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:

**class** Employee{

Address address;

Employee(){

address=**new** Address();

}

}

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:

**class** Employee{

Address address;

Employee(Address address){

**this**.address=address;

}

}

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 saves 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.

**BeanFactory**

The BeanFactory is the actual container which instantiates, configures, and manages a number of beans. These beans typically collaborate with one another, and thus have dependencies between themselves. These dependencies are reflected in the configuration data used by the BeanFactory (although some dependencies may not be visible as configuration data, but rather be a function of programmatic interactions between beans at runtime).

**ApplicationContext**

While the beans package provides basic functionality for managing and manipulating beans, often in a programmatic way, the context package adds ApplicationContext, which enhances BeanFactoryfunctionality in a more framework-oriented style. Many users will use ApplicationContext in a completely declarative fashion, not even having to create it manually, but instead relying on support classes such as ContextLoader to automatically start an ApplicationContext as part of the normal startup process of a Java EE web-app. Of course, it is still possible to programmatically create an ApplicationContext.