks peatükiks. Esimene peatükk tutvustab Spring Boot raamistikku, selle nõudeid ja paigaldamist.

Teine peatükk keskendub Spring Boot funktsionaalsustele ning võimalustele. Teise peatüki alapeatükid tutvustavad lähemalt Spring Boot võimalusi, mis on suurimateks raamistiku eelisteks konkurentide ees.

Kolmas peatükk räägib teemadel, mis puudutavad rakenduse haldust. See peatükk on jaotatud kolmeks alapeatükiks, kus räägin seirest, logimisest ja paigaldamisest.

Teema valisin seoses tööalase kogemusega erinevate Spring Boot rakenduste arendamisel. Raamistikul puudus ka eestikeelne materjal. Materjalist arusaamise eelduseks on teadmised programmeerimiskeelest Java ning kasuks tulevad ka teadmised Spring raamistikust. Töö sisaldab Spring Boot raamistiku omapärasid, mis kohati võivad kattuda Spring raamistikuga.

1 Mis on Spring Boot

Spring Boot on osa Java raamistikust Spring[1], mis on disainitud võimaldamaks luua kvaliteetset Spring rakendust kiirelt ja lihtsalt. Selle tagamiseks eelistab Spring Boot konventsiooni konfiguratsioonile. Spring Boot valib rakenduse seadistused ja kolmanda osapoole teeke (*libraries*) sinu eest, mis võimaldavad rakenduse loomisel keskenduda arendamisele ja mitte seadistamisele. Enamus rakendused vajavad vähe seadistamist, kuid Spring Boot jätab võimaluse seadistada rakendust samaväärselt Spring-le (Webb et al., kuupäev puudub).

1.1 Spring Boot raamistiku nõuded

Spring Boot raamistik vajab Java 7 ning Spring Framework 4.3.3.RELEASE või uuemat versiooni. Kasutada võib ka Java 6 koos lisa seadistustega, kuid ametlikult on soovitatav kasutada Java 8, et saada kasu kõikidest Spring Boot võimalustest. Tehniline tugi on tagatud alates Maven[2] 3.2 ning Gradle[3] 1.12 ja 2.x versioonist. Spring Boot toetab järgnevaid sisse ehitatud servlet (*servlet*) konteinereid (Tabel 1) (Webb et al., kuupäev puudub).

Tabel 1. Servlet versioonid

Nimi	**Servlet versioon**	**Java versioon**
Tomcat 8	3.1	Java 7+
Tomcat 7	3.0	Java 6+
Jetty 9.3	3.1	Java 8+
Jetty 9.2	9.1	Java 7+
Jetty 8	3.0	Java 6+
Undertow 1.3	3.1	Java 7+

[1] https://spring.io/

[2] https://maven.apache.org/

[3] https://gradle.org/

1.2 Spring Boot installeerimine

Spring Boot installeerimine ei nõua spetsiaalset tööriista ning toimib nagu tavalise java teegi lisamine. Lisada tuleb vastava versiooni spring-boot-*.jar fail oma rakenduse klassiteesse (*classpath*). Kuigi tavapärane *jar* faili kopeerimine toimib, soovitab Spring Boot kasutada mõnda rakenduse ehitus tööriista, mis toetab sõltuvuste haldust (*dependency management*). Maven-ga Spring Boot rakenduse loomisel tuleb lisada *org.springframework.boot* maven pom.xml faili (Joonis 1), gradle puhul build.gradle faili (Joonis 2) (Webb et al., kuupäev puudub).

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.4.1.RELEASE</version>
</parent>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
</dependencies>
```

Joonis 1. Sõltuvuse lisamine Maven-i

```
dependencies {
    compile("org.springframework.boot:spring-boot-starter-
web:1.4.1.RELEASE")
}
```

Joonis 2. Sõltuvuse lisamine Gradle

2 Spring Boot võimalused

2.1 Koodi struktuur ja automaatne seadistus

Spring Boot lisab annotatsiooni (*annotation*) @SpringBootApplication (Joonis 3), mis sisaldab arendajate poolt enimkasutatud peamise (*main*) meetodi annotatsioone @Configuration, @EnableAutoConfiguration ja @ComponentScan (Joonis 4). Rakendus käivitatakse SpringApplication.run() meetodit välja kutsudes rakenduse main() meetodist (Webb et al., kuupäev puudub).

```
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Joonis 3. @SpringBootApplication annotatsioon

@EnableAutoConfiguration annotatsioon seadistab automaatselt klassiteel asuvad oad (*beans*). Näiteks *tomcat-embedded.jar* olemasolul klassiteel seadistab @EnableAutoConfiguration rakenduse käivitamisel kasutama sisse ehitatud tomcat servelt konteinerit. @Configuration annotatsioon java klassi peal näitab, et klass sisaldab @Bean annotatsiooni ubade seadistamiseks. @Configuration ja @Bean on alternatiivid traditsioonilisele Spring xml faili <*bean*/> seadistustele. @ComponentScan annotatsioon skaneerib rakenduse vaikimisi paketist (*package*) @Controller, @Component, @Service, @Repository jt annotatsioone ning seadistab kõik rakenduse oad vaikimisi lisatuks. @ComponentScan annotatsioon on alternatiiv traditsioonilisele Spring xml faili <*context:component-scan*> seadistusele.

```
@Configuration
@EnableAutoConfiguration
@ComponentScan
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Joonis 4. @Configuration, @EnableAutoConfiguration ja @ComponentScan

Spring Boot ei vaja kindalt koodi struktuuri, kuid eelistatud on parimate tavade kasutamine (Joonis 28). Kui java klass (*class*) ei sisalda paketti, siis Spring Boot loeb klassi asukohaks vaikimisi paketi. Vaikimisi paketi kasutamine on enamasti mitte soovitatav, kuna võib tekitada probleeme rakendustes, mis kasutavad @ComponentScan, @EntityScan või @SpringBootApplication annotatsioone.

@EnableAutoConfiguration annotatsioon aktiveerib rakenduse automaatse seadistamise. Rakendus seadistatakse vastavalt klassiteel olevatele teekidele ning käsitsi lisatud seadistustele. Spring-boot-starter-web-services sõltuvuse lisamisega kaasnevad Tomcat ja Spring MVC mille järel seadistatakse veebi rakendus. Spring automaatne seadistus on disainitud toimima koos starteritega (*statrters*). Kolmanda osapoole teekide lisamisel võib Spring vajada manuaalset seadistamist. Spring Boot automaatne seadistus eelistab java klasside põhiseid seadistusi, kuid suudab ka laadida seadistusi xml failist (Webb et al., kuupäev puudub).

2.2 Spring Boot starterid

Spring Boot pakub alustamiseks sobilikke starter teeke, mis aitavad kiiresti lisada hulgaliselt eelseadistatud funktsionaalsust. Ametlikud starterid kannavad nime spring-boot-starter-*, kus * tähistab teegi tüüpi. Veebirakenduse loomisel piisab spring-boot-starter-web-services starteri lisamisest, mis sisaldab endas alustamiseks vajalikke teeke. Starterid jagunevad kolme tüüpi: rakendus, tehnilised ja haldus starterid. Täielik nimekiri starteritest on saadavad vastava Spring Boot versiooni dokumentatsioonis alapeatükis *„Starters“*[4].

2.3 Arendustööriistad

Spring Boot rakendust saab lihtsalt arendada enamus java integreeritud arenduskeskkondades (IDE - *Integrated Development Environment*). Spring Tool Suite[5] (STS) on kõik ühes Eclipse baasile arendatud IDE. STS pakub valmis arenduskeskkonda rakenduse arendamiseks, silumiseks, käivitamiseks ja juurutamiseks ning sisaldab rohkelt enimkasutatud kolmanda osapoolte keskkondade tuge (Pivotal Software, Inc., kuupäev puudub).

Spring Boot käsurea tööriista (CLI[6]) on disainitud kiirelt erinevate funktsionaalsuste katsetamiseks. CLI võimaldab jooksutada Groovy[7] skripte, mis kasutab java laadset

[4] http://docs.spring.io/spring-boot/docs/1.4.1.RELEASE/reference/htmlsingle/#using-boot-starter
[5] https://spring.io/tools
[6] https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#cli
[7] http://www.groovy-lang.org/

süntaksit. CLI kasutamine ei ole vajalik, kuid see on kindlasti kiireim viis Spring Boot rakenduse loomiseks (Webb et al., kuupäev puudub).

2.4 Andmebaasid

Spring Boot toetab suurel hulgal SQL *(Structured Query Language)* andmebaase. SQL andmebaasiga suhtlemiseks kasutab Spring Boot JDBC (*Java Database Connectivity*) või ORM (*Object-relational mapping*) tehnoloogiaid. *JdbcTemplate* ja *NamedParameterJdbcTemplate* klassid on automaatselt seadistatud ning lisatavad komponentidele (*Beans*) kasutades @Autowired annotatsiooni (Joonis 5) (Webb et al., kuupäev puudub).

```
@Component
public class MyBean {
    private final JdbcTemplate jdbcTemplate;
    @Autowired
    public MyBean(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }
    // ...
}
```

Joonis 5. JdbcTemplate lisamine komponendile

JPA (*Java Persistence API*) on standardne tehnoloogia, mis võimaldab kaardistada java objekte relatsioonandmebaasiga. Spring-boot-starter-data-jpa starter pakub kiiret lahendust alustamiseks sisaldades Hibernate[8], Spring Data JPA[9] ja Spring ORM[10] sõltuvusi. Traditsioonilises Spring rakenduses kirjeldatakse entiteet (*Entity*) klassid persistence.xml faili. Spring Boot kasutad entiteet klasside leidmiseks automaatset otsingut (*Entity Scanning*). Vaikimisi otsitakse kõikidest pakettidest, mis asuvad allpool peamist (@EnableAutoConfiguration või @SpringBootApplication annoteeritud klass) seadistus klassist. Klassid mis sisaldavad @Entity, @Embeddable või @MappedSuperclass annotatsiooni arvestatakse entiteet klassiks. Spring Data JPA lisab rakendusele funktsionaalsuse võimaldamaks genereerida andmebaasi päringuid otse hoidla (*repository*) meetodist (Joonis 6).

[8] http://hibernate.org/
[9] http://projects.spring.io/spring-data-jpa/
[10] http://docs.spring.io/spring/docs/current/spring-framework-reference/html/orm.html

```
public interface EmployeeRepository extends CrudRepository<Employee, Long>
{
    Employee findByFirstName(String firstName);
    List<Employee> findByLastName(String lastName);
}
```

Joonis 6. Spring Data JPA hoidla meetod

Java javax.sql.DataSource liidest pakub standardse viisi andmebaasiga ühendamiseks. DataSource seadistamiseks tuleb lisada application.properties faili andmebaasi URL (*uniform resource locator*) koos mõningate lisa parameetritega (Joonis 7).

```
spring.datasource.url=jdbc:mysql://localhost/test
spring.datasource.username=dbuser
spring.datasource.password=dbpass
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
```

Joonis 7. DataSource parameetrid

Spring Boot toetab ka mitmeid mälusse integreeritavaid andmebaase nagu H2[11], HSQL[12] ja Derby[13]. Nende andmebaasidega töötamisel ei ole vaja seadistada DataSource vaid piisab sõltuvuse lisamisest Maven pom.xml või Gradle build.gradle faili (Joonis 8).

```
<dependency>
    <groupId>org.hsqldb</groupId>
    <artifactId>hsqldb</artifactId>
    <scope>runtime</scope>
</dependency>
```

Joonis 8. Mälusse integreeritava andmebaasi sõltuvuse lisamine pom.xml faili

```
dependencies {
     compile group: 'com.h2database', name: 'h2', version: '1.4.193'

}
```

Joonis 9 Mälusse integreeritava andmebaasi sõltuvuse lisamine build.gradle faili

Spring Data[14] pakub mitmeid lisaprojekte, mis võimaldavad kasutada erinevaid NoSQL (*non SQL*) tehnoloogilal põhinevaid andmebaase, sealhulgas MongoDB[15],

[11] http://www.h2database.com/html/main.html
[12] http://hsqldb.org/
[13] http://db.apache.org/derby/
[14] http://projects.spring.io/spring-data/
[15] https://www.mongodb.com/

Neo4J[16], Elasticsearch[17], Solr[18], Redis[19], Gemfire[20], Couchbase[21] ja Cassandra[22]. Spring Boot pakub automaatset seadistust MongoDB, Neo4J, Elasticsearch, Solr, Redis ja Cassandra, teiste Spring Data poolt pakutavate NoSQL andmebaasi projektid tuleb ise seadistada (Webb et al., kuupäev puudub).

2.5 Testimine

Spring Boot pakub hulgaliselt tööriistu ja annotatsioone aitamaks rakenduse testimisel. Testimist toetab kaks põhi moodulit spring-boot-test, mis sisaldab põhifunktsionaalsust ning spring-boot-test-autoconfigure, mis sisaldab automaatset testide seadistamist. Ametlikult on soovitatav kasutada spring-boot-starter-test starterit, mis sisaldab mõlemat moodulit ning lisaks JUnit[23], AssertJ[24], Hamcrest[25], Spring Test & ja Spring Boot Test, Mockito[26], JSONAssert[27] ja JsonPath[28] teeke. Need on Spring Boot poolt toetatud kolmanda osapoole teegid (Webb et al., kuupäev puudub).

2.5.1 Komponent testid

Spring Boot rakenduse komponent (*unit*) testimine toimib sarnaselt tüüpilisele Spring rakendusele ja ei vaja lisaseadistusi. Üks suurimaid eeliseid sõltuvuste lisamisel on võimalus testida rakendust ilma käivitamata. Testimiseks vajalikke objekte saab luua kasutades uus (*new*) operaatorit ning objekte on võimalik ka võltsida (*mock*). Alates Spring Framework (versioonist) 4.3 on lihtne luua komponente kasutades konstruktori lisamist (*constructor injection)* mis ei vaja enam @Autowired annotatsiooni. Vähemalt ühe konstruktori olemasolul arvestab Spring välja vaikimisi kui lisatud (*autowired*) (Joonis 10) (Webb, 2016).

[16] https://neo4j.com/
[17] https://www.elastic.co/
[18] https://lucene.apache.org/solr/
[19] http://redis.io/
[20] https://pivotal.io/big-data/pivotal-gemfire
[21] http://www.couchbase.com/
[22] http://cassandra.apache.org/
[23] http://junit.org/junit4/
[24] http://joel-costigliola.github.io/assertj/
[25] http://hamcrest.org/JavaHamcrest/
[26] http://site.mockito.org/
[27] https://github.com/skyscreamer/JSONassert
[28] https://github.com/jayway/JsonPath

```
@Component
public class MyComponent {
    private final SomeService service;
    public MyComponent(SomeService service) {
        this.service = service;
    }
    public String someMethod(){
        return service.getSomeString();
    }
}
```

Joonis 10. Konstruktori lisamine

MyComponent klassi testimisel on vaja lihtsalt luua objekt ja kutsuda välja testitav meetod ning vajadusel võltsida väliseid sõltuvusi kasutades mock() meetodit (Joonis 11).

```
@Test
public void testSomeMethod() {
    SomeService service = mock(SomeService.class);
    MyComponent component = new MyComponent(service);
    when(service.getSomeString()).thenReturn("Some String");
    String someString = component.someMethod();
    assertEquals("Some String", someString);
}
```

Joonis 11. Komponent test

2.5.2 Integratsiooni testid

Lisaks komponent testidele on tihti vajadus ka rakendusele lisada integratsiooni testid. Spring Boot pakub selleks @SpringBootTest annotatsiooni. @SpringBootTest asendab @ContextConfiguration annotatsiooni ning lisab testi algseadistamisel Spring Boot funktsionaalsuse ja laeb application.properties faili seadistused. Integratsiooni testi klassidele tuleb lisada @RunWith(SpringRunner.class), mis käivitab testi kasutades Spring Test moodulit. SpringRunner.class on uus klass alates Spring Boot 1.4 versioonist, mis asendab varasemat SpringJUnit4ClassRunner.class-i (Webb, 2016).

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class MyTest {
    // ...
}
```

Joonis 12. @SpringBootTest annotatsioon

Spring Framework ja Spring Boot varasemate versioonide puhul integratsiooni testimisel tuli tihtipeale kasutada mõnda järgnevatest kombinatsioonidest (Lisa 1):

1) @ContextConfoguration ja SpringApplicationContextLoader (Joonis 29)
2) @SpringApplicationConfiguration (Joonis 30)
3) 1 või 2 ja @IntegrationTest (Joonis 31)
4) 1 või 2 ja @WebIntegrationTest (Joonis 32)
5) 1 või 2 ja @IntegrationTest + @WebAppConfiguration (Joonis 33)

@SpringBootTest seadistab automaatselt ka TestRestTemplate, mis võimaldab testida reaalset rakenduse REST (*Representational state transfer*) lõpp-punkte (*endpoint*). TestRestTemplate on eelseadistatud lahendama suhtelisi lõpp-punkte aadressil *http://localhost:${local.server.port}*. Spring Boot pakub ka võimalust rakenduse teenuseid võltsida ja luurata (*spy*), kasutades selleks @MockBean ja @SpyBean annotatsioone (Joonis 11). @DirtiesContext annotatsiooni pole vaja kasutada, kuna võltsingud algseadistatakse peale igat testi automaatselt (Webb, 2016).

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class SampleTestApplicationWebIntegrationTests {
    @Autowired
    private TestRestTemplate restTemplate;
    @MockBean
    private VehicleDetailsService vehicleDetailsService;
    @SpyBean
    private VehicleService vehicleService;
    @Before
    public void setup() {
        // setup mock
    }
    @Test
    public void test() {
        // ...
    }
}
```

Joonis 13. @MockBean ja @SpyBean annotatsioon

Spring Boot võimaldab testimiseks automaatselt seadistada ainult selleks vajamineva osa rakendusest. Andmebaasi päringute testimiseks mõeldud @DataJpaTest annotatsioon seadistab automaatselt mälu sisese andmebaasi ja Spring Data (Joonis 14) (Webb, 2016).

```
@RunWith(SpringRunner.class)
@DataJpaTest
public class MyJpaRepositoryTests {
    @Autowire
    private JpaRepository jpaRepository
    @Test
    public void getSomeDataTest();
    assertEquals(sameData, jpaRepository.getSomeData());
}
```

Joonis 14. @DataJpaTest annotatsioon

MVC kontrollerite testimiseks on loodud @WebMvcTest annotatsioon, mis laeb rakenduse kontrollerid ning seadistab Spring MVC, Jackson, Gson, Message jne muundurid ning MockMVC (Joonis 15) (Webb, 2016).

```
@RunWith(SpringRunner.class)
@WebMvcTest(SomeController.class)
public class MyControllerTests {
    @Autowired
    private MockMvc mockMvc;
    @MockBean
    private SomeService someservice;
    @Test
    public void testSomeControllerGetEndpoint(){
        when(someservice.getSomeServiceData()).thenReturn(someData);
        mockMvc.perform(get("/someEndpoint"))
        .andExpect(status().isOk())
    }
}
```

Joonis 15. @WebMvcTest annotatsioon

@JsonTest annotatsioon seadistab Jackson ja/või Gson ning laeb @JsonComponent klassid, et testida JSON *serialization* ja *deserialization* (Joonis 16) (Webb, 2016).

```
@RunWith(SpringRunner.class)
@JsonTest
public class MyJsonTests {
    private JacksonTester<SomeObject> json;
    @Test
    public void serialize() throws IOException {
        SomeObject object = new SomeObject("Some json data");
        JsonContent<SomeObject> write = this.json.write(object);
        assertThat(this.json.write(object)).isEqualToJson("expected.json");
    }
    @Test
    public void deserialize() throws IOException {
        String content = "{\"data\":\"Some json data\"}";
        assertThat(this.json.parse(content)).isEqualTo(new SomeObject("Some
json data"));
    }
}
```

Joonis 16. @JsonTest annotatsioon

Rest klientide testimiseks saab kasutada @RestClientTest annotatsiooni, mis seadistab automaatselt Jackson ja/või Gson ja RestTemplateBuilder ning lisab MockRestServiceServer toe (Joonis 17) (Webb, 2016).

```
@RunWith(SpringRunner.class)
@RestClientTest({SomeClient.class, SomeClientProperties.class})
public class SomeDataClientTests {
    @Rule
    public ExpectedException thrown = ExpectedException.none();
    @Autowired
    private SomeClient client;
    @Autowired
    private MockRestServiceServer server;
    @Test
    public void getSomeDataFromRestService() {
        server.expect(requestTo("/")).andRespond(withSuc-
cess("{\"data\":\"Some data from REST service\"}", MediaType.APPLICA-
TION_JSON));
        SomeDataClientResponse response = client.getSomeData();
        assertThat(response.getData()).isEqualTo("Some data from REST ser-
vice");
    }
}
```

Joonis 17. @RestClientTest annotatsioon

Nimekiri seadistustest, mis lisatakse nende annotatsioonide kasutamisel on leitav vastava Spring Boot versiooni dokumentatsioonist alapeatükist „Appendix D. Test auto-configuration annotations[29]“ (Webb, 2016).

[29] http://docs.spring.io/spring-boot/docs/1.4.1.RELEASE/reference/htmlsingle/#test-auto-configuration

2.6 Turvalisus

2.6.1 Spring Security

Spring Security[30] sõltuvuse lisamisel klassiteele seadistatakse automaatselt kõikidele veebirakenduse HTTP lõpp-punktidele autentimine. Spring Security on ülimalt võimas ja kohaldatav autentimise ja ligipääsu kontrolli tagamis raamistik. Meetodi tasemel turvalisuse lisamiseks veebirakendusele tuleb lisada @EnableGlobalMethodSecurity koos soovitud seadistustega (Joonis 18).

```
@Configuration
@EnableGlobalMethodSecurity
public class MethodSecurityConfig extends GlobalMethodSecurityConfiguration
{
  // Security settings
}
```

Joonis 18. @EnableMethodSecurity annotatsioon

Lisa informatsiooni meetodi taseme turvalisusest leiab Spring Security dokumentatsioonist alapeatükist „Method Security[31]“. Turvalisuse lisamisel seatakse vaikimise AuthenticationManager ühe kasutajaga. Kasutaja vaikimisi kasutajanimi on „user“ ning parool juhuslik, mis trükitakse INFO tasemel rakenduse logisse rakenduse käivitumisel (Joonis 19).

```
Using default security password: 78fa095d-3f4c-48b1-ad50-e24c31d5cf35
```

Joonis 19. Juhuslik parool logis

Kasutaja informatsiooni on võimalik muuta lisades „security“ eesliidesega parameetrid application.properties faili (Joonis 20).

```
security.user.name=admin
security.user.password=secret
management.security.roles=SUPERUSER
```

Joonis 20. Kasutaja parameetrid

Vaikimisi turvalisuse seadistused on realiseeritud SecurityAutoConfiguration klassis, mis impordib SpringBootWebSecurityConfiguration klassi veebirakenduse turvalisuseks ja AuthenticationManagerConfiguration klassi autentimise seadistustega. Vaikimisi veebirakenduse turvalisuse välja lülitamiseks tuleb lisada

[30] http://projects.spring.io/spring-security/
[31] http://docs.spring.io/spring-security/site/docs/4.1.3.RELEASE/reference/htmlsingle/#jc-method

@EnableWebSecurity ja @Configuration annotatsioon klassile, mis laiendab WebSecurityConfigurationAdapter klassi (Joonis 21).

```
@Configuration
@EnableWebSecurity
public class MyWebSecurityConfiguration extends
WebSecurityConfigurerAdapter {
    // Customized security settings
}
```

Joonis 21 @EnableWebSecurity annotatsioon

Vaikimise seadistused ei lisa autentimise vajadust tüüpilistele staatilise ressursi asukohtadele nagu /css/**, /js/**, /images/**, /webjars/** and **/favicon.ico. Edukad ja edutud autentimise ja ligipääsu katsed edastatakse automaatselt ApplicationEventPublisher klassile. Spring Security seadistab ja aktiveerib automaatselt HSTS (*Strict Transport Security*), XSS (*Cross-site scripting*), CSRF (*Cross-Site Request Forgery*) ja vahemälu (*caching*) turvalisuse funktsionaalsused. Kõiki funktsionaalsusi saab välja lülitada või muuta kasutades vastavaid security.* parameetreid. Olemasolevate reeglite üle kirjutamiseks ilma automaatseid seadistusi muutmata tuleb lisada WebSecurityConfigurerAdapter tüüpi @Bean koos @Order(SecurityProperties.ACCESS_OVERRIDE_ORDER) ning seadistada see vastavalt oma vajadustele (Joonis 22).

```
@Order(SecurityProperties.ACCESS_OVERRIDE_ORDER)
public static class WebSecurityConfig extends WebSecurityConfigurerAdapter
{
    // Customized security settings
}
```

Joonis 22. Turvalisuse reeglite ülekirjutamine

Spring Boot pakub hulgaliselt näidis rakendusi, mis sisaldavad ka erinevaid turvalisuse näiteid. Näidisrakendused asuvad Spring Boot Github[32] koodihoidlas spring-boot-samples[33] kataloogis.

Spring Boot actuator lõpp-punktid on vaikimisi turvatud ka siis, kui rakenduse lõpp-punktid on turvamata. Turvalisuse sündmused transformeeritakse auditi sündmuseks (*AuditEvents*) ning edastatakse AuditService klassile. Vaikimisi kasutajal on administraatori ja kasutaja roll. Acurator turvalisus parameetreid saab muuta kasutades management.security.* eesliidest (Webb et al., kuupäev puudub).

[32] https://github.com/
[33] https://github.com/spring-projects/spring-boot/tree/master/spring-boot-samples

2.6.2 OAuth 2

OAuth 2[34] on uus versioon OAuth[35] protokollist. OAuth on avatud protokoll, mis võimaldab turvalist volitamist lihtsal ja standardsel viisil veebi-, mobiili- ja töölauarakendustele. OAuth 2 lisamiseks tuleb klassiteele lisada spring-security-oauth2 sõltuvus. Spring Boot lisab mõningad automaatsed seadistused aitamaks paigaldada volitus< või ressursi serverit. Täpsemate juhiste kohta leiab infot Spring Security OAuth2 dokumentatsioonist ja juhenditest[36] (Webb et al., kuupäev puudub).

OAuth 2 kliendi loomiseks Spring Boot veebirakenduses tuleb lisada @EnableOAuth2Client annotatsioon (Joonis 23).

```
@Configuration
@EnableOAuth2Client
public class MyWebSecurityConfiguration extends
WebSecurityConfigurerAdapter {
  // ...
}
```

Joonis 23. @EnableOAuth2Client annotatsioon

@EnableOAuth2Client loob automaatselt OAuth2ClientContext ja OAuth2ProtectedResourceDetails, mis on vajalikud OAuth2RestOperations loomiseks. Vastab uba (*bean*) tuleb ise luua (Joonis 24).

```
@Bean
public OAuth2RestTemplate oauth2RestTemplate(OAuth2ClientContext
oauth2ClientContext,
        OAuth2ProtectedResourceDetails details) {
    return new OAuth2RestTemplate(details, oauth2ClientContext);
}
```

Joonis 24 OAuth2RestTemplate uba

Selline seadistus kasutab security.oauth2.client.* parameetreid ning lisaks serveri volituse ja žetooni uri-d (*Uniform Resource Identifier*), mis tuleb lisada application.yml faili (Joonis 25) (Webb et al., kuupäev puudub).

[34] https://oauth.net/2/
[35] https://oauth.net/
[36] http://projects.spring.io/spring-security-oauth/docs/oauth2.html

```
security:
    oauth2:
        client:
            clientId: bd1c0a783ccdd1c9b9e4
            clientSecret: 1a9030fbca47a5b2c28e92f19050bb77824b5ad1
            accessTokenUri: https://github.com/login/oauth/access_token
            userAuthorizationUri: https://github.com/login/oauth/authorize
            clientAuthenticationScheme: form
```

Joonis 25. application.yml seadistused

OAuth 2 volitamis serveri loomiseks tuleb lisada @EnableAuthorizationServer ning määrata security.oauth2.client.client-id ja security.oauth2.client.client-secret parameetrid application.properties faili. Klient registreeritakse rakenduse mälu sisesesse hoidlasse. Ligipääsu žetooni loomiseks saab kasutada kliendi parameetreid client-id ja client-secret ning kasutaja parameetrid kasutajanimi „user“ ja vaikimisi Spring Security juhuslikku parooli (Joonis 26).

```
$ curl client:secret@localhost:8080/oauth/token -d grant_type=password -d
username=user -d password=pwd
```

Joonis 26. Ligipääsu žetooni loomine

Automaatsete seadistusete välja lülitamiseks tuleb lisada @Bean AuthorizationServerConfigurer.

Ressursi serveri loomiseks tuleb lisada @EnableResourceServer ning mõningad lisa parameetrid, mis võimaldab serveril dekodeerida ligipääsu žetoone. Kui rakendus on volitamis server siis puudub vajadus ressursi serveri lisamiseks, kuna volitamis server sisaldab ligipääsu žetooni dekodeerimis võimekust (Webb et al., kuupäev puudub).

3 Spring Boot haldus

3.1 Seire

Spring Boot pakub hulgaliselt võimalusi jälgida oma rakendust kogu elutsükli jooksul. Seire funktsionaalsuste lisamiseks on Spring Boot loonud starteri spring-boot-starter-actuator. Starter sisaldab endas automaatset rakenduse auditeerimise, oleku ja erinevate mõõdikute jälgimise funktsionaalsusi. Starteri lisamisel seadistatakse automaatselt HTTP (*Hypertext Transfer Protocol*) lõpp-punktid koos JMX (*Java Management Extensions*) ja SSH (*Secure Shell*). HTTP lõpp-punktid on saadaval ainult Spring MVC baasil põhinevatel rakendustel. Enamus rakendused kasutavad HTTP seiret, mis paljastab määratud lõpp-punkte. Rakenduse oleku jälgimise vaikimisi lõpp-punkt on „/*health*“. Spring Boot võimaldab kõiki automaatselt lisatud lõpp-punkte ka seadistada ning vajadusel lisada juurde kohandatud lõpp-punkte (Webb et al., kuupäev puudub).

3.2 Logimine

Spring Boot kasutab Common Logging API[37] kõikide rakendussiseste tegevuste logimiseks. Logimisfunktsionaalsuse lisamiseks soovitab Spring Boot kasutada spring-boot-starter-logging starterit. Spring-boot-starter-web starteri olemasolul ei ole vaja logimise starterit enam lisada kuna *web* starter juba sisaldab sama funktsionaalsust. Vaikimisi seadistused on lisatud Java Util Logging[38], Log4J2[39] ja Logback[40] teekidel. Need teegid on eelseadistatud kasutama konsooli väljundit ning valikuliselt on saadaval ka faili väljund. Starteri lisamisel on vaikimisi seadistatud kasutama Logback teeki. Kõiki toetatud logimis süsteemide taset saab seadistada Spring keskkonna muutujatega. Tasemete muutmiseks tuleb kirjeldada application.properties faili logging.level.*=LEVEL’ kus ‘LEVEL’ on TRACE, DEBUG, INFO, WARN, ERROR, FATAL või OFF (Joonis 27) (Webb et al., kuupäev puudub).

[37] http://commons.apache.org/proper/commons-logging/
[38] http://docs.oracle.com/javase/7/docs/api/java/util/logging/package-summary.html
[39] http://logging.apache.org/log4j/2.x/
[40] http://logback.qos.ch/

```
logging.level.root=WARN
logging.level.org.springframework.web=DEBUG
logging.level.org.hibernate=ERROR
```

Joonis 27. Logimise tasemed

3.3 Paigaldamine

Spring Boot võimaldab paindlikku rakenduse pakendamis (*packaging*) võimalust, mis pakub hulgaliselt võimalusi rakenduse paigaldamiseks (*deploying*). Rakendus on kergesti paigaldatav paljudele pilvel põhinevatele platvormidele, konteineritesse, virtuaalsetesse serveritesse ja füüsilistesse serveritesse. Spring Boot täidetav (*executable*) purgi (*jar*) faili on valmis pakendatud ning sobivad enamikele populaarsetele PaaS (*Platform as a service*) teenuse pakkujatele. Unix/Linux teenusena käivitamiseks tuleb rakendus pakendada kui täielikult käivitatav (*fully executable*) rakendus ning käivitamisel kasutada init.d või systemd. Windows-i teenusena käivitamiseks saab kasutada winsw[41] (Webb et al., kuupäev puudub).

[41] https://github.com/kohsuke/winsw

Kokkuvõte

Käesolev seminaritöö tutvustas Spring Boot raamistikku ning selle võimalusi. Seminaritöö sisaldab erinevaid võimalusi illustreerivaid koodinäiteid lihtsustamaks arusaama funktsionaalsuse kasutamisest.

Esimene peatükk tutvustas Spring Boot rakendust ning selle nõudeid. Lühidalt vaatasime ka kuidas Spring Boot rakendust installeerida.

Seminaritöö teine peatükk keskendus Spring Boot raamistiku võimalustele. Lähemalt vaatasime pakutavaid arendustööriistasid, andmebaasi sobivusi, testimisvõimalusi ning rakenduse turvalisust. Kõikides nendes teemades tutvustasime ka Spring Boot startereid ning nende automaatset seadistamist, mis on ühed suurimaid Spring Boot raamistiku eeliseid konkurentide ees.

Töö kolmandas peatükis tutvusime rakenduse haldamise võimalustega. Lähemalt vaatasime, kuidas jälgida töötavat rakendust kogu elutsükli vältel. Spring Boot pakub selleks mitmeid seire ja logimise funktsionaalsusi lihtsustavaid startereid. Lisaks tutvustasin ka rakenduse erinevaid paigaldamisvõimalusi füüsilistesse ja pilveserveritesse.

Kasutatud kirjandus

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). Introducing Spring Boot. Loetud aadressil http://docs.spring.io/autorepo/docs/spring-boot/1.4.1.RELEASE/reference/htmlsingle/#getting-started-introducing-spring-boot

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). System Requirements. Loetud aadressil http://docs.spring.io/autorepo/docs/spring-boot/1.4.1.RELEASE/reference/htmlsingle/#getting-started-system-requirements

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). Installing Spring Boot. Loetud aadressil http://docs.spring.io/autorepo/docs/spring-boot/1.4.1.RELEASE/reference/htmlsingle/#getting-started-installing-spring-boot

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). Using the @SpringBootApplication annotation. Loetud aadressil http://docs.spring.io/autorepo/docs/spring-boot/1.4.1.RELEASE/reference/htmlsingle/#using-boot-using-springbootapplication-annotation

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). Starters. Loetud aadressil http://docs.spring.io/autorepo/docs/spring-boot/1.4.1.RELEASE/reference/htmlsingle/#using-boot-starter

Pivotal Software, Inc. (kuupäev puudub). Spring Tool Suite. https://spring.io/tools/sts

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). Installing the Spring Boot CLI. Loetud aadressil https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#getting-started-installing-the-cli

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). Working with NoSQL technologies. Loetud aadressil http://docs.spring.io/autorepo/docs/spring-boot/1.4.1.RELEASE/reference/htmlsingle/#boot-features-nosql

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). Testing. Loetud aadressil http://docs.spring.io/autorepo/docs/spring-boot/1.4.1.RELEASE/reference/htmlsingle/#boot-features-testing

Wepp, P. (2016, 15. aprill). [ajaveebipostitus] Testing improvements in Spring Boot 1.4. Loetud aadressil https://spring.io/blog/2016/04/15/testing-improvements-in-spring-boot-1-4

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). Security. Loetud aadressil http://docs.spring.io/autorepo/docs/spring-boot/1.4.1.RELEASE/reference/htmlsingle/#boot-features-security

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). OAuth2. Loetud aadressil http://docs.spring.io/autorepo/docs/spring-boot/1.4.1.RELEASE/reference/htmlsingle/#boot-features-security-oauth2

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). Client. Loetud aadressil https://docs.spring.io/spring-boot/docs/1.4.1.RELEASE/reference/htmlsingle/#boot-features-security-custom-user-info-client

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). Authorization Server. Loetud aadressil https://docs.spring.io/spring-boot/docs/1.4.1.RELEASE/reference/htmlsingle/#boot-features-security-oauth2-authorization-server

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). Spring Boot Actuator: Production-ready features. Loetud aadressil https://docs.spring.io/spring-boot/docs/1.4.1.RELEASE/reference/htmlsingle/#production-ready

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). Logging. Loetud aadressil https://docs.spring.io/spring-boot/docs/1.4.1.RELEASE/reference/htmlsingle/#boot-features-logging

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. & Deleuze, S. (kuupäev puudub). Deploying Spring Boot applications. Loetud aadressil https://docs.spring.io/spring-boot/docs/1.4.1.RELEASE/reference/htmlsingle/#deployment

LISAD

Lisa 1

```
com
 +- example
     +- myproject
         +- Application.java
         |
         +- domain
         |   +- Customer.java
         |   +- CustomerRepository.java
         |
         +- service
         |   +- CustomerService.java
         |
         +- web
             +- CustomerController.java
```

Joonis 28. Koodi struktuur

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes=MyApp.class,
loader=SpringApplicationContextLoader.class)
public class MyTest {
    // ...
}
```

Joonis 29. @ContextConfiguration annotatsioon

```
@RunWith(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(MyApp.class)
public class MyTest {
    // ...
}
```

Joonis 30. @SpringApplicationConfiguration annotatsioon

```
@RunWith(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(MyApp.class)
@IntegrationTest
public class MyTest {
    // ...
}
```

Joonis 31. @IntegrationTest annotatsioon

```
@RunWith(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(MyApp.class)
@WebIntegrationTest
public class MyTest {
    // ...
}
```

Joonis 32. @WebIntegrationTest annotatsioon

```
@RunWith(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(classes=TestApplication.class)
@WebAppConfiguration
@IntegrationTest
public class MyTest {
    // ...
}
```

Joonis 33. @WebAppConfiguration ja @IntegrationTest annotatsioon