06/04/2022, 10:59 QA Community

## **COURSEWARE**

# **Professional Skills** Agile Fundamentals Jira Git **Databases Introduction** Java Beginner Maven Testing (Foundation) Java Intermediate HTML **CSS** Javascript Spring Boot Introduction to Spring Boot Multi-Tier Architecture Beans Bean Scopes Bean Validation Dependency Injection Components Configuration Connecting to a Database **Entities** Postman Controllers 0 Services Repositories **Custom Queries** Data Transfer Objects Lombok **Custom Exceptions** Swagger **Profiles**

## Components

#### Contents

- Overview
  - <u>@Component</u>
  - <a>@Configuration</a>
  - @Repository
  - <u>@Service</u>
  - Controller
    - RestController
- <u>@ComponentScan</u>
- <u>Tutorial</u>
- Exercises

## Overview

A Spring Component is a special type of bean, rather than creating a definition and annotating it with @Bean we instead declare an entire class as a bean. Components are single-use or singleton beans, which is why the annotation goes on the class - because we're only ever going to have to create one once.

## @Component

The most basic type of component, annotating a class with @Component simply instructs Spring to create a bean of that class at runtime.

For more *specific* usage Spring provides several annotations that use @Component as a base annotation - these have more *specialised* functionality.

## @Configuration

Configuration classes are used to create beans, conventionally called AppConfig, if you want to have a bean dependent on another bean you *must* define it inside of a configuration class.

#### @Repository

Used on data-access interfaces, @Repository allows for the conversion of common data-access exceptions into a single, easy to handle DataAccessException.

## @Service

Service, unlike the other types of component, offers no special functionality over @Component and is instead merely used to further show the *intent* of the class.

## Controller

Allows for the implementation of Spring Web endpoints using @RequestMapping.

#### RestController

A controller specifically for creating REST endpoints - applies @ResponseBody to each endpoint which causes them to send responses in JSON format.

## @ComponentScan

@ComponentScan is responsible for telling Spring where to look for components.

This annotation is part of <code>@SpringBootApplication</code> which can be found on the main class of any Spring Boot application.

Unit testing with Mockito

Pre-Populating Databases for Testing

Testing Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB **Express NodeJS** React **Express-Testing** Networking Security Cloud Fundamentals **AWS Foundations AWS Intermediate** Linux DevOps Jenkins Introduction Jenkins Pipeline Markdown

**IDE Cheatsheet** 

By default, Spring will search within the package that the main class is located, along with all of its child packages.

As such, it is *very important* to only put components in the *same package* or a *child package*:

```
└─ src
 — main
     — java
       L_ com
           └─ qa
               └─ demo
                   ├─ config
                      └─ AppConfig.java
                      - persistence
                       — domain
                           └─ Entity.java
                       └─ repos
                           └─ ExampleRepo.java
                       └─ ExampleController.java
                     - services
                       └─ ExampleService.java

SpringExampleApplication.java

     - resources
       — application.properties
         — static
       └─ templates
  - test
   └─ java
       L— com
           └─ qa
               └─ demo
                   SpringExampleApplicationTests.java
```

In this example, SpringExampleApplication (the main class) is *higher up* than all of the other classes (in com.qa.demo).

This means the **component scan** will check *every* single class in src/main/java.

It's also worth noting that the package structure in src/test/java is the same as in src/main/java.

When running tests that require accessing the ApplicationContext, Spring will discover them automatically - provided that the packages match.

## **Tutorial**

There is no tutorial for this module.

## **Exercises**

There are no exercises for this module.