Spring Dependency
ıjection (Java config)

- Spring MVC beginner tutorial
- Spring MVC Exception Handling
- Spring MVC and log4j
- Spring MVC Send email
- Spring MVC File Upload
- Spring MVC Form Handling
- Spring MVC Form Validation
- Spring MVC File Download
- Spring MVC JdbcTemplate
- Spring MVC CSV view
- Spring MVC Excel View
- Spring MVC PDF View
- Spring MVC XstlView
- Spring MVC + Spring Data JPA + Hibernate - CRUD
- Spring MVC Security (XML)
- Spring MVC Security (Java config)
- Spring & Hibernate Integration (XML)
- Spring & Hibernate Integration (Java

config)

- Spring & Struts Integration (XML)
- Spring & Struts Integration (Java config)
- 14 Tips for Writing Spring MVC Controller

Spring Dependency Injection Example with Annotations

Written by Nam Ha Minh
Last Updated on 24 June 2019 | Print Email

This Spring tutorial helps you understand how to use Java annotations to configure dependency injection for classes in an application. Besides using XML for dependency injection configuration, Spring also allows programmers to embed some special annotations into Java classes to do the same thing.

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.

NOTE: If you want to understand the core concepts of dependency injection, please read What is Dependency Injection with Java Code Example tutorial.

1. Understand Spring Annotations for Dependency Injection

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.

ll Spring to inject an instance of another class into this class, declare an instance field
:he **`@Autowired`** annotation, for example:

```
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 **`MyService`**with the name **`service`**and inject into the instance **`client1`**of `MyClientImpl` class. Therefore, an implementation class of `MyService` should be annotated like this:

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

@Component("service")
public class MyServiceImpl2 implements MyService {
…
}
```

You see, this MyServiceImpl2class is annotated with the `@Component` annotation with the name `service` which matches the name of the corresponding field in the MyClientImplclass:

```
@Autowired
private MyService service;
```

Also the MyServiceImpl2class must implements the `MyService` interface to match the type of the autowired field in the MyClientImplclass. You see, using the `@Autowired` annotation on a field is simpler than setter and constructor injection.

NOTES:

- 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.

Now, let’s see how to create sample project in Eclipse IDE to demonstrate dependency injection with Spring framework.

Create Maven Project in Eclipse

To create a simple Maven project in Eclipse, click menu **File > New > Maven Project**. Then check the option *Create a simple project (skip archetype selection)*. Enter project's Group Id and Artifact Id, and then add the following XML in the `pom.xml` file:

```
<dependencies>
  <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.0.7.RELEASE</version>
  </dependency>
</dependencies>
```

This dependency `spring-context` is the minimum requirement to use dependency injection with Spring.

Next, write code for Java classes in the package `net.codejava` under the `src/main/java` folder as below:

Code of the `MyClient` interface:

```
package net.codejava;

public interface MyClient {
        void doSomething();
}
```

Code of the `ClientImpl` class:

```
package net.codejava;

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

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

        @Autowired
        private MyService service;

        @Override
        public void doSomething() {
                String info = service.getInfo();
                System.out.println(info);
        }

}
```

Code of the `MyService` interface:

```
  <age net.codejava;

public interface MyService {
        String getInfo();
}
```

An implementation of `MyService` interface:

```
package net.codejava;

import org.springframework.stereotype.Component;

@Component("service1")
public class MyServiceImpl1 implements MyService {

        @Override
        public String getInfo() {
                return "Service 1's Info";
        }

}
```

Another implementation of `MyService` interface:

```
package net.codejava;

import org.springframework.stereotype.Component;

@Component("service")
public class MyServiceImpl2 implements MyService {

        @Override
        public String getInfo() {
                return "Service 2's Info";
        }

}
```

3. Test Spring Dependency Injection with Annotations

Create a class with main method as shown below:

```
   ‹age net.codejava;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class SpringDependencyInjectionExample {

        public static void main(String[] args) {
                AnnotationConfigApplicationContext appContext = new AnnotationConfigA
pplicationContext();
                appContext.scan("net.codejava");
                appContext.refresh();

                MyClient client = (MyClient) appContext.getBean("client1");
                client.doSomething();
        }

}
```

Here, you see an instance of AnnotationConfigApplicationContextis created to scan a Java package to instantiate Spring-annotated classes as managed beans in the container:

```
appContext.scan("net.codejava");
```

Then we need to refresh the application context to update the changes after scanning:

```
appContext.refresh();
```

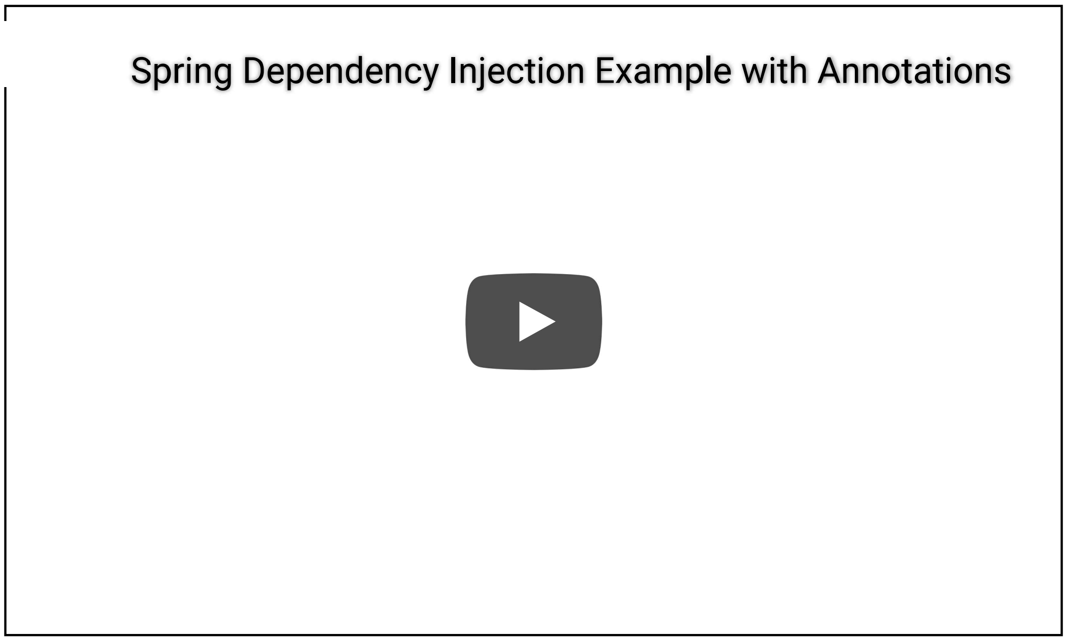
Then we get the bean named `client1` in the container, cast it to `MyClient` type and invoke its method:

```
MyClient client = (MyClient) appContext.getBean("client1");
client.doSomething();
```

The program prints the following output:

```
Service 2's Info
```

That’s basically how to use Spring annotations `@Component` and `@Autowired` to configure dependency injection in an application. We hope you found this tutorial helpful in terms of helping you getting started with Spring - one of the most popular frameworks for developing enterprise Java applications.



References:

- Annotation-based configuration (Spring docs)

Related Spring Dependency Injection Tutorials:

- Spring Dependency Injection Example with XML Configuration
- Spring Dependency Injection Example with Java Config

Other Spring Tutorials:

- Understand the core of Spring framework
- Understand Spring MVC
- Understand Spring AOP
- Spring MVC beginner tutorial with Spring Tool Suite IDE
- Spring MVC Form Handling Tutorial
- Spring MVC Form Validation Tutorial
- 14 Tips for Writing Spring MVC Controller
- Spring Web MVC Security Basic Example (XML Configuration)
- Understand Spring Data JPA with Simple Example

About the Author:

Nam Ha Minh is certified Java programmer (SCJP and SCWCD). He started programming with Java in the time of Java 1.4 and has been falling in love with Java since then. Make friend with him on Facebook.

Attachments:

SpringDependencyInjectionAnnotation.zip	**[Sample project for Spring dependency injection using annotations]**	**14 kB**

Add comment

Name E-mail

comment

Notify me of follow-up comments

I'm not a robot

reCAPTCHA

Privacy - Terms

Send

Comments

/