Spring FrameWork Notes

Spring is a lightweight framework.

Application Context:Spring ApplicationContext is where Spring holds instances of objects that it has identified to be managed

and distributed automatically. These are called beans.Using the Inversion of Control principle, Spring collects bean instances from our application and uses them at the appropriate time.

Inversion Of Control (IOC):

The main tasks performed by IoC container are:

to instantiate the application class

to configure the object

to assemble the dependencies between the objects

There are two types of IoC containers. They are:

BeanFactory-->org.springframework.beans.factory.BeanFactory(interface),The XmlBeanFactory is the implementation class for the BeanFactory interface.

    Resource resource=new ClassPathResource("applicationContext.xml");

    BeanFactory factory=new XmlBeanFactory(resource);

    ApplicationContext-->org.springframework.context.ApplicationContext(interface),It adds some extra functionality than BeanFactory.

    ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

In Spring framework, IOC container is responsible to inject the dependency.

We provide metadata to the IOC container either by XML file or annotation.

Dependency Injection:

Dependency Injection (DI) is a design pattern that removes the dependency from the programming code

In such case we provide the information from the external source such as XML file.

It makes our code loosely coupled and easier for testing.

Spring framework provides two ways to inject dependency:

By Constructor-->We can inject the dependency by constructor.

The <constructor-arg> subelement of <bean> is used for constructor injection where we can pass value and type of parameter.

By Setter method-->The <property> subelement of <bean> is used for setter injection.setter injection is flexible than constructor injection.

Autowiring in Spring:

Autowiring feature of spring framework enables you to inject the object dependency implicitly.

It internally uses setter or constructor injection.

Spring (AOP)Aspect Oriented Programming:

AOP is a programming paradigm that aims to increase modularity by allowing the separation of cross-cutting concerns.

1.Aspect:This is a module which has a set of APIs providing cross-cutting requirements. For example, a logging module would be called AOP aspect for logging.

An application can have any number of aspects depending on the requirement.

2.Join point:This represents a point in your application where you can plug-in the AOP aspect. You can also say, it is the actual place in the application where an action will be taken using Spring

AOP framework.

3.Advice: This is the actual action to be taken either before or after the method execution. This is an actual piece of code that is invoked during the program execution by Spring AOP framework.

4.Pointcut: This is a set of one or more join points where an advice should be executed. You can specify pointcuts using expressions or patterns as we will see in our AOP examples.

5.Introduction: An introduction allows you to add new methods or attributes to the existing classes.

6. Target object : The object being advised by one or more aspects. This object will always be a proxied object, also referred to as the advised object.

7. Weaving : Weaving is the process of linking aspects with other application types or objects to create an advised object. This can be done at compile time, load time, or at runtime.

Types of Advice:

1. before : Run advice before the a method execution.

2. after: Run advice after the method execution, regardless of its outcome.

3. after-returning : Run advice after the a method execution only if method completes successfully.

4. after-throwing: Run advice after the a method execution only if method exits by throwing an exception.

5.around : Run advice before and after the advised method is invoked.

JdbcTemplate Class:

The JDBC Template class executes SQL queries, updates statements, stores procedure calls, performs iteration over ResultSets, and extracts returned parameter values. It also catches JDBC exceptions and translates them to the generic, more informative, exception hierarchy defined in the org.springframework.dao package.

Transaction Management:

Atomicity − A transaction should be treated as a single unit of operation, which means either the entire sequence of operations is successful or unsuccessful.

Consistency − This represents the consistency of the referential integrity of the database, unique primary keys in tables, etc.

Isolation − There may be many transaction processing with the same data set at the same time. Each transaction should be isolated from others to prevent data corruption.

Durability − Once a transaction has completed, the results of this transaction have to be made permanent and cannot be erased from the database due to system failure.

Following are the possible values for isolation level

1.TransactionDefinition.ISOLATION_DEFAULT : This is the default isolation level.

2. TransactionDefinition.ISOLATION_READ_COMMITTED : Indicates that dirty reads are prevented; non-repeatable reads and phantom reads can occur.

3. TransactionDefinition.ISOLATION_READ_UNCOMMITTED : Indicates that dirty reads, non-repeatable reads, and phantom reads can occur.

4. TransactionDefinition.ISOLATION_REPEATABLE_READ : Indicates that dirty reads and non-repeatable reads are prevented; phantom reads can occur.

5. TransactionDefinition.ISOLATION_SERIALIZABLE : Indicates that dirty reads, non-repeatable reads, and phantom reads are prevented.

Spring mvc:

The Model encapsulates the application data and in general they will consist of POJO.

The View is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.

The Controller is responsible for processing user requests and building an appropriate model and passes it to the view for rendering.

The DispatcherServlet delegates the request to the controllers to execute the functionality specific to it.

The @Controller annotation indicates that a particular class serves the role of a controller.

The @RequestMapping annotation is used to map a URL to either an entire class or a particular handler method.

Annotation Based configuration:

1.@Required : The @Required annotation applies to bean property setter methods.

2. @Autowired : The @Autowired annotation can apply to bean property setter methods, non-setter methods, constructor and properties.

3. @Qualifier : The @Qualifier annotation along with @Autowired can be used to remove the confusion by specifiying which exact bean will be wired.

Bean Scope:

1. singleton : This scopes the bean definition to a single instance per Spring IoC container (default).

2. prototype : This scopes a single bean definition to have any number of object instances.

3. request : This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.

4. session : This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

5. global-session : This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

Bean Properties and Description:

1. class: This attribute is mandatory and specifies the bean class to be used to create the bean.

2. name : This attribute specifies the bean identifier uniquely. In XMLbased configuration metadata, you use the id and/or name attributes to specify the bean identifier(s).

3. scope : This attribute specifies the scope of the objects created from a particular bean definition.

4. constructor-arg: This is used to inject the dependencies.

5. properties : This is used to inject the dependencies.

6. autowiring mode : This is used to inject the dependencies.

7. lazy-initialization mode : A lazy-initialized bean tells the IoC container to create a bean instance when it is first requested, rather than at the startup.

8. initialization method : A callback to be called just after all necessary properties on the bean have been set by the container.

9. destruction method : A callback to be used when the container containing the bean is destroyed.