

TODO App Backend

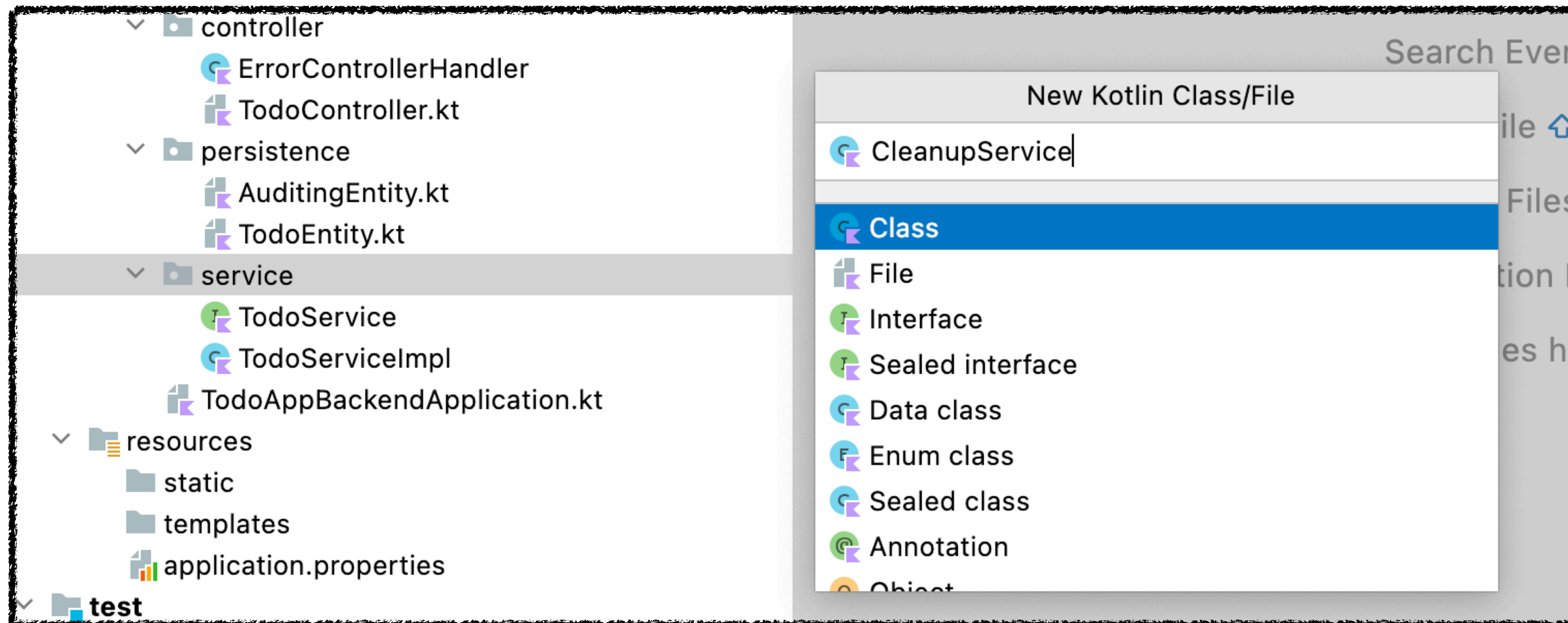
Schedulers

Exercise 9

- In this tutorial, we'll illustrate how the Spring @Scheduled annotation can be used to configure and schedule tasks.
- The simple rules that we need to follow to annotate a method with @Scheduled are:
 - the method should typically have a void return type (if not, the returned value will be ignored)
 - the method should not expect any parameters

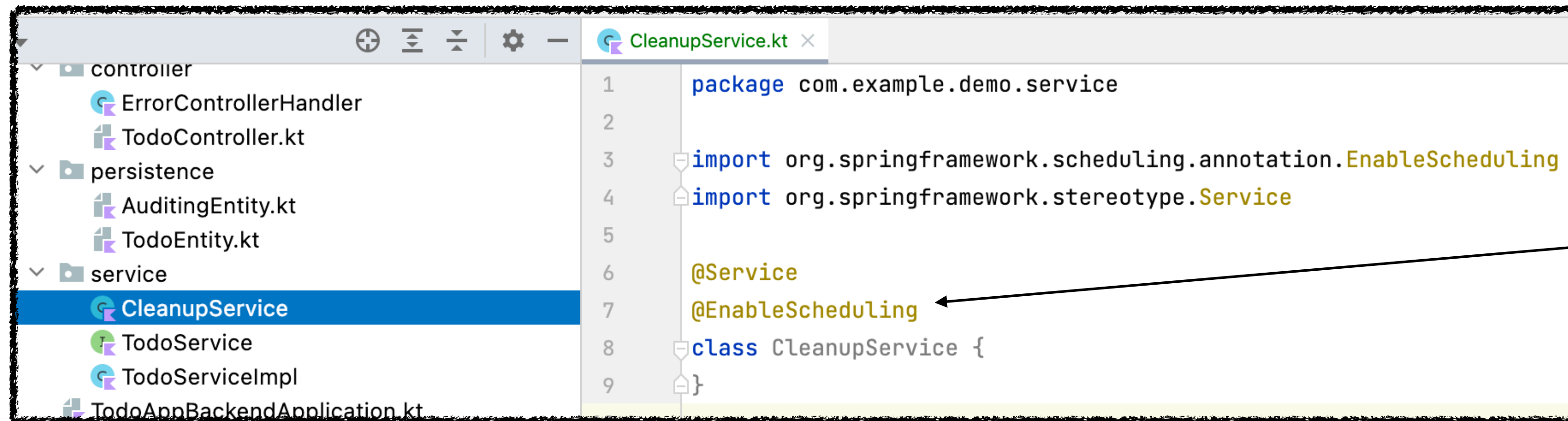
Create a new Service

Exercise 9



Enable Support for Scheduling

Exercise 9



```
1 package com.example.demo.service
2
3 import org.springframework.scheduling.annotation.EnableScheduling
4 import org.springframework.stereotype.Service
5
6 @Service
7 @EnableScheduling
8 class CleanupService {
9 }
```

To enable scheduling in Spring you can only need to add `@EnableScheduling` to a class that will be picked up during spring's component scan on application startup. Or in other words on any bean (like a `@Service` or `@Configuration` annotated class).

Schedule a Task at Fixed Delay

Exercise 9

```
@Service
@EnableScheduling
class CleanupService {

    @Scheduled(fixedDelay = 10_000)
    fun scheduleFixedDelayTask() {
        println("fixed scheduler executed at: ${LocalDateTime.now()}")
    }
}
```

Let's start by configuring a task to run after a fixed delay of 10 seconds

In this case, the duration between the end of the last execution and the start of the next execution is fixed. The task always waits until the previous one is finished.

This option should be used when it's mandatory that the previous execution is completed before running again.

See it printing every 10 seconds after app has been started

```
TodoAppBackendApplicationKt x
fixed scheduler executed at: 2022-04-20T22:14:20.365499
2022-04-20 22:14:20.365 INFO 12890 --- [main] c.e.demo.TODOAppBackendApplicationKt
fixed scheduler executed at: 2022-04-20T22:14:30.365413
fixed scheduler executed at: 2022-04-20T22:14:40.366344
fixed scheduler executed at: 2022-04-20T22:14:50.366931
fixed scheduler executed at: 2022-04-20T22:15:00.371838
fixed scheduler executed at: 2022-04-20T22:15:10.374019
fixed scheduler executed at: 2022-04-20T22:15:20.378096
fixed scheduler executed at: 2022-04-20T22:15:30.380428
fixed scheduler executed at: 2022-04-20T22:15:40.381906
fixed scheduler executed at: 2022-04-20T22:15:50.384163
```


Schedule a Task Using Cron Expressions

Exercise 9

Sometimes delays are not enough, and we need the flexibility of a cron expression to control the schedule of our tasks.

```
const val EVERY_NIGHT_AT_3 = "0 0 3 * * *"

@Service
@EnableScheduling
class CleanupService {

    @Scheduled(cron = EVERY_NIGHT_AT_3, zone = "Europe/Berlin")
    fun removeCompletedTasks() {

    }

    @Scheduled(fixedDelay = 10_000)
    fun scheduleFixedDelayTask() {
```

We define a so called cron expression here

By default, Spring will use the server's local time zone for the cron expression.

However, we can use the zone attribute to change this timezone

With this configuration, Spring will schedule the annotated method to run every night at 03:00 AM in Berlin time.

To Generate and understand cron expression syntax you can have a look at <https://www.freeformatter.com/cron-expression-generator-quartz.html>

Implement scheduler

Exercise 9

```
const val EVERY_NIGHT_AT_3 = "0 0 3 * * *"
```

```
@Service
```

```
@EnableScheduling
```

```
class CleanupService{
```

```
    private val todoService: TodoService
```

```
} {
```

```
    @Scheduled(cron = EVERY_NIGHT_AT_3, zone = "Europe/Berlin")
```

```
    fun removeCompletedTasks() {
```

```
        todoService.removeAllCompleted()
```

```
    }
```

```
    @Scheduled(fixedDelay = 10_000)
```

Inject todoService

Use todoService