**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](https://www.javatpoint.com/struts-2-tutorial), [Hibernate](https://www.javatpoint.com/hibernate-tutorial), Tapestry, [EJB](https://www.javatpoint.com/ejb-tutorial), [JSF](https://www.javatpoint.com/jsf-tutorial), 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.

**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.

**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](https://www.javatpoint.com/jms-tutorial), [JDBC](https://www.javatpoint.com/java-jdbc), JPA and JTA.

**7) Declarative support**

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

## **Steps to create spring application**

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 using Spring Initlizr**

**Spring Initializr** is a **web-based tool** provided by the Pivotal Web Service. With the help of **Spring Initializr**, we can easily generate the structure of the **Spring Boot Project**. It offers extensible API for creating JVM-based projects

1. Navigate to [https://start.spring.io](https://start.spring.io/). This service pulls in all the dependencies you need for an application and does most of the setup for you.
2. Choose either Gradle or Maven and the language you want to use. This guide assumes that you chose Java.
3. Click **Dependencies** and select **Spring Web**.
4. Click **Generate**.
5. Download the resulting ZIP file, which is an archive of a web application that is configured with your choices.

# **Spring JdbcTemplate**

Spring **JdbcTemplate** is a powerful mechanism to connect to the database and execute SQL queries. It internally uses JDBC api, but eliminates a lot of problems of JDBC API

## **Problems of JDBC API**

The problems of JDBC API are as follows:

* We need to write a lot of code before and after executing the query, such as creating connection, statement, closing resultset, connection etc.
* We need to perform exception handling code on the database logic.
* We need to handle transaction.
* Repetition of all these codes from one to another database logic is a time consuming task.

## **Advantage of Spring JdbcTemplate**

Spring JdbcTemplate eliminates all the above mentioned problems of JDBC API. It provides you methods to write the queries directly, so it saves a lot of work and time.

## **Spring Jdbc Approaches**

Spring framework provides following approaches for JDBC database access:

* JdbcTemplate
* NamedParameterJdbcTemplate
* SimpleJdbcTemplate
* SimpleJdbcInsert and SimpleJdbcCall

# **Spring with ORM Frameworks**

Spring provides API to easily integrate Spring with ORM frameworks such as Hibernate, JPA(Java Persistence API), JDO(Java Data Objects), Oracle Toplink and iBATIS.

**Advantage of ORM Frameworks with Spring**

There are a lot of advantage of Spring framework in respect to ORM frameworks. There are as follows:

* **Less coding is required**
* **Easy to test**:
* **Better exception handling**
* **Integrated transaction management**

# **Hibernate and Spring Integration**

We can simply integrate **hibernate application with spring application**.In hibernate framework, we provide all the database information hibernate.cfg.xml file.

But if we are going to integrate the hibernate application with spring, we don't need to create the hibernate.cfg.xml file. We can provide all the information in the applicationContext.xml file.

### Advantage of Spring framework with hibernate

## The Spring framework provides **HibernateTemplate** class, so you don't need to follow so many steps like create Configuration, BuildSessionFactory, Session, beginning and committing transaction etc. So it saves a lot of code.

## **Steps**

Let's see what are the simple steps for hibernate and spring integration:

1. **create table in the database** It is optional.
2. **create applicationContext.xml file** It contains information of DataSource, SessionFactory etc.
3. **create Employee.java file** It is the persistent class
4. **create employee.hbm.xml file** It is the mapping file.
5. **create EmployeeDao.java file** It is the dao class that uses HibernateTemplate.
6. **create InsertTest.java file** It calls methods of EmployeeDao class.

Methods of HibernateTemplate class

Let's see a list of commonly used methods of HibernateTemplate class.

1) **void persist(Object entity)**: persists the given object.

2) **Serializable save(Object entity)**: persists the given object and returns id.

3) **void saveOrUpdate(Object entity)**: persists or updates the given object. If id is found, it updates the record otherwise saves the record.

4) **void update(Object entity):** updates the given object.

5) **void delete(Object entity)**: deletes the given object on the basis of id.

6) **Object get(Class entityClass, Serializable id):** returns the persistent object on the basis of given id.

7) **Object load(Class entityClass, Serializable id)**: returns the persistent object on the basis of given id.

8) **List loadAll(Class entityClass)**: returns the all the persistent objects.

# **Spring Data JPA**

Spring Data JPA API provides JpaTemplate class to integrate spring application with JPA.

JPA (Java Persistent API) is the sun specification for persisting objects in the enterprise application. It is currently used as the replacement for complex entity beans

## **Advantage of Spring JpaTemplate**

You don't need to write the before and after code for persisting, updating, deleting or searching object such as creating Persistence instance, creating EntityManagerFactory instance, creating EntityTransaction instance, creating EntityManager instance, commiting EntityTransaction instance and closing EntityManager.

# **Spring MVC**

A Spring MVC is a Java framework which is used to build web applications. It follows the Model-View-Controller design pattern. It implements all the basic features of a core spring framework like Inversion of Control, Dependency Injection.

A Spring MVC provides an elegant solution to use MVC in spring framework by the help of **DispatcherServlet**. Here, **DispatcherServlet** is a class that receives the incoming request and maps it to the right resource such as controllers, models, and views.

* **Model** - A model contains the data of the application. A data can be a single object or a collection of objects.
* **Controller** - A controller contains the business logic of an application. Here, the @Controller annotation is used to mark the class as the controller.
* **View** - A view represents the provided information in a particular format. Generally, JSP+JSTL is used to create a view page. Although spring also supports other view technologies such as Apache Velocity, Thymeleaf and FreeMarker.
* **Front Controller** - In Spring Web MVC, the DispatcherServlet class works as the front controller. It is responsible to manage the flow of the Spring MVC application.

## **Advantages of Spring MVC Framework**

Let's see some of the advantages of Spring MVC Framework:-

* **Separate roles** - The Spring MVC separates each role, where the model object, controller, command object, view resolver, DispatcherServlet, validator, etc. can be fulfilled by a specialized object.
* **Light-weight** - It uses light-weight servlet container to develop and deploy your application.
* **Powerful Configuration** - It provides a robust configuration for both framework and application classes that includes easy referencing across contexts, such as from web controllers to business objects and validators.
* **Rapid development** - The Spring MVC facilitates fast and parallel development.
* **Reusable business code** - Instead of creating new objects, it allows us to use the existing business objects.
* **Easy to test** - In Spring, generally we create JavaBeans classes that enable you to inject test data using the setter methods.
* **Flexible Mapping** - It provides the specific annotations that easily redirect the page.

Steps for Spring Web MVC framework:

* Load the spring jar files or add dependencies in the case of Maven
* Create the controller class
* Provide the entry of controller in the web.xml file
* Define the bean in the separate XML file
* Display the message in the JSP page
* Start the server and deploy the project