in28minutes / **jpa-with-hibernate**

Branch: **master** | Create new file | Find file | History

jpa-with-hibernate / 00-framework-tool-introductions / **02.Spring-Boot-Introduction-In-10-Steps** /

Ranga Rao Karanam and **Ranga Rao Karanam** I'm too lazy to put in a comment — Latest commit `a7fbfb0` on Mar 19, 2018

..		
.mvn/wrapper	first commit	3 years ago
src	first commit	3 years ago
.gitignore	first commit	3 years ago
code-21July2017.zip	first commit	3 years ago
notes.txt	I'm too lazy to put in a comment	3 years ago
pom.xml	I'm too lazy to put in a comment	2 years ago
readme.md	I'm too lazy to put in a comment	2 years ago

readme.md

First 10 Steps in Spring Boot

- Step 1 : Introduction to Spring Boot - Goals and Important Features
- Step 2 : Developing Spring Applications before Spring Boot
- Step 3 : Using Spring Initializr to create a Spring Boot Application
- Step 4 : Creating a Simple REST Controller
- Step 5 : What is Spring Boot Auto Configuration?
- Step 6 : Spring Boot vs Spring vs Spring MVC
- Step 7 : Spring Boot Starter Projects - Starter Web and Starter JPA
- Step 8 : Overview of different Spring Boot Starter Projects
- Step 9 : Spring Boot Actuator
- Step 10 : Spring Boot Developer Tools
- Spring Boot - Conclusion

in old spring we needed to decide each dependency and their versions
Version conflict happened! → Implement default exception handling

Complete Code Example

/notes.txt

microservices 1) build quickly

```
Goals
1) Enable building production ready applications quickly
2) Provide common non-functional features
- embedded servers
- metrics
- health checks
- externalized configuration

What Spring Boot is NOT!
ZERO code generation  no code generation
Neither an application server nor a web server
                      it provides integration with servers

Features
Quick Starter Projects with Auto Configuration
 - Web
 - JPA → Starter JPA → JPA → with hibernate and auto configuration → jar. Package along with application jar
Embedded Servers - Tomcat, Jetty or Undertow
```

```
Production-ready features
 - metrics and health checks
 - externalized configuration

http://localhost:8080/books => Few hardcoded books
```

→ Configuration differ by environment — Springboot supports it

/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.in28minutes.springboot.basics</groupId>
        <artifactId>springboot-in-10-steps</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>springboot-in-10-steps</name>
        <description>Demo project for Spring Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.0.0.RELEASE</version>
                <relativePath/> <!-- lookup parent from repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
        </properties>

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

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-actuator</artifactId>
                </dependency>
<dependency>
                        <groupId>org.springframework.data</groupId>
                        <artifactId>spring-data-rest-hal-browser</artifactId>
                </dependency>
<!--
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-jpa</artifactId>
                </dependency>
                 -->

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-test</artifactId>
                        <scope>test</scope>
                </dependency>

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

Rest service → web

Rest service

→ Prepared starter for web applications as well as - restful webservice

http-trace

If used in Production it will have an performance impact.

→ for monitoring application

→ means, how many services worked or failed

→ rest services are compliant with a standard called hal standard → will need hal browser → how open should work.

Kind of an interface

SOAP web

unit test

any java changes would be automatically picked up.

```
		</dependency>

	</dependencies>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

	<repositories>
		<repository>
			<id>spring-snapshots</id>
			<name>Spring Snapshots</name>
			<url>https://repo.spring.io/snapshot</url>
			<snapshots>
				<enabled>true</enabled>
			</snapshots>
		</repository>
		<repository>
			<id>spring-milestones</id>
			<name>Spring Milestones</name>
			<url>https://repo.spring.io/milestone</url>
			<snapshots>
				<enabled>false</enabled>
			</snapshots>
		</repository>
	</repositories>

	<pluginRepositories>
		<pluginRepository>
			<id>spring-snapshots</id>
			<name>Spring Snapshots</name>
			<url>https://repo.spring.io/snapshot</url>
			<snapshots>
				<enabled>true</enabled>
			</snapshots>
		</pluginRepository>
		<pluginRepository>
			<id>spring-milestones</id>
			<name>Spring Milestones</name>
			<url>https://repo.spring.io/milestone</url>
			<snapshots>
				<enabled>false</enabled>
			</snapshots>
		</pluginRepository>
	</pluginRepositories>


</project>
```

Springboot starters data rest
↓
expose entities from data JPA as restful services

/src/main/java/com/in28minutes/springboot/basics/springbootin10steps/Book.java

```
package com.in28minutes.springboot.basics.springbootin10steps;

public class Book {
	long id;
	String name;
	String author;

	public Book(long id, String name, String author) {
		super();
		this.id = id;
		this.name = name;
		this.author = author;
```

```
	}

	public long getId() {
		return id;
	}

	public String getName() {
		return name;
	}

	public String getAuthor() {
		return author;
	}

	@Override
	public String toString() {
		return String.format("Book [id=%s, name=%s, author=%s]", id, name, author);
	}

}
```

/src/main/java/com/in28minutes/springboot/basics/springbootin10steps/BooksController.java

```
package com.in28minutes.springboot.basics.springbootin10steps;

import java.util.Arrays;
import java.util.List;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class BooksController {
	@GetMapping("/books")
	public List<Book> getAllBooks() {
		return Arrays.asList(
				new Book(1l, "Mastering Spring 5.2", "Ranga Karanam"));
	}
}
```

Serve request (rest)
type of http method.
Get is to retrieve data
return a list of books
only doing this will return the data in json format

/src/main/java/com/in28minutes/springboot/basics/springbootin10steps/SpringbootIn10StepsApplication.java

```
package com.in28minutes.springboot.basics.springbootin10steps;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.ConfigurableApplicationContext;

@SpringBootApplication
public class SpringbootIn10StepsApplication {

	public static void main(String[] args) {
		ApplicationContext applicationContext =
			SpringApplication.run(SpringbootIn10StepsApplication.class, args);
		for (String name : applicationContext.getBeanDefinitionNames()) {
			System.out.println(name);
		}
	}
}
```

Indicates that it is a Spring Context file
enables auto-configuration
enables Component Scan → auto scans packages and subpackages for beans
returns application context
list of all the beans present in the application context.
auto configuration

/src/main/resources/application.properties

```
#logging.level.org.springframework = DEBUG
management.endpoints.web.exposure.include=*
```

→ Start application in debug mode, shows good information about autoconfiguration and what enabled it.

→ used by actuator for monitoring health.

→ to enable web exposure of all management endpoints

/src/test/java/com/in28minutes/springboot/basics/springbootin10steps/SpringbootIn10StepsApplicationTests.java

```
package com.in28minutes.springboot.basics.springbootin10steps;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
public class SpringbootIn10StepsApplicationTests {

	@Test
	public void contextLoads() {
	}

}
```

Loosely Coupled applications can be easily unit tested

Testability → Dependency injection → build loosely coupled applications

Inversion of Control

Prevents duplication eg: Jdbc

good integration

Simple way of developing web application

Seperation of concern

Monitoring → Springboot starter actuator