Spring-koulutus

Spring

Spring-perusteet

Spring

“Spring Framework” -projekti koostuu moduuleista, joita sovelluksissa voi käyttää yhdessä ja erikseen.

Spring voi kontekstista riippuen tarkoittaa joko pelkästää projektia “Spring Framework” tai sen päälle toteutettujen projektien ryhmää.

Springin ydin koostuu moduuleista, jotka toteuttavat olioiden (JavaBean) säilön, konfiguraatiomallin ja riippuvuuksien hallinnan

IoC

“Inversion of Control” tai “Dependency Injection” on malli, jossa olioiden riippuvuudet asetetaan ainoastaa rakentimen argumenteissa, “factory”-metodien argumenteissa tai olioiden ominaisuuksina (“settereissä”). Oliot eivät itse luo muita olioita eivätkä käytä ns. “Service Locator” -palveluita.

Säiliö

Olioita ja niiden riippuvuuksia hallitaan Spring-säiliössä (container). Säiliö toteuttaa rajapinnat *BeanFactory* ja *ApplicationContext*. Säiliön, sen sisältämät oliot ja niiden riippuvuudet voi konfiguroida XML-tiedostoilla, Groovylla, annotaatioilla tai Java-koodilla.

```
ApplicationContext context =
  new ClassPathXmlApplicationContext("SpringApplicationContext.xml");
```

Spring-esimerkit

Spring Bean -oliota kutsutaan alla yksinkertaisesti olioksi.

Esimerkki 1: spring-demo-all

Tehtävä 1.1: Avaa Maven-projekti spring-demo-all Eclipsessä ja suorita se. Tutki lyhyesti projektin rakennetta.

- Maven-projektin rakenne ja riippuvuudet
- Spring-säiliön luonti ja pääohjelma (*DemoApplication.java*)
- Spring-säiliön konfigurointi XML-tiedostolla (*SpringApplicationContext.xml*)
- Olioiden hakeminen säiliöstä
- Yksinkertaiset Spring-bean -oliot (*SimpleBean.java*)
- Olion alias
- Singleton-olio vs. prototyyppi
- Listan initialisointi säiliön konfiguraatiossa (*ListBean.java*)
- Sisäisen olion initialisointi konfiguraatiossa (*NestedBean.java*)
- Olion initialisointi erillisen metodin avulla (*InitMethodBean.java*)
- “Tehdas”-metodin käyttäminen olion luonnissa (*FactoryMethodBean.java*)
- “Tehdas”-olion käyttäminen olion luonnissa (*SimpleBeanFactory.java*)
- Riippuvuuksien asettaminen (*DependentBean.java*)
- Riippuvuuksien asettaminen automaatisesti nimien tai luokkien avulla (*AutowireBean.java*)

Tehtävä 1.2: Lisää säiliöön vapaasti uusia olioita erilaisilla tavoilla.

Esimerkki 2: spring-demo-resources

Tehtävä 2.1: Avaa Maven-projekti spring-demo-resources Eclipsessä ja suorita se. Tutki

lyhyesti projektin rakennetta.

- Resurssien käyttö Spring-säiliön kautta
- Resurssin määrittely XML-kofiguraatiossa
- Resurssin haku olion sisällä

Tehtävä 2.2: Lisää projektiin toinen resurssi ja hae se olion käyttöön.

Esimerkki 3: spring-demo-properties

Tehtävä 3.1: Avaa Maven-projekti spring-demo-properties Eclipsessä ja suorita se. Tutki lyhyesti projektin rakennetta.

- Asetusten määrittely properties-tiedostossa
- Properties-tiedoston käyttöönotto Spring-konfiguraatiossa
- Asetusten haku olioiden käyttöön (*SomeBean.java*)

Tehtävä 3.2: Lisää SomeBean-olioon uusi attribuutti, jonka asetat properties-tiedoston avulla.

Esimerkki 4: spring-demo-database

Tehtävä 4.1: Tee MySQL-tietokanta (tai MariaDB), jossa on taulu *users*. Taulussa on kolme saraketta, *username*, *firstname* ja *lastname*. Järjestä tietokantaan pääsy käyttäjälle *demo*, jonka salasana on “demopwd”. Avaa Maven-projekti spring-demo-database Eclipsessä ja suorita se. Tutki lyhyesti projektin rakennetta.

- DataSource-objektin määrittely Spring-säiliössä asetustiedoston avulla
- DataSource-olion käyttö (*DaoBean.java*)

- JdbcTemplate-luokan käyttö tietokantahaussa (*DaoBean.java*)

Tehtävä 4.2: Toteuta *DaoBean-luokkaan* uuden käyttäjärivin lisäys. Välitä parametrina *DemoUser*-olio tai sen attribuutin erikseen. Käytä metodia *JdbcTemplate.update*. Testaa lisäystä pääohjelmassa.

Esimerkki 5: spring-demo-messages

Tehtävä 5.1: Avaa Maven-projekti spring-demo-messages Eclipsessä ja suorita se. Tutki lyhyesti projektin rakennetta.

- Merkkijonojen haku properties-tiedostoista
- MessageSource-olion luonti ja käyttö (*SomeBean.java*)
- Lokalisaatio

Tehtävä 5.2: Lisää projektiin uusi properties-tiedosto *fields.properties* ja hae sieltä merkkijono pääohjelmassa.

Esimerkki 6: spring-demo-annotations

Tehtävä 6.1: Avaa Maven-projekti spring-demo-annotations Eclipsessä ja suorita se. Tutki lyhyesti projektin rakennetta.

- Annotaatioiden käyttö olioiden luonnissa ja riippuvuuksien määrittelyssä
- *@Repository*, *@Service*, *@Component*, *@Autowire*
- Spring-context -nimiavaruuden käyttäminen XML-konfiguraatiossa
- Olioiden “skannaus” projektin sisältä

Tehtävä 6.2: Lisää projektiin *@Component* -annotaatiolla varustettu luokka *ConsumerBean*, johon lisätään riippuvuus *ServiceBean*-olioon (“autowire”). Luokassa on yksi metodi, joka kutsuu *ServiceBean*-olion metodia *getData()*. Testaa pääohjelmassa.

Esimerkki 7: spring-demo-annotations-java

Tehtävä 7.1: Avaa Maven-projekti spring-demo-annotations-java Eclipsessä ja suorita se. Tutki lyhyesti projektin rakennetta.

- Spring-säiliön konfigurointi kokonaan ilman XML-tiedostoja (*AppConfig.java*)
- *@Configuration*, *@ComponenScan*

Esimerkki 8: spring-demo-database-annotation

Tehtävä 8.1: Avaa Maven-projekti spring-demo-database-annotation Eclipsessä ja suorita se. Tutki lyhyesti projektin rakennetta.

- Asetusten käyttö *@Value*-annotaatioilla (*ApplicationConfig.java*)
- Olion luonti “tehdas” -metodilla ja *@Bean*-annotaatiolla (*ApplicationConfig.java*)
- Asetusten käyttö ilman XML-konfiguraatiota (*ApplicationConfig2.java*)
- *@ImportResource*, *@PropertySource*

Spring-boot -perusteet

Spring-boot on yksinkertainen tapa luoda Spring-pohjaisia sovelluksia. Sovelluksia voi ajaa suoraan jar-tiedostoista

```
java -jar spring-boot-app.jar
```

tai esimerkiksi Mavenilla suoraan projektihakemistosta

```
mvn spring-boot:run
```

Spring-boot -projektin pohjan voi luoda helposti Spring Initializr -sivustolla

https://start.spring.io/

Spring-boot -projekteilla on tavallisesti “parent” -projektina *spring-boot-starter-parent*. Tiedostossa pom.xml on

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.5.9.RELEASE</version>
  <relativePath/>
</parent>
```

Spring-bootissa on määritelty ns. “Starter” -riippuvuuksia, joilla on helppo lisätä projektiin kokonaisuuksia, jotka yhdessä toteuttavat jonkin ominaisuuksia joukon. Tyypillinen esimerkki on spring-boot-starter-web.

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


```
  </dependency>
</dependencies>
```

Tämä “starter” lisää sovellukseen tuen web-sovellusten tekemiseen. Mukana on silloin mm. RESTfull, Spring MVC ja Tomcat.

Muita “startereita” löytyy esimerkiksi täältä

https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-build-systems.html#using-boot-starter

Tehtävä: Kokeile Spring Initializr -palvelua ja luo Spring-boot -projekti, jossa on starter-web -riippuvuus ja joitakin muita riippuvuuksia. Avaa projekti Eclipsessä tarkastele sen rakennetta.

Spring-boot -esimerkit

Esimerkki 9: spring-boot-demo-rest

Tehtävä 9.1: Avaa Maven-projekti spring-boot-demo-rest Eclipsessä ja suorita se. Tutki lyhyesti projektin rakennetta.

- Spring-boot -projektin rakenne ja riippuvuudet (*pom.xml*)
- *@SpringBootApplication* -annotaatio (*@Configuration, @EnableAutoConfiguration, @ComponentScan*)
- Annotaatioiden käyttö Spring-säiliön määrittelyssä
- Rest-palvelun toteutus, @RestController (*UserController.java*)
- *@GetMapping*, *@PutMapping*, *ResponseEntity*, …
- Asetusten määrittely (*application.properties*)
- Tietokannan käyttö (*UserDao.java*)

Tehtävä 9.2: Asenna selaimeen lisäosa (tai erillinen sovellus), jolla voit testata REST-rajapintoja. Kokeile rajapintoja

http://localhost:18080:/users
http://localhost:18080/user/user1

Tehtävä 9.3: Toteuta *UserController*-luokkaan metodi *deleteUser()*, jolla voi poistaa käyttäjän. Tarvitset annotaatiota *@DeleteMapping*. Lisäksi *UserDao*-luokkaan pitää lisätä vastaava metodi.

Esimerkki 10: spring-boot-demo-web

Tehtävä 10.1: Avaa Maven-projekti spring-boot-demo-web Eclipsessä. Tutki lyhyesti projektin rakennetta. Käynnistä projekti ja kokeile selaimella osoitetta

http://localhost:28080/users

Tehtävä 10.2: Käynnistä myös projekti spring-boot-demo-rest ja kokeile sen jälkeen osoitetta

http://localhost:28080/users

- Web-sovelluksen toteutus *(UserController.java*)
- REST-palvelun käyttö (*RestTemplate, UserClient.java*)
- Yksinkertainen poikkeuksien käsittely (*UserController.java*)
- Freemarker-kaavojen käyttö

Tehtävä 10.3: Lisää palveluun yksittäisen käyttäjän tietojen näyttäminen. Lisää UserController-luokkaan vastaava metodi. Lisää myös Freemarker-kaava tietojen näyttämiseen.