# Creating a Controller

We'll cover the following



## Controllers

One of the core capabilities of Spring Boot is to auto-discover application components based on what it finds in the classpath. Among those components are controllers, which provide the routes for a web application. In the examples in this chapter, we'll use this facility.

> **Use Kofu to Explicitly Configure Applications**
>
> If you'd like to explicitly configure the application, refer to the Spring-fua initiative and the Kofu DSLb that provide a fluent syntax to configure various properties. In addition to benefiting from a fluent Kotlin API, Kofu-based applications benefit from a significantly less startup time compared to auto-configured applications.
>
> a. https://github.com/spring-projects/spring-fu
>
> b. https://github.com/spring-projects/spring-fu/blob/master/kofu/README.adoc

## Creating routes

Our controller needs to support three different routes/operations. Let's start with one route. The first route we'll create has an endpoint `task`, supports the GET HTTP method, and simply returns a message "to be implemented" as a starting

HTTP method, and simply returns a message "to be implemented" as a starting point.

## Kotlin vs. Java #

We'll name our controller `TaskController`. If we were writing this application in Java, our controller would look like this:

```
//Java code only for comparison purpose
package com.agiledeveloper.todo;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.http.ResponseEntity;

@RestController
@RequestMapping("/task")
class TaskController {
    @GetMapping
    String tasks() {
        return "to be implemented";
    }
}
```

Since we're using Kotlin, we can hope for less clutter than that. Create a file named `TaskController.kt` under the directory `todo/src/main/kotlin/com/agiledeveloper/todo/` and add the following code to that file:

```
package com.agiledeveloper.todo

import org.springframework.web.bind.annotation.*
import org.springframework.http.ResponseEntity

@RestController
@RequestMapping("/task")
class TaskController {
    @GetMapping
    fun tasks() = "to be implemented"
}
```

TaskController.kt

The `TaskController` class belongs to the `com.agiledeveloper.todo` package. The class is first annotated with the Spring Boot annotation `@RestController`, to declare that this class will serve as a controller and will support REST endpoints. The `@RequestMapping` annotation defines the endpoint supported by this controller. The `tasks()` method of the `TaskController` class is annotated with the `@GetMapping` annotation, and it returns a sample string.

## Running the application #

We're ready to build the code and exercise this controller's route. Using the commands we saw in Creating a Starter Project, build the project and execute the java command to run the program.

We'll use curl to verify that the endpoint works as expected. If you don't have curl already, you may download it for your operating system. Alternatively, you may use other tools like Advanced REST Client extension for Chrome, for example, that you typically use to interact with REST APIs.

Here's the curl command:

```
curl -w "\n" http://localhost:8080/task
```

This command sends a GET request to the mentioned URL, and the response should be the string returned by the `tasks()` method of `TaskController`.

These steps show that the controller we wrote using Kotlin works. That's a good first step.

---

In the next lesson, let's create an entity class to represent persistent data.