COURSEWARE

Beans

Contents

Overview

In Spring, a **bean** is a *managed object*; this means that the creation, management and destruction of that object is purely under the control of the Spring framework.

The practice of relinquishing control of objects to a framework in this way is known as **inversion of control**, which is why Spring is referred to as an *inversion of control container*.

Giving control up to the framework simplifies the life of a developer - they no longer need to keep track of, and manage, all of the objects required for an enterprise application (and can instead focus on writing functionality).

Spring helpfully puts all of its beans into one centralised location - the `ApplicationContext`.

Application Context

When beans are created they are saved to an instance of `ApplicationContext`; you can access this instance from the `main()` method of any Spring project:

```
@SpringBootApplication
public class SpringExampleApplication {

    public static void main(String[] args) {
        ApplicationContext context =
SpringApplication.run(SpringExampleApplication.class, args);
    }

}
```

Tutorial

To create a bean, we'll use the `@Bean` annotation.

note: ideally, beans should be defined in a class dedicated to that purpose, `AppConfig` for example, but this will suffice for now.

```
@SpringBootApplication
public class SpringExampleApplication {

    public static void main(String[] args) {
        ApplicationContext context =
SpringApplication.run(SpringExampleApplication.class, args);
    }

    @Bean
    public String greeting() {
        return "Hello, World";
    }
}
```

Once defined by the `greeting()` method, the bean is instantiated when the Spring app starts up and is then added into the `ApplicationContext`.

If we want to access the bean from the `ApplicationContext`, we will need some combination of its name and its type.

The name of the bean is (unless otherwise specified) the same as the name of the method that defined it (in **camelCase**) - so, in this case, the `greeting()` method would produce a bean with the name `greeting`:

```
public static void main(String[] args) {

    ApplicationContext context =
SpringApplication.run(SpringExampleApplication.class, args);

    Object byName = context.getBean("greeting");
    String byType = context.getBean(String.class);
    String byBoth = context.getBean("greeting", String.class);

    System.out.println(byName);
    System.out.println(byType);
    System.out.println(byBoth);

}
```

Exercises

1. Create a new Spring Boot application.
2. Create a new `@Bean` method in `AppConfig` that uses `LocalTime.now()` to return the current time.
3. Print this bean out to the console by accessing it in three different ways:
    - by name
    - by type
    - by name and type

We will be returning to this code later, so please ensure that it is pushed to your GitHub account.