Theory: Basic project structure

5 minutes 0 / 3 problems solved

In this topic, you will consider the basic structure of any Spring Boot application. If you don't have such an application, just visit the Spring Initializr site and generate it with Gradle and Java.

§1. Gradle as a skeleton

The basic structure of a Spring Boot application depends on a build tool that we use for the project. Since we use Gradle, our project has the `build.gradle` file that describes how to build and manage the dependencies of the project.

```
.
├── build.gradle
├── gradle
│   └── ...
├── gradlew
├── gradlew.bat
├── HELP.md
├── settings.gradle
└── src
    ├── main
    │   ├── java
    │   │   └── org
    │   │       └── hyperskill
    │   │           └── demo
    │   │               └── DemoApplication.java
    │   └── resources
    │       └── application.properties
    └── test
        └── java
            └── org
                └── hyperskill
                    └── demo
                        └── DemoApplicationTests.java
```

There are also other Gradle-related files that you probably already know.

The initially generated Spring Boot application has several dependencies specified in the `build.gradle` file.

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}
```

The first dependency adds the Spring Boot framework to this project, and the second one brings test libraries integrated with Spring Boot. It is enough for the simplest Spring Boot application. As you can see, no boring configurations or excessive amount of dependencies!

The source code is placed in the `src` directory in `main` and `test` subdirectories.

Current topic:

Basic project structure

Topic depends on:

✕ External resources

✕ Dependency management

✕ Getting started with Spring Boot

Topic is required for:

Rest controller

Beans and components

H2 database

Introduction to JPA

Table of contents:

↑ Basic project structure

§1. Gradle as a skeleton

§2. The application properties

§3. The Application class

Feedback & Comments

§2. The application properties

Spring Boot uses **Convention Over Configuration** approach. It means that a developer only needs to specify unconventional aspects of the application, while all other aspects work by default.

To configure some aspects of a Spring Boot application, there is an `application.properties` file located in `src/main/resources`. In a newly generated project, this file is empty, but the application still works thanks to default implicit configurations.

We will modify this file in the next topics.

§3. The Application class

The entry point of our application is `DemoApplication` class located in `org.hyperskill.demo` package. This class contains the `main` method that is commonly-known among Java developers.

```
@SpringBootApplication
public class DemoApplication {

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

The presented program looks very simple, almost like a typical *hello-world* Java program. Inside `main`, the `SpringApplication.run()` method is invoked to launch the application with given arguments.

There is also an important annotation `@SpringBootApplication` that does all the Spring Boot magic! It makes your application work with incredible abilities: autoconfiguration, managing lifecycle of application components and a lot of other useful things that we will consider later. This annotation shows the convenient approach in Spring Boot: annotating classes and their members to get all features provided by Spring Boot.

That is all about the basic structure of Spring Boot applications. In the next topics, you will create new classes and set up configurations to develop an application that behaves as you like.

Report a typo

143 users liked this theory. **4** didn't like it. **What about you?**

Start practicing

Comments (0) Hints (0) Useful links (0) Show discussion