

# Spring Framework

## Цель работы

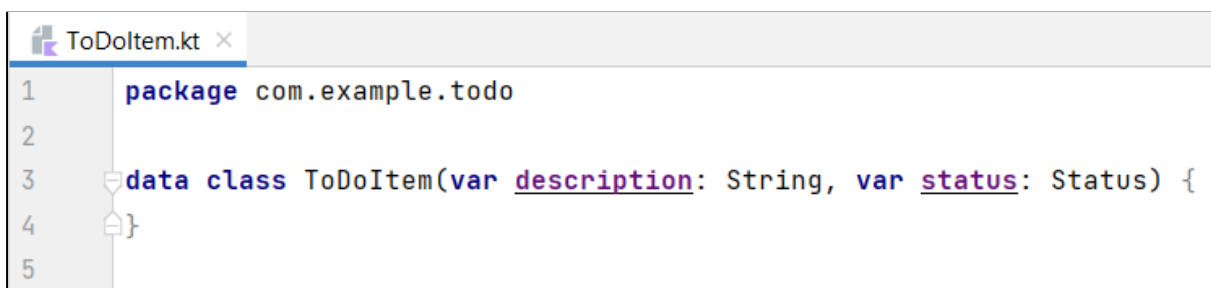
- Обучиться приемам работы с Spring Framework

## Дополнительные материалы

## Практическая работа

### Создание ToDo приложения

1. Открываем IntelliJ Idea.
2. Создаем gradle проект. Выбираем Kotlin DSL и Kotlin/JVM.
3. Переходим в build.gradle и создаем зависимость spring core (org.springframework:spring-core:5.3.15) и spring context (org.springframework:spring-context:5.3.15).
4. Создаем объект ToDo. Задаем ему свойства в виде описания и статуса .



```
1 package com.example.todo
2
3 data class ToDoItem(var description: String, var status: Status) {
4 }
5
```

5. Теперь для статуса создадим класс перечисления(enum).

```
ToDoItem.kt x
1 package com.example.todo
2
3 data class ToDoItem(var description: String, var status: Status) {
4 }
5
6 enum class Status{
7     DONE, ACTIVE
8 }
9
```

6. Теперь создадим сам список дел, который будет состоять из объектов, созданных ранее.

```
ToDoItem.kt x ToDo.kt x
1 package com.example.todo
2
3 class ToDo(private var todoList: MutableList<ToDoItem> = mutableListOf()) {...}
```

7. В этом классе вся работа будет происходить со списком, который содержит объект ToDoItem.

8. Методы данного класса:

a. Добавление

```
fun add(item: ToDoItem) = todoList.add(item)
```

b. Удаление. Разделяется на несколько видов: удаление по описанию, удалению по статусу, полное удаление.

```
fun delete(description: String): Boolean = todoList.removeIf { it.description.equals(description)}

fun deleteActive(): Boolean = todoList.removeIf { it.status.equals(Status.ACTIVE) }

fun deleteDone(): Boolean = todoList.removeIf { it.status.equals(Status.DONE) }

fun deleteAll() = todoList.clear()
```

c. Изменение. Изменение по описанию и по статусу.

```

fun changeDescription(descriptionOld: String, descriptionNew: String, statusNew: Status): Boolean {
    var item = find(descriptionOld)

    if (item != null && descriptionNew != null) {
        toDoList.set(toDoList.indexOf(item), item.apply { this.description = descriptionNew })
        return true
    }
    else
        return false
}

```

```

fun changeStatus(description: String, status: Status): Boolean {
    var item = find(description)

    if (item != null) {
        toDoList.set(toDoList.indexOf(item), item.apply { this.status = status })
        return true
    }
    else
        return false
}

```

d. Сортировка списка по виду: только активные, только выполненные, все.

```

fun listToDo(status: Status? = null): List<ToDoItem> = when(status) {
    Status.ACTIVE -> toDoList.filter { it.status.equals(Status.ACTIVE) }
    Status.DONE -> toDoList.filter { it.status.equals(Status.DONE) }
    null -> toDoList
}

```

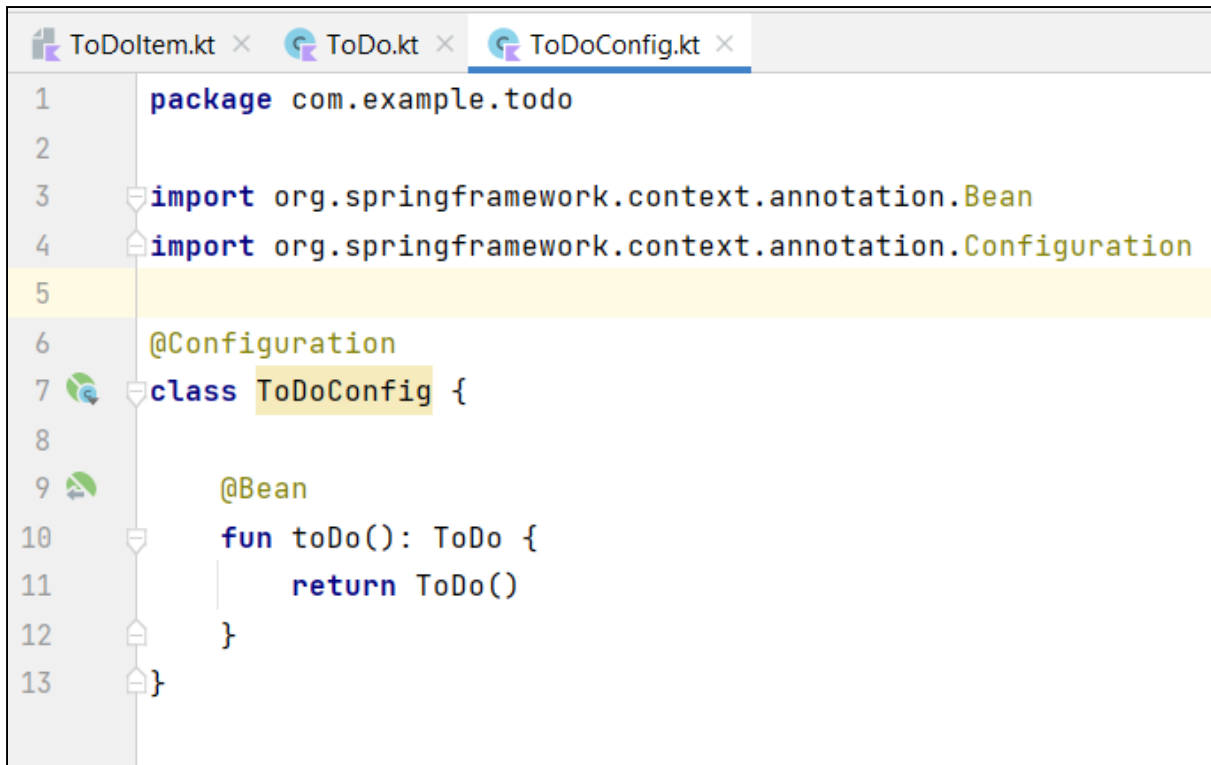
е. Поиск элемента по описанию.

```

private fun find(description: String): ToDoItem? = toDoList.find { it.description.equals(description) }

```

9. Создаем класс конфигурации ToDoConfig. Затем внутри класса создаем bean с классом ToDo.



```
1 package com.example.todo
2
3 import org.springframework.context.annotation.Bean
4 import org.springframework.context.annotation.Configuration
5
6 @Configuration
7 class ToDoConfig {
8
9     @Bean
10    fun todo(): Todo {
11        return Todo()
12    }
13 }
```

10. Создаем файл kotlin и в нем создаем main.



```
Doltem.kt x Main.kt x ToDo.
import ...
fun main() {
```

11. Для работы с контейнером Spring понадобится класс AnnotationConfigApplicationContext. В качестве аргумента передает созданную конфигурацию.

```
val ctx: ApplicationContext =
    AnnotationConfigApplicationContext(ToDoConfig::class.java)
```

12. Далее мы получаем bean с помощью метода getBean. Затем добавляем “дела” и статусы, пробуем удалить и проверяем, что получилось. Метод listOutPut выводит полученный. Способ выводы списка может быть любым.

```

var todo: ToDo = ctx.getBean(ToDo::class.java)

todo.add(ToDoItem( description: "Помыть посуду", Status.ACTIVE))
todo.add(ToDoItem( description: "Убраться", Status.ACTIVE))
todo.add(ToDoItem( description: "Погулять с собакой", Status.DONE))
todo.add(ToDoItem( description: "Отдохнуть", Status.ACTIVE))

todo.listOutPut(todo.listToDo())

todo.deleteDone()

todo.listOutPut(todo.listToDo())

```

```

0)  Описание: Помыть посуду      Статус: ACTIVE
1)  Описание: Убраться          Статус: ACTIVE
2)  Описание: Погулять с собакой  Статус: DONE
3)  Описание: Отдохнуть         Статус: ACTIVE

0)  Описание: Помыть посуду      Статус: ACTIVE
1)  Описание: Убраться          Статус: ACTIVE
2)  Описание: Отдохнуть         Статус: ACTIVE

```

## Практические задания

### Расширение приложения ToDo.

1. Нужно сделать двухуровневые ToDo, которые привязаны к спискам, либо с прикрепленными файлами.
2. Также нужно добавить дополнительные свойства к объектам (дата, дополнительные сведения и т.д.).
3. Еще нужно сделать меню по управлению списком дел.

## Теоретические вопросы

1. Что такое фреймворк?
2. Плюсы и минусы фреймворков?
3. Что такое Spring Framework?
4. Что такое DI?
5. Дать краткое описание аннотациям @Bean, @Configuration, @Component, @Scope.
6. Что такое зависимость?
7. Кратко рассказать о модуле Spring MVC.
8. Кратко рассказать о модуле Spring AOP.
9. Кратко рассказать о модуле Spring Data JPA.
10. Что такое @Transactional?
11. Что такое Spring Security?