Spring Framework

It helps you to create various types of applications like Desktop, Web applications, Enterprise applications, Cloud based applications

Spring Modules

1. Spring IoC
2. Spring Web MVC
3. Spring JDBC
4. Spring Boot
5. Spring Cloud

Spring IoC:

Inversion of Control, because object creation process is inverted, it helps spring container to create the dependencies and supply to other objects.

Spring Libraries for IOC

1. Spring Context: This gives you all the necessary jar files to get the Spring IOC benefits like DI, Design Patterns, Container etc.

Dependency Injection:

It is an approach of supplying an object into another object to make loosely coupled application.

There are two types of Dependency Injection

1. Setter Injection: It uses setter method to initialize the object
2. Constructor Injection: It uses constructor argument to initialize the object

For setter injection you will use <property> tag

For constructor injection you will use <constructor-arg> tag

Configuring complex dependencies

Whenever an object depends on another object we may need to supply these complex dependencies we can configure a <bean> and supply that to another <bean> using ref attribute.

class MySQLImpl implements DBOperations {   
  
 Datasource ds;

}

<property name = “ds” ref = “beanId” />

Bean Scopes

By default beans are singleton, you can change the scope to prototype when you need multiple instance of the same bean id.

<bean id = “b1” class = “com.A”>

The object of class A is singleton

<bean id = “b2” class = “com.B” scope = “prototype”>

The object of class B is prototype i.e., on each getBean(“”) call you get a new object

Annotation based configuration

Spring gives you various annotations to register the beans in the container so that you can avoid configuring XML as much as possible

Spring has given one base annotation called @Component which registers the object in the spring container there are other component type annotations that would do the same job which are:-

* @Repository
* @Service
* @Controller
* @RestController

@Component  
class A {   
}  
@Service   
class B {   
}  
@Repository   
class C {   
}

XYZ >> class Testing { }

Advantages of annotations

1. Simple to use
2. Most of the complex dependencies can be supplied to objects using @Autowired annotation & it doesn’t need any setter or constructor to initialize

@Repository  
class A {   
}  
@Service  
class B {   
 @Autowired  
 private A obj;  
}

Note:

Annotation configuration will work only when you use <component-scan> tag in your XML file that let spring know from where it should scan the classes

AOP

Aspect Oriented Programming, which helps you to call crossing cutting concerns without actually invoking it, this is achieved using some of the AOP features

1. Aspect: These are the classes that will have cross-cutting concerns
2. Advice: These are the cross-cutting logics which are executed before or after the actual method (join point)
3. Join Point: These are the actual methods of the application where you want the advice to be run around it.
4. Point Cut Expressions: This is an expression written on the advice that tells AOP on which Join point the advice must be run

Pointcut expressions

execution(“public void com.TestService.test()”)

execution(“\* com.TestService.\*(..)”): Selects all the methods of TestService

execution(“\* com.\*.set\*(..)”): Selects setter methods of all the classes inside the com package

execution(“\* com.TestService.login(String, \*)”): Selects login method of TestService 1st parameter must be String & 2nd parameter could be anything

Spring MVC

It is used to develop web applications using MVC architecture, where you can abstract all the layers which are dependencies of another layer, so that you can have a loosely coupled application

i.e., Service layer depends on DAO layer, Controller layer depends on Service layer, here these dependency can be abstracted so that changes are limited

How Spring MVC helps you to create web application

It gives you a front-controller called DispatcherServlet who takes care of accepting all the requests and routing to appropriate controller

Tasks of Front Controller

1. Accepts all the incoming requests & route to appropriate controller
2. Configures the beans and initializes the spring container i.e, ApplicationContext
3. Uses a bean to resolve the JSP views

Configure web application in spring

1. We need an external server
2. We need to create a dynamic web project
3. Add spring WebMvc dependency
4. web.xml must have front-controller configuration which is DispatcherServlet
5. Spring Bean Configuration file

Note: DispatcherServlet will take care of loading the spring bean configuration file, it will know about the spring configuration xml file from web.xml

Front Controller: It is used to take requests and map to the controller

Controller: It is a class with a method that can handle the request & generate the response, controller depends on service layer

@Controller: It is the annotation used to create controller, this must be marked on top of the class, these will have one or more methods to handle the request & show the views

ReSTful webservices: These are online services/API’s which helps heterogenous applications to exchange the data, more commonly used data-format is JSON

@RestController: It is the annotation used to create controller that can handle rest calls and generate data instead of response.

We can have separate applications for front-end & backend because of webservices

ex: ATM machine of different banks can exchange data with different banking services

How to create API’s/design with ReST webservices

* We must create controller with specific URL
* We must map the controller with HTTP methods so that clients can use the same HTTP method to access the API
* We must specify what datastructure it can consume & produce

How to configure webservices in spring

1. We need to create controller using @RestController
2. We need to add a library that takes care of converting Java objects to JSON & Vice versa the library is Jackson-databind
3. We need to create API’s with right HTTP methods based on the type of operations
4. GET - Fetch
5. POST - Create
6. PUT - Update
7. DELETE - Remove

Note: We need Postman application to test the webservices

What are the ways Java can represent its data in JSON format

1. A java object can be represented in JSON format, because java objects will have properties
2. A key value pair data structure like Map

Spring JDBC

It is used to interact with the database, it provides you to configure the

1. DriverManagerDatasource: which will have properties like username, password, url, driverClassName
2. JdbcTemplate: which has property DataSource that helps JdbcTemplate to connect to the Database & it also gives you methods to perform crud operations like
   1. update(SQLquery, arguments): insert, update, delete queries
   2. query(SQLquery, arguments): select command for more than one records
   3. queryForObject(SQLquery, arguments): select command for one record

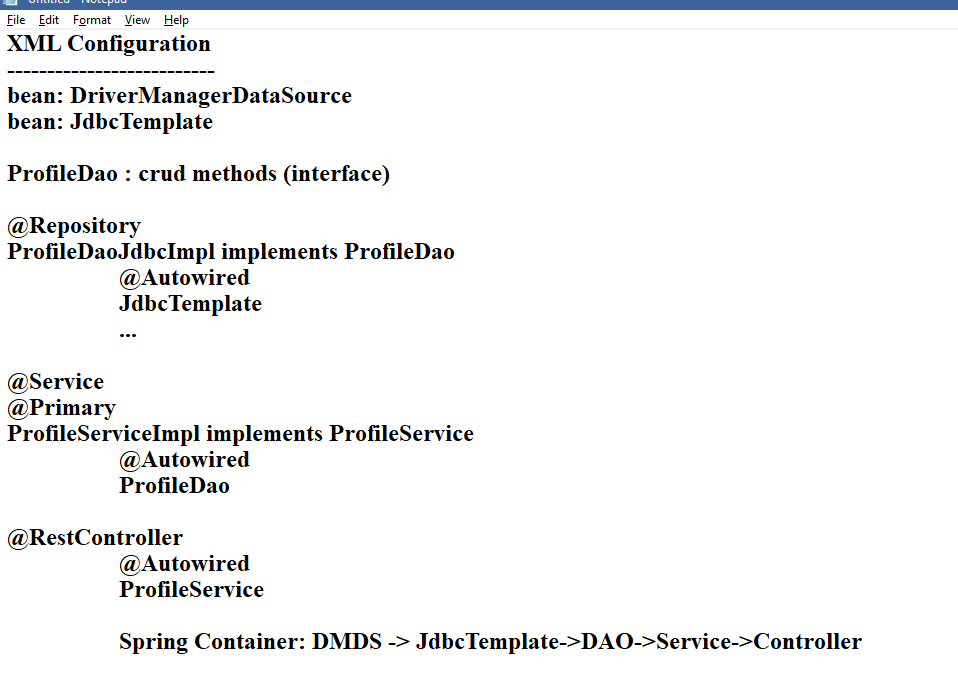
Limitations of JDBC

* You need to take care of writing SQL queries
* You need to take care of converting Java types to SQL types
* Writing complex join queries would be difficult when you want to get a result from multiple tables



Libraries we need are:

1. Spring Jdbc: spring-jdbc
2. Database driver library : derby-client



Spring ORM

It helps you to directly map java objects to the tables

ORM takes care of following things

1. Type conversion like java types to sql types
2. Mapping results to Java objects or Collections
3. No need to write queries
4. ORM generates the queries depending on the database the application is connected to
5. Complex joining queries are achieved with annotations

Spring ORM provides HibernateTemplate which performs all the CRUD operations for you, it will automatically generate SQL queries

HibernateTemplate depends on SessionFactory which depends on DriverManagerDataSource

Libraries required

1. Spring ORM
2. Hibernate Core
3. Derby Client

XML configuration

Configure tx namespace,

Configure DriverManagerDataSource

Configure LocalSessionFactoryBean

Configure HibernateTemplate

Configure TransactionManager