Spring Framework

Spring Tutorials

Spring is a replacement of EJB. It is created by Rod Johnson and owned by Spring Company.

Initial name of Spring was interface 21.

Spring has 2 features

1. Lightweight
2. Loosely Coupled

Suppose 2 classes are there namely A and B. If we create B class instance directly into A class, then two classes will become tightly coupled. And in future we won’t be able to upgrade from B class to C class. So, to overcome this problem we can go with interface model.

I -----> B I -----> C

Instead of creating this B class instance directly we can use the java RTP for this.

I obj = new B();

I obj = new C();

This object we are passing directly we can pass with the help of the xml model/annotations, that thing is basically called as Inversion of Control.

Spring recommends us to use association instead of inheritance.

By using runtime polymorphism and association they implemented Spring.

1. RTP
2. HAS A

To pass the runtime argument carrying the object to be replaced in the Interface model we can use Reflection class.

Else we need to have container support to carry it forward.

**IOC – 1. Core Container 2. J2EE Container**

**Spring MVC – 1. Web Container**

This web container is built on the top of IOC only.

These containers main job is to read data from xml and pass it to the POJO classes.

Core Container – BeanFactory(I)

J2EE Container – ApplicationContext(I)

Web Container – WebApplicationContext(I)

What is a container actually ?

Tomcat Container

1. Read web.xml
2. Configure servlet config and context objects. Find load on start-up servlet.
3. If load on start-up servlet is found create the object. (Basically the lifecycle methods like init , service and destroy).

Tomcat Container

1. Read xml file.
2. Create instances of xml beans.
3. Manage the lifecycle of bean classes.
4. Dynamic parameters supplies to bean claases.

IOC Container

1. Read xml file.
2. Create the object of POJO class.
3. Manage the lifecycle of the POJO classes.
4. Dynamic inputs you can pass from xml to POJO classes. This is nothing but the dependency injection.

\*\* Dependency Injection is the main asset of the IOC container.Just passing data from the xml file to the POJO classes.

Tomcat container we are switching on and off with the help of a button. Start and Stop button.

On the contrary IOC containers offers with some interface. Now how to start and stop an interface ?

We need to have implementations of those interfaces.

BeanFactory(I) --> XmlBeanFactory

ApplicationContext(I) --> ConfigurableApplicationContext(I) --> ClassPathApplicationContext

WebApplicationContext(I) --> Factory class is given named as WebApplicationContextUtil like DriverManager class.

To Start a class we need a main method in our case it is the DriverClass.

Class Test{

public static void main(String args[]){

new XmlBeanFactory();

new ClassPathApplicationContext();

WebApplicationContextUtil.getObject(); //This method will give webapplication context

}

}

So to start a application we need a DriverClass.

So for the servlet application container class is the Driver class.

Note :

In case at start-up you want to start your Spring container you need to write code in Spring init method. For each user request we have to write code in service method.

Main required components for Spring Application

1. POJO class
2. Xml file
3. Driver class

**POJO class**

Class Test{

public static hello(){

System.out.println(“Hello World”);

}

}

**Xml file (Spring.xml)**

dtd or xsd spring-beans.jar(org.springframework.core.factory.xml.spring-beans-2.0.dtd)

<beans>

<bean class=”Test” id=”t”/>

</beans>

**Driver Class**

Class Client{

public static void main(String args[]){

Resource r = new classPathResource(“spring.xml”);

BeanFactory factory = new XmlBeanFactory(r);

// Create test class Object

Test t = factory.getBean(“t”);

Note : Make this class a singleton , if we call this object multiple time also then also one object will be created , and same object will be assigned to all.

t.hello();

}

}

IOC factory is creating only one singleton object and same object is returned to all the users.

This scope is singleton by default and can be changed accordingly.

Singleton=true/false

Scope= singleton/prototype

<bean class=”Test” id=”t” scope=”prototype”/>

Scope singleton == Then for multiple user request it will create one single object.

Scope prototype == Then for multiple user request new object is created for every request.