Spring Boot

Цель работы

- Обучиться приемам работы с Spring Boot

Дополнительные материалы

Практическая работа

Преобразования прошлого проекта Spring Framework в Spring Boot

1. Открываем предыдущий проект ToDO.
2. Для начала переходим в build.gradle и заменяем зависимость spring core и context на зависимость spring boot.

```
dependencies {  this: DependencyHandlerScope
    implementation( dependencyNotation: "org.springframework.boot:spring-boot-starter")
    implementation( dependencyNotation: "org.jetbrains.kotlin:kotlin-reflect")
    implementation( dependencyNotation: "org.jetbrains.kotlin:kotlin-stdlib-jdk8")
    testImplementation( dependencyNotation: "org.springframework.boot:spring-boot-starter-test")
}
```

3. Теперь на нужно включить механизм автоконфигурации Spring Boot. Для этого создадим класс. Например, класс ToDoApplication и включим аннотацию @SpringBootApplication.

ToDoApplication.kt

```kotlin
package com.example.todo

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication

@SpringBootApplication
class ToDoApplication

fun main(args: Array<String>) {
    runApplication<ToDoApplication>(*args)
}
```

4. Наше приложение должно запускаться с помощью CommandLineRunner. CommandLineRunner - это простой интерфейс загрузки Spring с методом запуска. Spring Boot автоматически вызовет метод запуска всех компонентов, реализующих этот интерфейс, после загрузки контекста приложения. Наш созданный класс должен реализовать интерфейс CommandLineRunner.

ToDoApplication.kt

```kotlin
package com.example.todo

import org.springframework.boot.CommandLineRunner
import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication

@SpringBootApplication
class ToDoApplication: CommandLineRunner {
    override fun run(vararg args: String?) {
        TODO( reason: "Not yet implemented")
    }
}

fun main(args: Array<String>) {
    runApplication<ToDoApplication>(*args)
}
```

5. Пробуем запустить консольное приложения. Для проверки можно создать logger для отображения информации. Logger — это некий объект, который отвечает за запись информации

в лог-файлы, опираясь на заданные уровни логирования. Основная задача логгера— не пропустить событие, которое нужно записать в лог-файл.

ToDoApplication.kt

```
5   import org.springframework.boot.CommandLineRunner
6   import org.springframework.boot.autoconfigure.SpringBootApplication
7   import org.springframework.boot.runApplication
8
9
10  @SpringBootApplication
11  class ToDoApplication: CommandLineRunner {
12
13      var logger: Logger = LoggerFactory.getLogger(ToDoApplication::class.java)
14
15      override fun run(vararg args: String?) {
16          logger.info("EXECUTING : command line runner");
17      }
18  }
19
20  fun main(args: Array<String>) {
21      runApplication<ToDoApplication>(*args)
22  }
```

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.6.3)

2022-01-26 12:58:35.644  INFO 5492 --- [           main] com.example.todo.ToDoApplicationKt       : Starting ToDoApplicationKt using Java 15.0.1 on LAPTOP-BQOL2I4T wit
2022-01-26 12:58:35.646  INFO 5492 --- [           main] com.example.todo.ToDoApplicationKt       : No active profile set, falling back to default profiles: default
2022-01-26 12:58:36.244  INFO 5492 --- [           main] com.example.todo.ToDoApplicationKt       : Started ToDoApplicationKt in 0.907 seconds (JVM running for 1.426)
2022-01-26 12:58:36.246  INFO 5492 --- [           main] com.example.todo.ToDoApplication         : EXECUTING : command line runner

Process finished with exit code 0
```

6. Дальше нужно перенести консольный обработчик в метод run и проверить работоспособность кода.

7. Теперь напишем тесты для консольного приложения с помощью Spring Boot. Для этого можно перейти из main в test и создать класс для тестирования, добавив к нему аннотацию @SpringBootTest. Аннотация автоматически выполняет поиск @SpringBootConfiguration. Поэтому мы можем указать в классе зависимость от тестируемого объекта с помощью аннотации @Autowired. Затем проводим тесты.

ToDoTest.kt

```kotlin
package com.example.todo

import org.junit.jupiter.api.Test
import org.junit.jupiter.api.Assertions.*
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.boot.test.context.SpringBootTest

@SpringBootTest
internal class ToDoTest {

    @Autowired
    lateinit var toDo: ToDo

    @Test
    fun add() {
        toDo.add(ToDoItem( description: "Убраться", Status.ACTIVE))
        assertEquals( expected: 1, toDo.size())
    }

    @Test
    fun delete() {
        toDo.add(ToDoItem( description: "Убраться", Status.ACTIVE))
        toDo.add(ToDoItem( description: "Погулять с собакой", Status.ACTIVE))
        toDo.delete( description: "Погулять с собакой")

        assertEquals( expected: 1, toDo.size())
    }

```

Практические задания

1. Покрыть тестами все методы вашего класса.

Теоретические вопросы

1. Что такое Spring Boot и как работает?
2. Какие есть основные различия между Spring и Spring Boot?
3. Какие еще бывают аннотация помимо @Component?
4. За что отвечает @EnableAutoConfiguration?