Session-32

Hello World Application, Dependency Injection

Spring and Java HelloWorld Example

We have created a bean or a Java class called Hello, which accepts a String message as a dependency. This spring bean is initialized using a spring configuration file like spring-config.xml. All beans declared in the spring configuration file is created and managed by the Spring IOC container.

If you look Spring configuration file, then you will find that id of the bean is "hello", which will be further used to get the reference of this bean from Spring framework using the getBean() method of ApplicationContext or BeanFactory class.

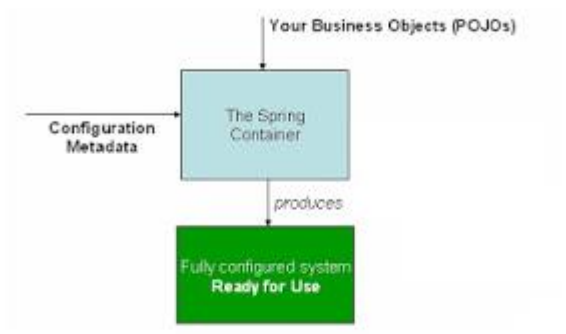
In order to test this Spring HelloWorld example, we have created a Main class, which has a classical public static void main(String args[]) method. In the main method, we are creating an instance of ClassPathXMLApplicationContext by providing spring-config.xml, which must be available in the classpath, in order for Spring to initialize beans.

Next two lines of code are self-explanatory, where we are getting the reference of Hello bean by using the "id" attribute of Hello bean declaration.

Btw, if you are a complete beginner on Spring framework and not familiar with essential Spring terminology like Beans, ApplicationContext etc, then I also suggest you to first go through Spring Fundamentals By Bryan Hansen on Pluralsight. A nice little course to understand the basics-of-basics of Spring framework

Spring IOC Containers

- How does an IoC container works?



You might have noticed that for printing HelloWorld in the console, Instead of using System.out.println, we have used log4j. As it's important to set up logging in every Java application because System.out.println() is not good enough for real-world Java application.

I have configured log4j using log4j.xml which is also available in the classpath. By the way, there are mainly two ways to inject dependency using Spring, Constructor injection, and Setter Injection, and this example uses setter injection to pass the message to Hello object.

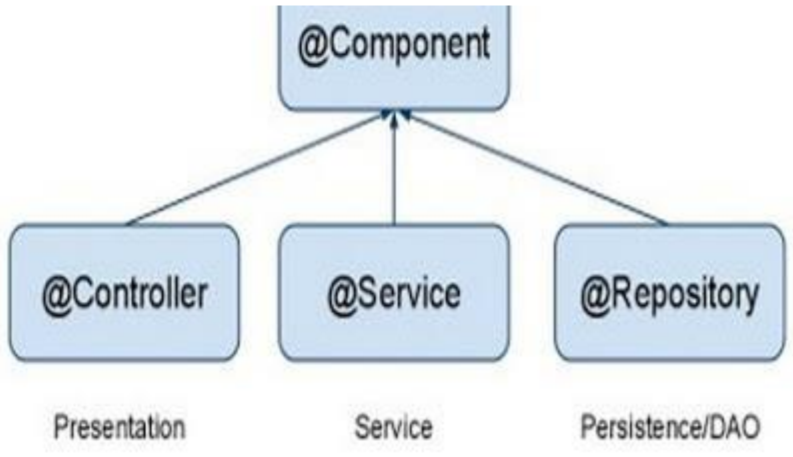
So this Spring hello world example contains the following configuration files and Java classes.

- Hello.java - Hello Bean which prints given message provided using spring dependency Injection
- Main.java - Test class for this Spring HelloWorld example
- spring-config.xml - Spring configuration file which contains bean declaration
- log4j.xml - log4j configuration file

If you want to implement this example using constructor injection then you need to make a couple of changes in Hello class and Spring configuration file.

Make sure your Hello class accept the String message in the constructor and when you configure that bean in Spring configuration file, instead of using property, use the constructor-arg tag.

If you are interested, you can also check **Spring Master Class - Beginner to Expert** course on Udemy for a couple of more examples on constructor injection and writing hello world using annotation in Spring 5.0.



Required Spring dependency JAR files

This Spring hello world example uses Spring 3.1.2.RELEASE.jar files but you can use the version you want like Spring 5.1 as well. Though, it's better to use Maven or Gradle for dependency injection because they will automatically download the right version of dependent JAR files for you.

- spring-core-3.1.2.RELEASE.jar
- commons-logging-1.1.1.jar
- log4j-1.2.16.jar
- spring-context-3.1.2.RELEASE.jar
- spring-aop-3.1.2.RELEASE.jar
- aopalliance-1.0.jar
- spring-beans-3.1.2.RELEASE.jar

In order to run this Spring hello world example, just run the Main class as Java application from command line or Eclipse IDE. Btw, if you are interested in learning Spring 5 and Spring Boot 2 from scratch, in a guided, code-focused way, **Learn Spring: The Certification Class** Eugen is another awesome course to try out.

Spring HelloWorld Java Class

```java
import org.apache.log4j.Logger;

/*
 * Java class which accept the message as dependency injected
 */
public class Hello {

	private static final Logger logger = Logger.getLogger(Hello.class);
	private String message;

	public void setMessage(String message) {
		this.message = message;
	}

	public String getMessage() {
		return message;
	}

	public void sayHello(){
		logger.info("Hello world Spring message is:" + message);	}

}
```

Main-Class to Test HelloWorld Bean

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

/*
 * Main class to start and test this Java application
 */
public class Main {

	public static void main(String args[]){
		ApplicationContext context = new ClassPathXmlApplicationContext(
					"spring-config.xml");
		 Hello hello = (Hello) context.getBean("hello");
		 hello.sayHello();
	}
}
```

log4j.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration xmlns:log4j='http://jakarta.apache.org/log4j/'>

	<appender name="console" class="org.apache.log4j.ConsoleAppender" >
	   <layout class="org.apache.log4j.PatternLayout">
	       <param name="ConversionPattern" value="%d %-4r [%t] %-5p %c %x - %m%n" />
	   </layout>
	</appender>

	<logger name="org.springframework" >
	    <level value="ERROR" />
	    <appender-ref ref="console" />
	</logger>

	<logger name="org.apache">
	    <level value="DEBUG" />
	    <appender-ref ref="console" />
	</logger>

	<root>
	    <level value="DEBUG" />
```

```
        <appender-ref ref="console" />
    </root>
</log4j:configuration>
```

Spring Configuration File spring-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">

    <bean id="hello" class="Hello">
        <property name="message" value="Good Afternoon" />
    </bean>

</beans>
```

Output

```
Hello world Spring message is: Good Afternoon
```