Spring Basics

Spring framework is an open source Java platform. It was initially written by Rod Johnson and was first released under the Apache 2.0 license in June 2003.

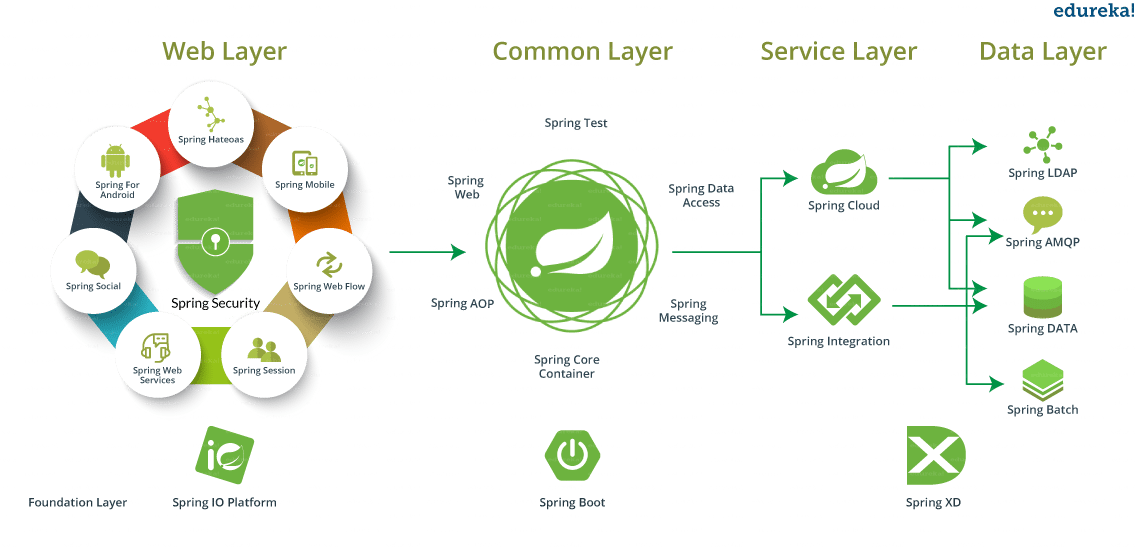
Spring is lightweight when it comes to size and transparency. The basic version of Spring framework is around 2MB.

The core features of the Spring Framework can be used in developing any Java application, but there are extensions for building web applications on top of the Java EE platform. Spring framework targets to make J2EE development easier to use and promotes good programming practices by enabling a POJO-based programming model.

**Features Of Spring Framework**

* **Lightweight:**Spring Framework is lightweight with respect to size and transparency.
* **Inversion Of Control (IOC):**In Spring Framework, loose coupling is achieved using Inversion of Control. The objects give their own dependencies instead of creating or looking for dependent objects.
* **Aspect Oriented Programming (AOP):**By separating application business logic from system services, Spring Framework supports Aspect Oriented Programming and enables cohesive development.
* **Container:**Spring Framework creates and manages the life cycle and configuration of application objects.
* **MVC Framework:**Spring Framework is a MVC web application framework. This framework is configurable via interfaces and accommodates multiple view technologies.
* **Transaction Management:**For transaction management,Spring framework provides a generic abstraction layer. It is not tied to J2EE environments and it can be used in container-less environments.
* **JDBC Exception Handling:**The JDBC abstraction layer of the Spring Framework offers an exception hierarchy, which simplifies the error handling strategy.

Since its origin till date, Spring has spread its popularity across various domains. Spring Framework now is the foundation for various other Spring Projects that have come up in the offerings in the last two to three years. Check the below image to find out various projects undertaken by Spring.



**Spring Tutorial: Advantages Of Using Spring Framework**

* Works on **POJO**s (Plain Old Java Object) which makes your application lightweight.
* Provides predefined templates for JDBC, Hibernate, JPA etc., thus reducing your effort of writing too much code.
* Because of dependency injection feature, your code becomes loosely coupled.
* Using Spring Framework, the development of **Java Enterprise Edition** (JEE) applications became faster.
* It also provides strong abstraction to Java Enterprise Edition (JEE) specifications.
* It provides declarative support for transactions, validation, caching and formatting.

**Spring Tutorial: First Spring Application**

Once you are done with the installation, you are ready to develop your first application in Spring. For that, you need to follow five simple steps:

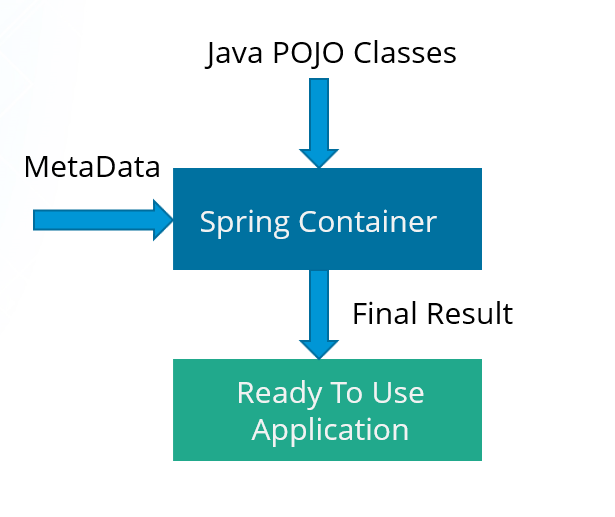
1. Create Maven project with required dependencies
2. Create the **Bean.java** class
3. Create a XML/ configuration file
4. Create the **main** class
5. Run the application

**Spring Tutorial: Spring IoC Container**

So what exactly is an IoC container in Spring? Well, Spring IoC stands for Inversion of Control. It is the heart of the Spring Framework. The important tasks performed by the IoC container are:

1. Instantiating the bean
2. Wiring the beans together
3. Configuring the beans
4. Managing the bean’s entire life-cycle

The IoC container receives metadata from either an XML file, Java annotations, or Java code and works accordingly. IoC adds the flexibility and control of application, and provides a central place of configuration management for Plain Old Java Objects (POJO) of our application.

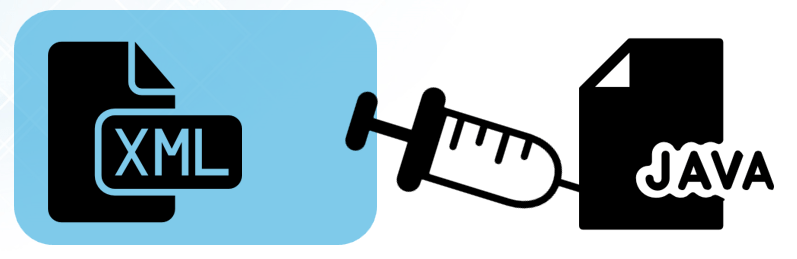


**Inversion of control as a design guideline** serves the following purposes:

1. There is a decoupling of the execution of a certain task from implementation.
2. Every module can focus on what it is designed for.
3. Modules make no assumptions about what other systems do but rely on their contracts.
4. Replacing modules has no side effect on other modules.

## ****Spring Tutorial: Dependency Injection****

Dependency Injection is also one of the **core concepts** of Spring Framework. It is a design pattern that removes the dependency from the code. That is, the Spring Framework provides the dependencies of the class itself so that it can be easy to manage and test the application. You can provide information from external source such as XML file. Here, you do not create the objects instead you just define how they should be created and IoC container will create the objects for you.



In Spring, dependencies can be injected in two ways:

1. By constructor
2. By setter method

**By Constructor**

* The **<constructor-arg>** subelement of **<bean>** is used for constructor injection. e.g:

|  |  |
| --- | --- |
| 1 | <constructor-arg value="101" type="int"></constructor-arg> |

* By default when the Spring container loads the bean, it instantiates the bean with the default constructor. But you can also define a constructor argument in bean definition, using an argument constructor.

**By setter method**

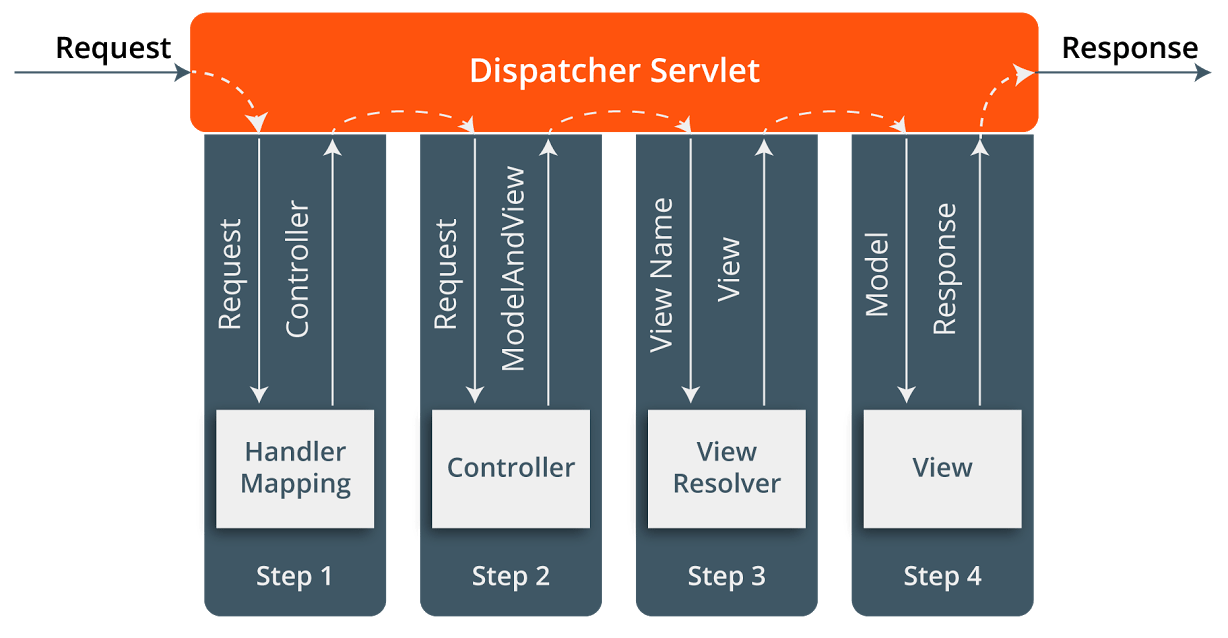
* The **<property>** subelement of **<bean>** is used for setter injection.e.g:

|  |  |
| --- | --- |
| 1 | <property name="id" value="101"></property> |

* Setter-based Dependency Injection is accomplished by the container calling setter methods on your beans after invoking a no-argument.

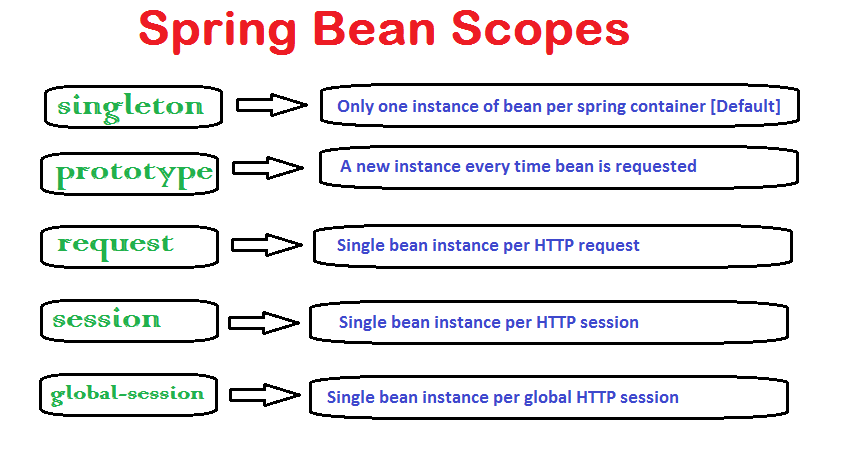
## ****Spring Tutorial: Spring MVC (Model-View-Controller)****

Spring MVC helps in building flexible and loosely coupled web applications. The Model-View-Controller design pattern helps in separating the business logic, presentation logic and navigation logic. It provides an elegant solution to use MVC in Spring Framework with the help of DispatcherServlet.



When a request is sent to the Spring MVC Framework, the following takes place.

* A request is received by DispatcherServlet.
* The DispatcherServlet communicates with HandlerMapping and calls the Controller associated with the request.
* The request is processed by the Controller by calling the appropriate service methods and a ModelAndView object to the DispatcherServlet is returned.
* The view name is sent to a ViewResolver by the DispatcherServlet to find the actual View toinvoke.
* Now the model object DispatcherServlet is passed to *View* to render the result.
* With the help of the model data, the View renders the result back to the user.



**Spring Bean Life Cycle**



## 2. Spring Bean Life Cycle Callback Methods

Spring framework provides following **4 ways for controlling life cycle events** of a bean:

1. InitializingBean and DisposableBean callback interfaces
2. \*Aware interfaces for specific behavior
3. Custom init() and destroy() methods in bean configuration file
4. @PostConstruct and @PreDestroy annotations

#### 7.5. What is Bean Factory ?

A BeanFactory is like a factory class that contains a collection of beans. The BeanFactory holds Bean Definitions of multiple beans within itself and then instantiates the bean whenever asked for by clients.

BeanFactory is able to create associations between collaborating objects as they are instantiated. This removes the burden of configuration from bean itself and the beans client. BeanFactory also takes part in the life cycle of a bean, making calls to custom initialization and destruction methods.

#### 7.6. What is Application Context?

A bean factory is fine to simple applications, but to take advantage of the full power of the Spring framework, you may want to move up to Springs more advanced container, the application context. On the surface, an application context is same as a bean factory.Both load bean definitions, wire beans together, and dispense beans upon request. But it also provides:

* A means for resolving text messages, including support for internationalization.
* A generic way to load file resources.
* Events to beans that are registered as listeners.

#### 7.7. What are the common implementations of the Application Context?

The three commonly used implementation of ApplicationContext are:

1. ClassPathXmlApplicationContext : It Loads context definition from an XML file located in the classpath, treating context definitions as classpath resources. The application context is loaded from the application’s classpath by using the code .

|  |
| --- |
| ApplicationContext context = new ClassPathXmlApplicationContext("bean.xml"); |

1. FileSystemXmlApplicationContext : It loads context definition from an XML file in the filesystem. The application context is loaded from the file system by using the code .

|  |
| --- |
| ApplicationContext context = new FileSystemXmlApplicationContext("bean.xml"); |

1. XmlWebApplicationContext : It loads context definition from an XML file contained within a web application.

# **References**

<https://www.udemy.com/spring-hibernate-tutorial/>

<https://www.edureka.co/blog/spring-tutorial/>

<https://www.tutorialspoint.com/spring/index.htm>

<https://howtodoinjava.com/java-spring-framework-tutorials/>