Java Spring & AWS
Sep 7, 2021 - Oct 8, 2021
Monday to Friday
9:30 AM ET - 4:30 PM ET

Authorized & published by Summitworks Technologies Inc

Agenda

- **Spring DI**

When the application is being loaded, the Spring IoC (Inversion of Control) container scans the classes and if Spring-annotated classes are found, it creates instances of these classes and wires them together according to the annotations used - hence dependency injection is made

Spring DI

Dependency injection is a technique that allows the client code to be independent from the services it is relying on. The client does not control how objects of the services are created - it works with an implementation of the service through interface. This is somewhat in inverse to trivial programming so dependency injection is also called ***inversion of control***.
It makes the code more flexible, extensible, maintainable, testable and reusable - thus dependency injection is very popular in modern programming.

The first annotation you can use to mark a class as a managed bean in Spring is the @Component annotation. For example:

```
import org.springframework.stereotype.Component;

@Component("client1")
public class MyClientImpl implements MyClient {
...
}
```

Here, the class MyClientImpl is marked with the @Component annotation so Spring will create an instance of this class and manage it as a bean with the name client1 in the container.
To tell Spring to inject an instance of another class into this class, declare an instance field with the @Autowired annotation

Spring DI

```
import org.springframework.stereotype.Component;
import org.springframework.beans.factory.annotation.Autowired;

@Component("client1")
public class MyClientImpl implements MyClient {

   @Autowired
   private MyService service;

   ...
}
```

Here, Spring will find an instance of MyServicewith the name serviceand inject into the instance client1of MyClientImp

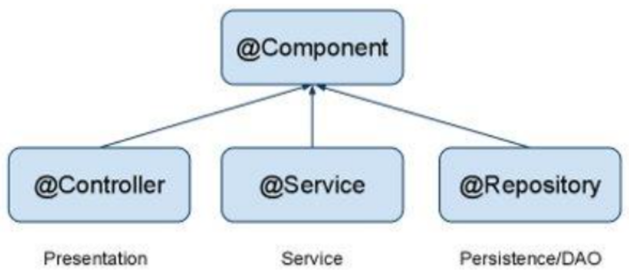
- There cannot have two beans in the IoC container with the same name, so the name you specify in the @Component must be unique.
- You can also mark a class with @Service and @Repository annotations. These annotations have same technical purpose as the @Component annotation. They have different names to mark classes in different layers of the application.
- The @Autowired annotation can be also applied on setter method and constructor.

Spring DI

- @Component beans are auto detectable by spring container. You don't need to define bean in configuration file, it will be automatically detected at runtime by Spring

- In Spring @Component, @Service, @Controller, and @Repository are Stereotype annotations which are used for:
- @Controller: where your request mapping from presentation page done i.e. Presentation layer won't go to any other file it goes directly to @Controller class and checks for requested path in @RequestMapping annotation which written before method calls if necessary.
- @Service: All business logic is here i.e. Data related calculations and all.This annotation of business layer in which our user not directly call persistence method so it will call this method using this annotation. It will request @Repository as per user request
- @Repository: This is Persistence layer(Data Access Layer) of application which used to get data from the database. i.e. all the Database related operations are done by the repository.
- @Component - Annotate your other components (for example REST resource classes) with a component stereotype.

Spring DI



Spring 2.5 introduces further stereotype annotations: @Component, @Service and @Controller. @Component serves as a generic stereotype for any Spring-managed component; whereas, @Repository, @Service, and @Controller serve as specializations of @Component for more specific use cases (e.g., in the persistence, service, and presentation layers, respectively).

Queries?