**Spring Tutorial**



This spring tutorial provides in-depth concepts of Spring Framework with simplified examples. It was **developed by Rod Johnson in 2003**. Spring framework makes the easy development of JavaEE application.

It is helpful for beginners and experienced persons.

**Spring Framework**

Spring is a *lightweight* framework. It can be thought of as a *framework of frameworks* because it provides support to various frameworks such as Struts, Hibernate, Tapestry, EJB, JSF etc. The framework, in broader sense, can be defined as a structure where we find solution of the various technical problems.

The Spring framework comprises several modules such as IOC, AOP, DAO, Context, ORM, WEB MVC etc. We will learn these modules in next page. Let's understand the IOC and Dependency Injection first.

**Inversion Of Control (IOC) and Dependency Injection**

These are the design patterns that are used to remove dependency from the programming code. They make the code easier to test and maintain. Let's understand this with the following code:

1. class Employee{
2. Address address;
3. Employee(){
4. address=new Address();
5. }
6. }

In such case, there is dependency between the Employee and Address (tight coupling). In the Inversion of Control scenario, we do this something like this:

1. class Employee{
2. Address address;
3. Employee(Address address){
4. this.address=address;
5. }
6. }

Thus, IOC makes the code loosely coupled. In such case, there is no need to modify the code if our logic is moved to new environment.

In Spring framework, IOC container is responsible to inject the dependency. We provide metadata to the IOC container either by XML file or annotation.

**Advantage of Dependency Injection**

* makes the code loosely coupled so easy to maintain
* makes the code easy to test

**Advantages of Spring Framework**

There are many advantages of Spring Framework. They are as follows:

**1) Predefined Templates**

Spring framework provides templates for JDBC, Hibernate, JPA etc. technologies. So there is no need to write too much code. It hides the basic steps of these technologies.

Let's take the example of JdbcTemplate, you don't need to write the code for exception handling, creating connection, creating statement, committing transaction, closing connection etc. You need to write the code of executing query only. Thus, it save a lot of JDBC code.

**2) Loose Coupling**

The Spring applications are loosely coupled because of dependency injection.

**3) Easy to test**

The Dependency Injection makes easier to test the application. The EJB or Struts application require server to run the application but Spring framework doesn't require server.

**4) Lightweight**

Spring framework is lightweight because of its POJO implementation. The Spring Framework doesn't force the programmer to inherit any class or implement any interface. That is why it is said non-invasive.

**5) Fast Development**

The Dependency Injection feature of Spring Framework and it support to various frameworks makes the easy development of JavaEE application.

**6) Powerful abstraction**

It provides powerful abstraction to JavaEE specifications such as JMS, JDBC, JPA and JTA.

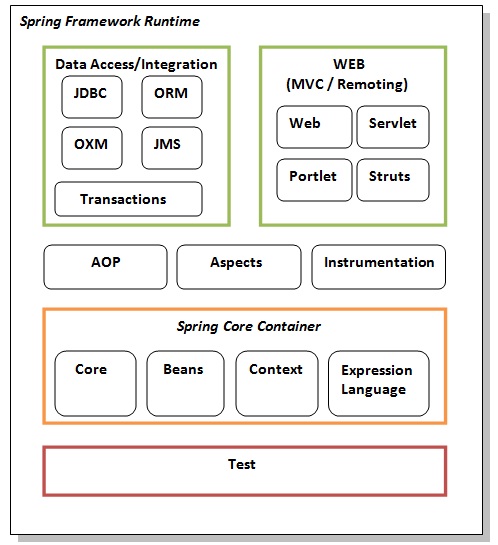
**7) Declarative support**

It provides declarative support for caching, validation, transactions and formatting.

**Spring Modules**

1. [Spring Modules](https://www.javatpoint.com/spring-modules)
2. [Test](https://www.javatpoint.com/spring-modules#sptest)
3. [Spring Core Container](https://www.javatpoint.com/spring-modules#spcore)
4. [AOP, Aspects and Instrumentation](https://www.javatpoint.com/spring-modules#spaop)
5. [Data Access / Integration](https://www.javatpoint.com/spring-modules#spda)
6. [Web](https://www.javatpoint.com/spring-modules#spweb)

The Spring framework comprises of many modules such as core, beans, context, expression language, AOP, Aspects, Instrumentation, JDBC, ORM, OXM, JMS, Transaction, Web, Servlet, Struts etc. These modules are grouped into Test, Core Container, AOP, Aspects, Instrumentation, Data Access / Integration, Web (MVC / Remoting) as displayed in the following diagram.



**Test**

This layer provides support of testing with JUnit and TestNG.

**Spring Core Container**

The Spring Core container contains core, beans, context and expression language (EL) modules.

**Core and Beans**

These modules provide IOC and Dependency Injection features.

**Context**

This module supports internationalization (I18N), EJB, JMS, Basic Remoting.

**Expression Language**

It is an extension to the EL defined in JSP. It provides support to setting and getting property values, method invocation, accessing collections and indexers, named variables, logical and arithmetic operators, retrieval of objects by name etc.

**AOP, Aspects and Instrumentation**

These modules support aspect oriented programming implementation where you can use Advices, Pointcuts etc. to decouple the code.

The aspects module provides support to integration with AspectJ.

The instrumentation module provides support to class instrumentation and classloader implementations.

**Data Access / Integration**

This group comprises of JDBC, ORM, OXM, JMS and Transaction modules. These modules basically provide support to interact with the database.

**Web**

This group comprises of Web, Web-Servlet, Web-Struts and Web-Portlet. These modules provide support to create web application.

Next Topic[Steps To Create Spring Application](https://www.javatpoint.com/steps-to-create-spring-application)

**Spring Example**

1. [Steps to create spring application](https://www.javatpoint.com/steps-to-create-spring-application)

Here, we are going to learn the simple steps to create the first spring application. To run this application, we are not using any IDE. We are simply using the command prompt. Let's see the simple steps to create the spring application

* **create the class**
* **create the xml file to provide the values**
* **create the test class**
* **Load the spring jar files**
* **Run the test class**

**Steps to create spring application**

Let's see the 5 steps to create the first spring application.

**1) Create Java class**

This is the simple java bean class containing the name property only.

1. package com.javatpoint;
3. public class Student {
4. private String name;
6. public String getName() {
7. return name;
8. }
10. public void setName(String name) {
11. this.name = name;
12. }
14. public void displayInfo(){
15. System.out.println("Hello: "+name);
16. }
17. }

This is simple bean class, containing only one property name with its getters and setters method. This class contains one extra method named displayInfo() that prints the student name by the hello message.

**2) Create the xml file**

In case of myeclipse IDE, you don't need to create the xml file as myeclipse does this for yourselves. Open the applicationContext.xml file, and write the following code:

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans
3. xmlns="http://www.springframework.org/schema/beans"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xmlns:p="http://www.springframework.org/schema/p"
6. xsi:schemaLocation="http://www.springframework.org/schema/beans
7. http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
9. <bean id="studentbean" class="com.javatpoint.Student">
10. <property name="name" value="Vimal Jaiswal"></property>
11. </bean>
13. </beans>

The **bean** element is used to define the bean for the given class. The **property** subelement of bean specifies the property of the Student class named name. The value specified in the property element will be set in the Student class object by the IOC container.

**3) Create the test class**

Create the java class e.g. Test. Here we are getting the object of Student class from the IOC container using the getBean() method of BeanFactory. Let's see the code of test class.

1. package com.javatpoint;
3. import org.springframework.beans.factory.BeanFactory;
4. import org.springframework.beans.factory.xml.XmlBeanFactory;
5. import org.springframework.core.io.ClassPathResource;
6. import org.springframework.core.io.Resource;
8. public class Test {
9. public static void main(String[] args) {
10. Resource resource=new ClassPathResource("applicationContext.xml");
11. BeanFactory factory=new XmlBeanFactory(resource);
13. Student student=(Student)factory.getBean("studentbean");
14. student.displayInfo();
15. }
16. }

The **Resource** object represents the information of applicationContext.xml file. The Resource is the interface and the **ClassPathResource** is the implementation class of the Reource interface. The **BeanFactory** is responsible to return the bean. The **XmlBeanFactory** is the implementation class of the BeanFactory. There are many methods in the BeanFactory interface. One method is **getBean()**, which returns the object of the associated class.

**4) Load the jar files required for spring framework**

There are mainly three jar files required to run this application.

* **org.springframework.core-3.0.1.RELEASE-A**
* **com.springsource.org.apache.commons.logging-1.1.1**
* **org.springframework.beans-3.0.1.RELEASE-A**

For the future use, You can download the required jar files for spring core application.

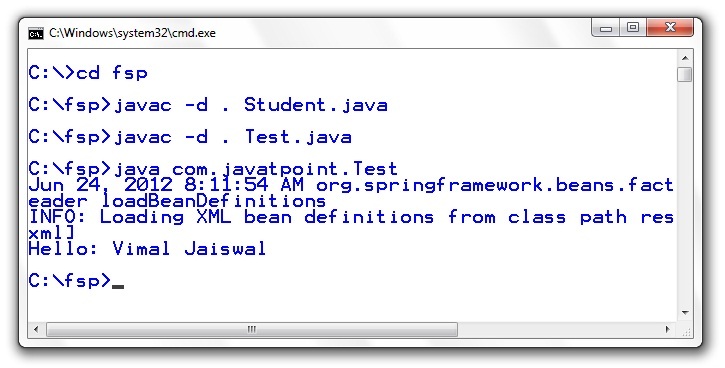
[download the core jar files for spring](https://static.javatpoint.com/src/sp/spcorejars.zip)

[download the all jar files for spring including core, web, aop, mvc, j2ee, remoting, oxm, jdbc, orm etc.](https://static.javatpoint.com/src/sp/springjars.zip)

To run this example, you need to load only spring core jar files.

**5) Run the test class**

Now run the Test class. You will get the output Hello: Vimal Jaiswal.



[download this example](https://static.javatpoint.com/src/sp/fsp.zip)

Next Topic[Example Of Spring Application In Myeclipse](https://www.javatpoint.com/example-of-spring-application-in-myeclipse)