Configuring Spring in Stand-Alone Apps

Here's a quick lesson in bringing Spring Configuration classes and functionality to your own stand-alone apps in the event you need them.

by Tomasz J-ski · Apr. 04, 18 · Java Zone · Tutorial

Spring is a powerful framework — and not only for dependency injection. It can strongly benefit applications as a whole. Sometimes, you need to create your own highly customized stand-alone application instead using an out-of-the-box Spring Boot solution.

Typical cases are custom utility applications used for one-time jobs like email scraping for needed information or custom database migration.

In those cases, you could use Spring Batch, but sometimes, if it is a one-time job, it is easier to just write own code.

You do not need to resign from using Spring in that case — just add it as a stand-alone context and use your favorite Spring features.

Context Loading in Stand-Alone Apps: The Most Critical Feature

AnnotationConfigApplicationContext is the most important class for loading Spring beans in stand-alone applications. It allows annotated classes as inputs, including component package scans and `@Configutration`-marked classes. In fact, this is the core of every stand-alone Spring application.

Typically, using AnnotationConfigApplicationContext looks like:

```
1 public static void main(String[] args) {
2                 AnnotationConfigApplicationContext ctx = new AnnotationConfigApplic
3                 ctx.register(AppConfig.class);
4                 ctx.refresh();
5 }
```

Where `AppConfig.class` is a separate `@Configuration` class with all needed beans.

In less complicated cases application main class could be also configuration, then code look like following

code look like following

```
@ComponentScan(basePackages = "YOUR_COMPONENTS_PACKAGE")
@Configuration
public class ApplicationX {

    private static final Logger LOGGER = LoggerFactory.getLogger(ApplicationX .class
    private SomeService someService;

    @Autowired
    public ApplicationX(SomeService someService) {
        this. someService = someService;
    }

    public static void main(String[] args) {
        ApplicationContext context = new AnnotationConfigApplicationContext(Applica

        ApplicationX appx = context.getBean(ApplicationX .class);
        appx.start();
    }

    private void start() {
        LOGGER.info("ApplicationX started.");
        // do stuff
    }
}
```

The constructor of the `AnnotationConfigApplicationContext` class takes multiple arguments of the `@Configuration` class. That means all configuration could be passed at once during the creation of the context.

That functionality enables adding separate configurations for separate components like databases or custom security features.

like databases or custom security features.

Configuration Classes: What They Are

All classes annotated by `@Configuration` in Spring are considered as configuration classes, but what does this really mean?

According to the Javadoc, the annotation `@Configuraion` indicates that a class declares one or more `@Bean` methods and may be processed by the Spring container to generate bean definitions and service requests for those beans at runtime.

So it is like a factory class to produce some beans declared in it.

The simplest example of a `@Configuration` class looks like the following:

```
@Configuration
public class ConnectionConfig {

    @Bean
    public MyConnectionService myConnection() {
        // some stuff
    }
}
```

Summary

Adding a Spring context to a stand-alone application is not difficult, and having it can significantly benefit applications. There is nothing wrong with using your own customized applications for one-time jobs, and Spring has special features to help

customized applications for one-time jobs, and Spring has special features to help with dependency injection in stand-alone applications.

By the techniques presented in this article, a developer can add their own Spring beans and inject them where needed — or add whole components like Spring Data to a stand-alone application.

Useful resources

- https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/Configuration.html
- https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/AnnotationConfigApplicationContext.html

Topics: BEAN, CONFIGURATION AS CODE, CONTEXT, JAVA, SPRING, STANDALONE CODE, TUTORIAL

Java Partner Resources