**IoC Container**

The IoC container is responsible to instantiate, configure and assemble the objects. The IoC container gets informations from the XML file and works accordingly. The main tasks performed by IoC container are:

* to instantiate the application class
* to configure the object
* to assemble the dependencies between the objects

There are two types of IoC containers. They are:

1. **BeanFactory**
2. **ApplicationContext**

### Difference between BeanFactory and the ApplicationContext

The org.springframework.beans.factory.**BeanFactory** and the org.springframework.context.**ApplicationContext** interfaces acts as the IoC container. The ApplicationContext interface is built on top of the BeanFactory interface. It adds some extra functionality than BeanFactory such as simple integration with Spring's AOP, message resource handling (for I18N), event propagation, application layer specific context (e.g. WebApplicationContext) for web application. So it is better to use ApplicationContext than BeanFactory.

Dependency Injection (DI) is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application

SI with setter :

1. <bean id="q" **class**="com.javatpoint.Question">
2. <property name="id" value="1"></property>
3. <property name="name" value="What is Java?"></property>
4. <property name="answers">
5. <list>
6. <value>Java is a programming language</value>
7. <value>Java is a platform</value>
8. <value>Java is an Island</value>
9. </list>
10. </property>
11. </bean>

SI with constructer:

1. <bean id="q" **class**="com.javatpoint.Question">
2. <constructor-arg value="111"></constructor-arg>
3. <constructor-arg value="What is java?"></constructor-arg>
4. <constructor-arg>
5. <list>
6. <value>Java is a programming language</value>
7. <value>Java is a Platform</value>
8. <value>Java is an Island of Indonasia</value>
9. </list>
10. </constructor-arg>
11. </bean>

Difference between constructor and setter injection

1. [Difference between constructor and setter injection](https://www.javatpoint.com/difference-between-constructor-and-setter-injection)

There are many key differences between constructor injection and setter injection.

1. **Partial dependency**: can be injected using setter injection but it is not possible by constructor. Suppose there are 3 properties in a class, having 3 arg constructor and setters methods. In such case, if you want to pass information for only one property, it is possible by setter method only.
2. **Overriding**: Setter injection overrides the constructor injection. If we use both constructor and setter injection, IOC container will use the setter injection.
3. **Changes**: We can easily change the value by setter injection. It doesn't create a new bean instance always like constructor. So setter injection is flexible than constructor injection.

# Autowiring in Spring

Autowiring feature of spring framework enables you to inject the object dependency implicitly. It internally uses setter or constructor injection.

## Advantage of Autowiring

It requires the **less code** because we don't need to write the code to inject the dependency explicitly.

[**next →**](https://www.javatpoint.com/dependency-injection-with-factory-method)[**← prev**](https://www.javatpoint.com/spring-tutorial-constructor-injection-with-dependent-object)

# Autowiring in Spring

Autowiring feature of spring framework enables you to inject the object dependency implicitly. It internally uses setter or constructor injection.

Autowiring can't be used to inject primitive and string values. It works with reference only.

## Advantage of Autowiring

It requires the **less code** because we don't need to write the code to inject the dependency explicitly.

## Disadvantage of Autowiring

No control of programmer.

It can't be used for primitive and string values.