What is the Spring Framework?

Spring is a Dependency Injection Framework to make java application loosely coupled.

It is an open source software development framework that provides infrastructure support for building primarily Java-based applications. One of the most popular Java Enterprise Edition (Java EE) frameworks, Spring helps developers create high-performing applications using plain old Java objects (POJOs). Other popular Java frameworks include Java Server Faces (JSF), Maven, Hibernate and Struts.

It was developed by Rod Jhonson in 2002.

Dependency Injection

Dependency Injection is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application. Dependency Injection makes our programming code loosely coupled.

Spring IoC (Inversion of Control) Container is the core of Spring Framework. It creates the objects, configures and assembles their dependencies, manages their entire life cycle. The Container uses Dependency Injection(DI) to manage the components that make up the application. It gets the information about the objects from a configuration file(XML) or Java Code or Java Annotations and Java POJO class. These objects are called Beans. Since the Controlling of Java objects and their lifecycle is not done by the developers, hence the name Inversion Of Control. The followings are some of the main features of Spring IoC,

- Creating Object for us,
- Managing our objects,
- Helping our application to be configurable,

- Managing dependencies

```
class Mario {
    Luigi l;
    void show() {
      }
}

class Luigi {
    void show() {
      }
}
```

Spring IoC Container

Spring IoC Container is the core of Spring Framework. It creates the objects, configures and assembles their dependencies, manages their entire life cycle. The Container uses Dependency Injection(DI) to manage the components that make up the application. It gets the information about the objects from a configuration file(XML) or Java Code or Java Annotations and Java POJO class.

These objects are called Beans. Since the Controlling of Java objects and their lifecycle is not done by the developers, hence the name Inversion Of Control.

1. Create the object
2. Hold it in the memory
3. Inject the object in another object as required

Beans – We need to set the spring container how many POJO classes need to be managed.

Configuration – In this case we will provide the configuration information and bean information. In configuration information we will provide the information of which bean is dependent on which bean. Then the spring container will use the configuration and create the objects of the bean and it will inject them as required.

Application – Now our application will use get method to use the objects.

ApplicationContext – It is an interface. ApplicationContext represents the Spring IoC container that holds all the beans created by the application. It is responsible for instantiating, configuring, and creating the beans. Additionally, it gets the bean's information from configuration metadata provided in XML or Java.

But we cannot create the object of it because it is an interface. So we can create the sub class's object which belongs to it.

1. ClassPathXmlApplicationContext – It searches the XML configuration in the java CLASSPATH. It loads the definitions of the beans from an XML file. Here we do not need to provide the full path of the XML file but you need to set CLASSPATH properly because this container will look like bean configuration XML file in CLASSPATH.
2. AnnotationConfigApplicationContext – It searches the beans in which we have used the annotation.
3. FileSystemXmlApplicationContext – It is used to load XML-based Spring Configuration files from the file system or from URL. We can get the application context using Java code.

There are two ways to perform dependency injection

1. Setter Injection
2. Constructor Injection

Setter Injection

Setter injection is a dependency injection in which the spring framework injects the dependency object using the setter method. The call first goes to no argument constructor and then to the setter method. It does not create any new bean instance. Let's see an example to inject dependency by the setter method.

```
class Employee {
    int id, name, salary;
    setId(id) {
    }
    setName(name) {
    }
    setSalary(salary) {
    }
}


class Salary {
    int grossSal, tax, totalSal;
    setGrossSal (grossSal) {
    }
    setTax(tax) {
    }
    setTotalSal(totalSal) {
    }
}
```

Constructor Injection

In Constructor Injection, the Dependency Injection will be injected with the help of constructors. Now to set the Dependency Injection as Constructor Dependency Injection in bean, it is done through the bean-configuration file.

```
class Employee {
     int id, name, salary;
     Employee(id, name, salary) {
   }
}


class Salary {
     int grossSal, tax, totalSal;
     Salary(grossSal, tax, totalSal) {
  }
}
```

Configuration File

It is the file where we declare beans and it's dependency.

Data Types(Dependencies)

1. Primitive DataTypes – byte, short, char, int, float, double, long, Boolean
2. Collection Type – List, Map, Set and Properties.
3. Reference Type – Other class Object

Ambiguity Problem

The ambiguities are those issues that are not defined clearly in the Java language specification. The different results produced by different compilers on several example programs support our observations.
Now, in case of constructor based dependency injection if our class contains multiple constructors with different types and same number of arguments then spring framework cause the constructor injection argument type ambiguities issue.