Spring boot and rest webservices

MOSTAFA RASTGAR

JUNE 2019

Agenda

- Spring Boot vs Spring MVC vs Spring
- Spring Initializer
- Spring Boot Auto Configuration
- Spring Boot Starters - Web and JPA
- Spring Boot Starter Parent
- Spring Boot Developer Tools and Live Reload
- Logging with Spring Boot - Logback, SLF4j and LOG4j2
- Introduction to Spring Data

Spring Boot vs Spring MVC vs Spring

- What is the core problem that Spring Framework solves?
  - Most important feature of Spring Framework is Dependency Injection. At the core of all Spring Modules is Dependency Injection or IOC Inversion of Control.

Example without Dependency Injection

```
@RestController
public class WelcomeController {

    private WelcomeService service = new WelcomeService();

        @RequestMapping("/welcome")
        public String welcome() {
                return service.retrieveWelcomeMessage();
        }
}
```

Same Example with Dependency Injection

```
@Component
public class WelcomeService {
    //Bla Bla Bla
}
 @RestController
public class WelcomeController {

    @Autowired
    private WelcomeService service;

        @RequestMapping("/welcome")
        public String welcome() {
                return service.retrieveWelcomeMessage();
        }
}
```

Spring Boot vs Spring MVC vs Spring

- What else does Spring Framework solve?
  - Does Spring Framework stop with Dependency Injection? No. It builds on the core concept of Dependency Injection with a number of Spring Modules
    - Spring JDBC
    - Spring MVC
    - Spring AOP
    - Spring ORM
    - Spring JMS
    - Spring Test
  - Good Integration with Other Frameworks
    - Hibernate for ORM
    - iBatis for Object Mapping
    - JUnit & Mockito for Unit Testing

Spring Boot vs Spring MVC vs Spring

- What is the core problem that Spring MVC Framework solves?

> *Spring MVC Framework provides decoupled way of developing web applications. With simple concepts like Dispatcher Servlet, ModelAndView and View Resolver, it makes it easy to develop web applications.*

- A conventional Spring MVC based project configuration:
  - component scan
  - dispatcher servlet
  - view resolver
  - web jars(for delivering static content) among other things
  - datasource, entity manager factory, transaction manager
  - ...

Spring Boot vs Spring MVC vs Spring

- Why do we need Spring Boot?

Problem #1 : Spring Boot Auto Configuration : Can we think different?

Can we bring more intelligence into this? When a spring mvc jar is added into an application, can we auto configure some beans automatically?

Spring Boot looks at a) Frameworks available on the CLASSPATH b) Existing configuration for the application. Based on these, Spring Boot provides basic configuration needed to configure the application with these frameworks. This is called `Auto Configuration` .

Problem #2 : Spring Boot Starter Projects : Built around well known patterns

Starters are a set of convenient dependency descriptors that you can include in your application. You get a one-stop-shop for all the Spring and related technology that you need, without having to hunt through sample code and copy paste loads of dependency descriptors. For example, if you want to get started using Spring and JPA for database access, just include the spring-boot-starter-data-jpa dependency in your project, and you are good to go.

Spring Boot vs Spring MVC vs Spring

- Spring Boot Starter Project Options
  - spring-boot-starter-web-services - SOAP Web Services
  - spring-boot-starter-web - Web & RESTful applications
  - spring-boot-starter-test - Unit testing and Integration Testing
  - spring-boot-starter-jdbc - Traditional JDBC
  - spring-boot-starter-hateoas - Add HATEOAS features to your services
  - spring-boot-starter-security - Authentication and Authorization using Spring Security
  - spring-boot-starter-data-jpa - Spring Data JPA with Hibernate
  - spring-boot-starter-cache - Enabling Spring Framework's caching support
  - spring-boot-starter-data-rest - Expose Simple REST Services using Spring Data REST
  - spring-boot-starter-actuator - To use advanced features like monitoring & tracing to your application out of the box
  - spring-boot-starter-undertow, spring-boot-starter-jetty, spring-boot-startertomcat - To pick your specific choice of Embedded Servlet Container
  - spring-boot-starter-logging - For Logging using logback
  - spring-boot-starter-log4j2 - Logging using Log4j2
- Spring Boot aims to enable production ready applications in quick time
  - Actuator : Enables Advanced Monitoring and Tracing of applications.
  - Embedded Server Integrations - Since server is integrated into the application, I would NOT need to have a separate application server installed on the server.
  - Default Error Handling

Spring Initializer

- Show Demo
- Run with jetty
- Run with undertow

```
<dependency>
	<groupId>org.springframework.boot</groupId>
	<artifactId>spring-boot-starter-web</artifactId>
	<exclusions>
		<!-- Exclude the Tomcat dependency -->
		<exclusion>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-tomcat</artifactId>
		</exclusion>
	</exclusions>
</dependency>
```

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

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

Spring Boot Auto Configuration

- Launch Spring Initializer and choose the following dependencies
  - Web
  - Actuator
  - DevTools
- Run the application and see the log

```
2019-06-03 08:13:49.103  INFO 8852 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized
with port(s): 8080 (http)
2019-06-03 08:13:49.227  INFO 8852 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service
[Tomcat]
2019-06-03 08:13:49.227  INFO 8852 --- [  restartedMain] org.apache.catalina.core.StandardEngine  : Starting Servlet
engine: [Apache Tomcat/9.0.19]
2019-06-03 08:13:49.426  INFO 8852 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring
embedded WebApplicationContext
2019-06-03 08:13:49.427  INFO 8852 --- [  restartedMain] o.s.web.context.ContextLoader            : Root
WebApplicationContext: initialization completed in 4104 ms
2019-06-03 08:13:52.124  INFO 8852 --- [  restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing
ExecutorService 'applicationTaskExecutor'
2019-06-03 08:13:53.268  INFO 8852 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is
running on port 35729
2019-06-03 08:13:53.302  INFO 8852 --- [  restartedMain] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 2 endpoint(s)
beneath base path '/actuator'
2019-06-03 08:13:53.513  INFO 8852 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on
port(s): 8080 (http) with context path ''
2019-06-03 08:13:53.524  INFO 8852 --- [  restartedMain] c.m.springboot.SpringbootApplication     : Started
SpringbootApplication in 8.861 seconds (JVM running for 10.864)
2019-06-03 08:13:54.122  INFO 8852 --- [on(4)-10.0.75.1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring
DispatcherServlet 'dispatcherServlet'
2019-06-03 08:13:54.123  INFO 8852 --- [on(4)-10.0.75.1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet
'dispatcherServlet'
```

Spring Boot Auto Configuration

- Where is Spring Boot Auto Configuration?
  - spring-bootautoconfigure.jar
  - /METAINF/spring.factories
  - Turn on debug logging to display auto-configured beans

```
============================
CONDITIONS EVALUATION REPORT
============================


Positive matches:
-----------------

   AuditAutoConfiguration#auditListener matched:
      - @ConditionalOnMissingBean (types: org.springframework.boot.actuate.audit.listener.AbstractAuditListener;
SearchStrategy: all) did not find any beans (OnBeanCondition)

   AuditAutoConfiguration.AuditEventRepositoryConfiguration matched:
      - @ConditionalOnMissingBean (types: org.springframework.boot.actuate.audit.AuditEventRepository; SearchStrategy:
all) did not find any beans (OnBeanCondition)

   AuditEventsEndpointAutoConfiguration matched:
      - @ConditionalOnEnabledEndpoint no property management.endpoint.auditevents.enabled found so using endpoint default
(OnEnabledEndpointCondition)
```

Spring Boot Starters - Web and JPA

- What if we do not have starter projects?

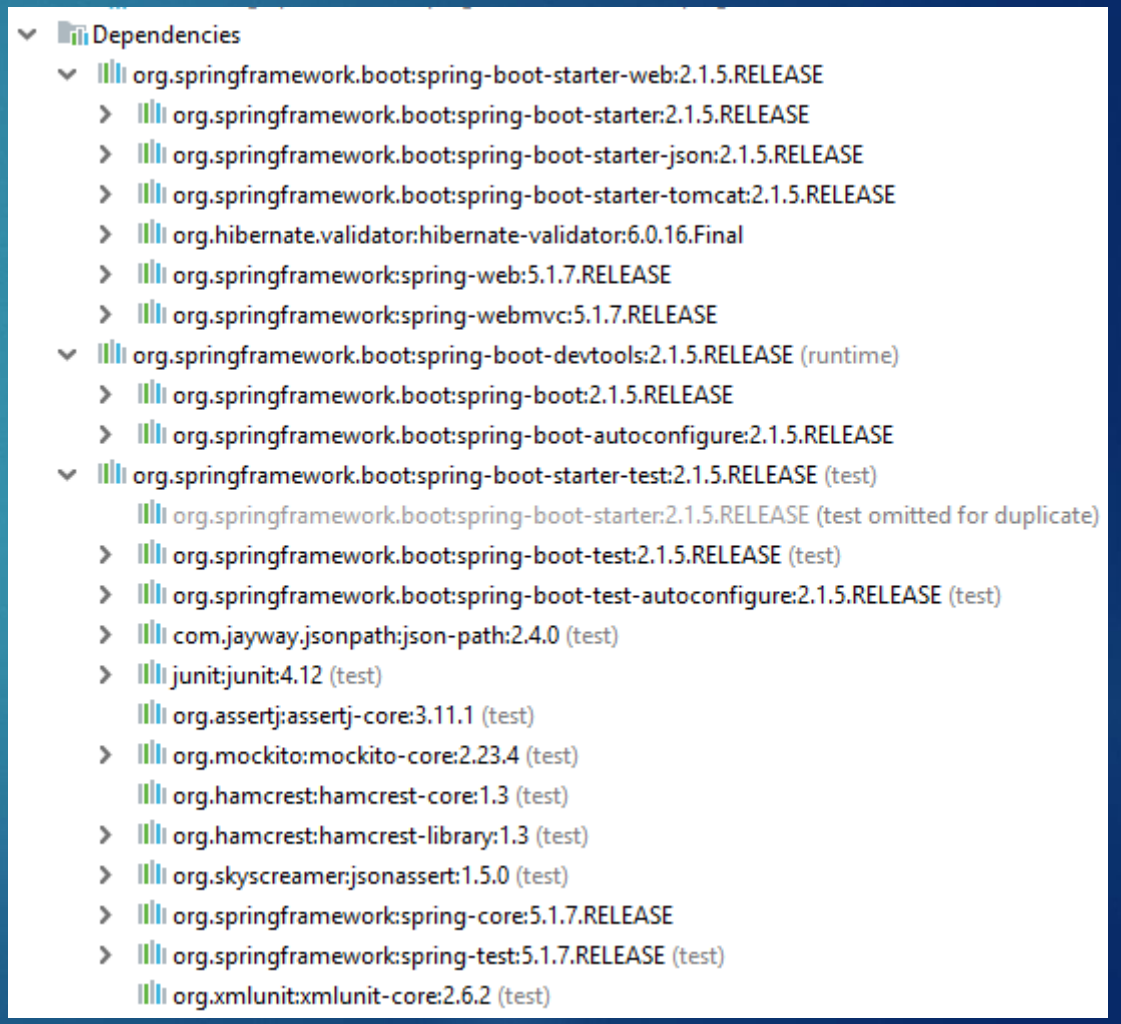
```
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-webmvc</artifactId>
<version>4.2.2.RELEASE</version>
</dependency>
<dependency>
<groupId>com.fasterxml.jackson.core</groupId>
<artifactId>jackson-databind</artifactId>
<version>2.5.3</version>
</dependency>
<dependency>
<groupId>org.hibernate</groupId>
<artifactId>hibernate-validator</artifactId>
<version>5.0.2.Final</version>
</dependency>
<dependency>
<groupId>log4j</groupId>
<artifactId>log4j</artifactId>
<version>1.2.17</version>
</dependency>
```

```
<bean
class="org.springframework.web.servlet.view.InternalResourceViewRes
olver">
<property name="prefix">
<value>/WEB-INF/views/</value>
</property>
<property name="suffix">
<value>.jsp</value>
</property>
</bean>
<bean id="messageSource"
class="org.springframework.context.support.ReloadableResourceBundl
eMessageSource">
<property name="basename" value="classpath:messages" />
<property name="defaultEncoding" value="UTF-8" />
</bean>
<mvc:resources mapping="/webjars/**" location="/webjars/"/>
<servlet>
<servlet-name>dispatcher</servlet-name>
<servlet-class>
org.springframework.web.servlet.DispatcherServlet
</servlet-class>
<init-param>
<param-name>contextConfigLocation</param-name>
<param-value>/WEB-INF/todo-servlet.xml</paramvalue>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>dispatcher</servlet-name>
<url-pattern>/</url-pattern>
</servlet-mapping>
```

```
<bean id="dataSource"
class="com.mchange.v2.c3p0.ComboPooledDataSource" destroy-
method="close">
<property name="driverClass" value="${db.driver}" />
<property name="jdbcUrl" value="${db.url}" />
<property name="user" value="${db.username}"/>
<property name="password" value="${db.password}" />
</bean>
<jdbc:initialize-database data-source="dataSource">
<jdbc:script location="classpath:config/schema.sql" />
<jdbc:script location="classpath:config/data.sql" />
</jdbc:initialize-database>
<bean
class="org.springframework.orm.jpa.LocalContainerEntityManagerFact
oryBean" id="entityManagerFactory">
<property name="persistenceUnitName" value="hsql_pu" />
<property name="dataSource" ref="dataSource" />
</bean>
<bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager">
<property name="entityManagerFactory" ref="entityManagerFactory"
/>
<property name="dataSource" ref="dataSource" />
</bean>
<tx:annotation-driven transactionmanager="transactionManager"/>
```

Spring Boot Starters - Web and JPA

- Spring Boot Starter Web
  - Compatible Dependencies that are needed to develop web applications
  - Auto Configuration
  - Dependencies can be classified into:
    - Spring - core, beans, context, aop
    - Web MVC - (Spring MVC)
    - Jackson - for JSON Binding
    - Validation - Hibernate Validator, Validation API
    - Embedded Servlet Container - Tomcat
    - Logging - logback, slf4j



Spring Boot Starter Parent

- What is Spring Boot Starter Parent?
  - All Spring Boot projects typically use spring-boot-starter-parent as the parent in pom.xml
  - Parent Poms allow you to manage the following things for multiple child projects and modules:
    - Configuration - Java Version and Other Properties
    - Dependency Management - Version of dependencies
    - Default Plugin Configuration

```
<parent>
	<groupId>org.springframework.boot</groupId>
	<artifactId>spring-boot-starter-parent</artifactId>
	<version>2.1.5.RELEASE</version>
	<relativePath/> <!-- lookup parent from repository -->
</parent>
```

Spring Boot Developer Tools and Live Reload

- Problem with Server Restarts
  - When we develop our applications (Web or RESTful API), we would want to be able to test our changes quickly
  - Typically, in the Java world, we need to restart the server to pick up the changes
- Adding Spring Boot Developer Tools to Your Project

```
<parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.5.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
</parent>
```

- Go ahead and make a simple change to your controller. You would see that these changes are automatically picked up(it is more eclipse friendly ☺)

Spring Boot Developer Tools and Live Reload

- What kind of changes does Spring Boot Developer Tools pick up?
  - By default, any entry on the classpath that points to a folder will be monitored for changes
  - These folders will not trigger reload by default
    - /META-INF/maven
    - /META-INF/resources
    - /resources
    - /static
    - /public
    - /templates
  - You can configure additional folders to scan
    - spring.devtools.restart.exclude=static/**,public/**

Logging with Spring Boot - Logback, SLF4j and LOG4j2

- Spring boot provides a default starter for logging - spring-boot-starter-logging
- It is included by default in spring-boot-starter which is included in all other starters

This means whenever you use any starters like `spring-boot-starter-web` or `spring-boot-starter-data-jpa`, you get logging for free!

- Let's look at what is present in the Logging Starter
- Configure Logging Levels
  - logging.level.some.package.path=DEBUG
  - logging.level.some.other.package.path=ERROR
  - logging.file=\path_to\logfile.log
- Logger logger = LoggerFactory.getLogger(this.getClass());
  - logger.trace("Your log - {}", value);
  - logger.debug("debug - {}", value);
  - logger.info("info- {}", value);
  - logger.warn("warn - {}", value);
  - logger.error("error - {}", value);

```
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-access</artifactId>
  <version>1.2.3</version>
</dependency>
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-to-slf4j</artifactId>
  <version>2.11.2</version>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>jul-to-slf4j</artifactId>
  <version>1.7.26</version>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>log4j-over-slf4j</artifactId>
  <version>1.7.26</version>
</dependency>
```

Logging with Spring Boot - Logback, SLF4j and LOG4j2

- Using Log4j2 for logging with Spring Boot
  - We would need to exclude the dependency on spring-boot-starterlogging and add a dependency on spring-boot-starter-log4j2
  - Custom configuration using log4j2.xml
  - You also have the option of using YAML or JSON with Log4j2
    - YAML - log4j2.yaml or log4j2.yml
    - JSON - log4j2.json or log4j2.jsn
  - However, you would need to include the appropriate dependency to handle yaml(jackson-dataformat-yaml) or json(jackson-databind)

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter</artifactId>
<exclusions>
<exclusion>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starterlogging</
artifactId>
</exclusion>
</exclusions>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-log4j2</artifactId>
</dependency>
```

Introduction to Spring Data

- What is Spring Data?
  - When Spring Framework was created, in early 2000s, the only kind of database was relational database - Oracle, MS SQL Server, My SQL etc
  - In the last few years, there are a wide variety of databases that are getting popular - most of them not relational and not using SQL. NoSQL, for example

> *Spring Data's mission is to provide a familiar and consistent, Spring-based programming model for data access while still retaining the special traits of the underlying data store. It makes it easy to use data access technologies, relational and non-relational databases, map-reduce frameworks, and cloud-based data services.*

  - To make it simpler, Spring Data provides Abstractions (interfaces) you can use irrespective of underlying data source

Introduction to Spring Data

- Crud Repository
  - public interface CrudRepository<T, ID> extends Repository<T, ID>
- PagingAndSortingRepository
  - Sort your data using Sort interface
  - Paginate your data using Pageable interface, which provides methods for pagination - getPageNumber(), getPageSize(), next(), previousOrFirst() etc.
- Defining custom queries
  - List<Person> findByFirstNameAndLastname(String firstName, String lastname);
  - Above method helps you search a data store by passing in the first name and last name of a person. This would generate the ppropriate query for the data store to return the person details
- Auditing with Spring Data

```
class Student {
@CreatedBy
private User createdUser;
@CreatedDate
private DateTime createdDate;
// … further properties omitted
}
```

Introduction to Spring Data

- Spring Data Implementations
  - Spring Data JPA - Connect to relational databases using ORM frameworks.
  - Spring Data MongoDB - Repositories for MongoDB.
  - Spring Data REST - Exposes HATEOAS RESTful resources around Spring Data repositories.
  - Spring Data Redis - Repositories for Redis.
- Spring Data JPA helps you to connect to relational databases using ORM frameworks.
  - public interface JpaRepository<T, ID> extends PagingAndSortingRepository<T, ID>, QueryByExampleExecutor<T>
- Exposing as a REST API
  - @RepositoryRestResource(collectionResourceRel = "rates", path = "rates")

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Introduction to Spring Data

- spring-boot-starter-data-rest

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-rest</artifactId>
</dependency>
```

  - Spring Data REST is using the HATEOAS (Hypermedia As The Engine Of Application State)
    - Each resource has its own URI
    - You always transfer the whole state of your object behind an endpoint
    - if you want to change it, manipulate it in your client and send it back to the server
  - supports HAL (Hypertext Application Language) as a semantic layer
    - As HATEOAS itself has no rules of how things like linking between resources, pagination, searches and other various meta data related task are handled, a few rivaling solutions exist, and HAL is for Spring Data REST HAL the winner

Introduction to Spring Data

- Explore in spring-boot-starter-data-rest with HAL Browser
  - http://localhost:8080/
    - __links_ is part of HAL and give an overview which links are available on exactly this endpoint
    - *profile* exposes additional meta data a potential client could use. We will ignore the profile and discover what *users* has to offer
  - Add a new rate
    - call exchangeValues with POST method
  - Modifying a rate
    - Call /X with PUT method to modify all fields
    - Call /X with PATCH method to modify specific fields
  - Detele a rate
    - Call /X with DETELTE method
  - /exchangeValues{?page,size,sort}
  - /search/findByFromAndTo{?from,to}

Introduction to Spring Data

- RepositoryDetectionStrategy
  - DEFAULT: all public repository interfaces but considers settings on annotations
  - ALL: all repositories independently of type visibility and annotations
  - ANNOTATION: only annotated repositories unless they are set to false
  - VISIBILITY: only public annotated repositories
- You can change the strategy by providing the RepositoryRestConfigurer in your @Configuration
- Annotations
  - *@RestResource* is used on model classes and define how to expose the model
  - *@RepositoryRestResource* is used on repositories and define how to expose the model