Let's begin with the first section of Spring interview questions that is the General Questions.

General Questions – Spring Interview Questions

1. What is a Spring Framework?

Spring is a powerful open source, application framework created to reduce the complexity of enterprise application development. It is light-weighted and loosely coupled. It has layered architecture, which allows you to select the components to use, while also providing a cohesive framework for J2EE application development. Spring framework is also called framework of frameworks as it provides support to various other frameworks such as Struts, Hibernate, Tapestry, EJB, JSF etc.

2. List the advantages of Spring Framework.

- Because of Spring Frameworks layered architecture, you can use what you need and leave which you don't.
- Spring Framework enables POJO (Plain Old Java Object) Programming which in turn enables continuous integration and testability.
- JDBC is simplified due to Dependency Injection and Inversion of Control.
- It is open-source and has no vendor lock-in.

3. What are the different features of Spring Framework?

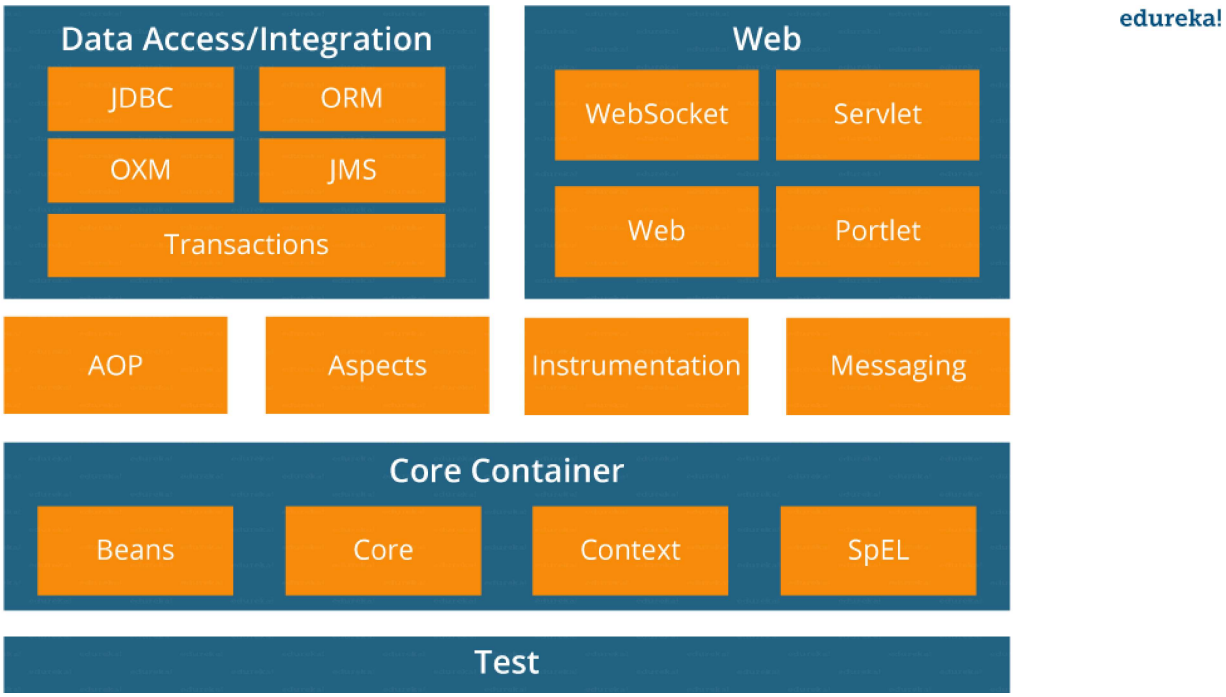
Following are some of the major features of Spring Framework :

- **Lightweight:** Spring is lightweight when it comes to size and transparency.
- **Inversion of control (IOC):** The objects give their dependencies instead of creating or looking for dependent objects. This is called Inversion Of Control.
- **Aspect oriented Programming (AOP):** Aspect oriented programming in Spring supports cohesive development by separating application business logic from system services.
- **Container:** Spring Framework creates and manages the life cycle and configuration of the application objects.
- **MVC Framework:** Spring Framework's MVC web application framework is highly configurable. Other frameworks can also be used easily instead of Spring MVC Framework.
- **Transaction Management:** Generic abstraction layer for transaction management is provided by the Spring Framework. Spring's transaction support can be also used in container less environments.
- **JDBC Exception Handling:** The JDBC abstraction layer of the Spring offers an exception

hierarchy, which simplifies the error handling strategy.

4. How many modules are there in Spring Framework and what are they?

There are around 20 modules which are generalized into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation and Test.



- **Spring Core Container –** This layer is basically the core of Spring Framework. It contains the following modules :

a. Spring Core
b. Spring Bean
c. SpEL (Spring Expression Language)
d. Spring Context

- **Data Access/Integration –** This layer provides support to interact with the database. It contains the following modules :

a. JDBC (Java DataBase Connectivity)
b. ORM (Object Relational Mapping)
c. OXM (Object XML Mappers)
d. JMS (Java Messaging Service)
e. Transaction

- **Web –** This layer provides support to create web application. It contains the following modules :

a. Web
b. Web – MVC
c. Web – Socket
d. Web – Portlet

- **Aspect Oriented Programming (AOP) –** In this layer you can use Advices, Pointcuts etc., to decouple the code.
- **Instrumentation –** This layer provides support to class instrumentation and classloader implementations.
- **Test –** This layer provides support to testing with JUnit and TestNG.

Few Miscellaneous modules are given below:

- **Messaging –** This module provides support for STOMP. It also supports an annotation programming model that is used for routing and processing STOMP messages from WebSocket clients.
- **Aspects –** This module provides support to integration with AspectJ.

5. What is a Spring configuration file?

A Spring configuration file is an XML file. This file mainly contains the classes information. It describes how those classes are configured as well as introduced to each other. The XML configuration files, however, are verbose and more clean. If it's not planned and written correctly, it becomes very difficult to manage in big projects.

6. What are the different components of a Spring application?

A Spring application, generally consists of following components:

- Interface: It defines the functions.
- Bean class: It contains properties, its setter and getter methods, functions etc.
- Spring Aspect Oriented Programming (AOP): Provides the functionality of cross-cutting concerns.
- Bean Configuration File: Contains the information of classes and how to configure them.
- User program: It uses the function.

7. What are the major features in different versions of Spring?

Since the release of Spring Framework back in the 2004, many Spring versions came out in the market. But few of them had some major additions like:

Spring 2.5: This version was released in 2007. It was the first version which supported annotations.

Spring 3.0: This version was released in 2009. It made full fledged use of improvements in Java5 and also provided support to JEE6.

Spring 4.0: This version was released in 2013. This was the first version to provide full support to Java 8.

8. What are the various ways of using Spring Framework?

Spring Framework can be used in various ways. They are listed as follows:

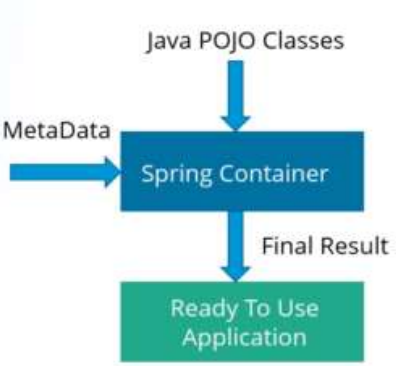
1. As a Full-fledged Spring web application.
2. As a third-party web framework, using Spring Frameworks middle-tier.
3. For remote usage.
4. As Enterprise Java Bean which can wrap existing POJOs (Plain Old Java Objects).

The next section of Spring Interview Questions is on *Dependency Injection and IoC container*.

Dependency Injection/ IoC Container – Spring Interview Questions

9. What is Spring IOC Container?

At the core of the Spring Framework, lies the Spring container. The container creates the object, wires them together, configures them and manages their complete life cycle. The Spring container makes use of Dependency Injection to manage the components that make up an application. The container receives instructions for which objects to instantiate, configure, and assemble by reading the configuration metadata provided. This metadata can be provided either by XML, Java annotations or Java code.



10. What do you mean by Dependency Injection?

In Dependency Injection, you do not have to create your objects but have to describe how they should be created. You don't connect your components and services together in the code directly, but describe which services are needed by which components in the configuration file. The IoC container will wire them up together.

11. In how many ways can Dependency Injection be done?

In general, dependency injection can be done in three ways, namely :

- Constructor Injection
- Setter Injection
- Interface Injection

In Spring Framework, only constructor and setter injections are used.

12. Differentiate between constructor injection and setter injection.

Constructor Injection vs Setter Injection

Constructor Injection	Setter Injection
There is no partial injection.	There can be partial injection.
It doesn't override the setter property.	It overrides the constructor property.
It will create a new instance if any modification is done.	It will not create new instance if any modification is done.
It works better for many properties.	It works better for few properties.

13. How many types of IOC containers are there in spring?

a. **BeanFactory**: BeanFactory is like a factory class that contains a collection of beans. It instantiates the bean whenever asked for by clients.
b. **ApplicationContext**: The ApplicationContext interface is built on top of the BeanFactory interface. It provides some extra functionality on top BeanFactory.

14. Differentiate between BeanFactory and ApplicationContext.

BeanFactory vs ApplicationContext

BeanFactory	ApplicationContext
It is an interface defined in org.springframework.beans.factory.**BeanFactory**	It is an interface defined in org.springframework.context.**ApplicationContext**
It uses Lazy initialization	It uses Eager/ Aggressive initialization
It explicitly provides a resource object using the syntax	It creates and manages resource objects on its own
It doesn't supports internationalization	It supports internationalization
It doesn't supports annotation based dependency	It supports annotation based dependency

15. List some of the benefits of IoC.

Some of the benefits of IoC are:

- It will minimize the amount of code in your application.
- It will make your application easy to test because it doesn't require any singletons or JNDI lookup mechanisms in your unit test cases.
- It promotes loose coupling with minimal effort and least intrusive mechanism.
- It supports eager instantiation and lazy loading of the services.

Let's move on to the next section of Spring Interview Questions, that is *Spring Beans Interview Questions*.

Spring Beans – Spring Interview Questions

16. Explain Spring Beans?

- They are the objects that form the backbone of the user's application.
- Beans are managed by the Spring IoC container.
- They are instantiated, configured, wired and managed by a Spring IoC container
- Beans are created with the configuration metadata that the users supply to the container.

17. How configuration metadata is provided to the Spring container?

Configuration metadata can be provided to Spring container in following ways:

- **XML-Based configuration:** In Spring Framework, the dependencies and the services needed by beans are specified in configuration files which are in XML format. These configuration files usually contain a lot of bean definitions and application specific configuration options. They generally start with a bean tag. For example:

```
<bean id="studentbean" class="org.edureka.firstSpring.StudentBean">
 <property name="name" value="Edureka"></property>
</bean>
```

- **Annotation-Based configuration**: Instead of using XML to describe a bean wiring, you can

configure the bean into the component class itself by using annotations on the relevant class, method, or field declaration. By default, annotation wiring is not turned on in the Spring container. So, you need to enable it in your Spring configuration file before using it. For example:

```
<beans>
<context:annotation-config/>
<!-- bean definitions go here -->
</beans>
```

- **Java-based configuration:** The key features in Spring Framework's new Java-configuration support are @Configuration annotated classes and @Bean annotated methods.

    1. @Bean annotation plays the same role as the <bean/> element.

    2.@Configuration classes allows to define inter-bean dependencies by simply calling other @Bean methods in the same class.

For example:

```
@Configuration
public class StudentConfig
{
@Bean
public StudentBean myStudent()
{ return new StudentBean(); }
}
```

18. How many bean scopes are supported by Spring?

The Spring Framework supports five scopes. They are:

- **Singleton:** This provides scope for the bean definition to single instance per Spring IoC container.
- **Prototype:** This provides scope for a single bean definition to have any number of object instances.
- **Request:** This provides scope for a bean definition to an HTTP-request.
- **Session:** This provides scope for a bean definition to an HTTP-session.
- **Global-session:** This provides scope for a bean definition to an Global HTTP-session.

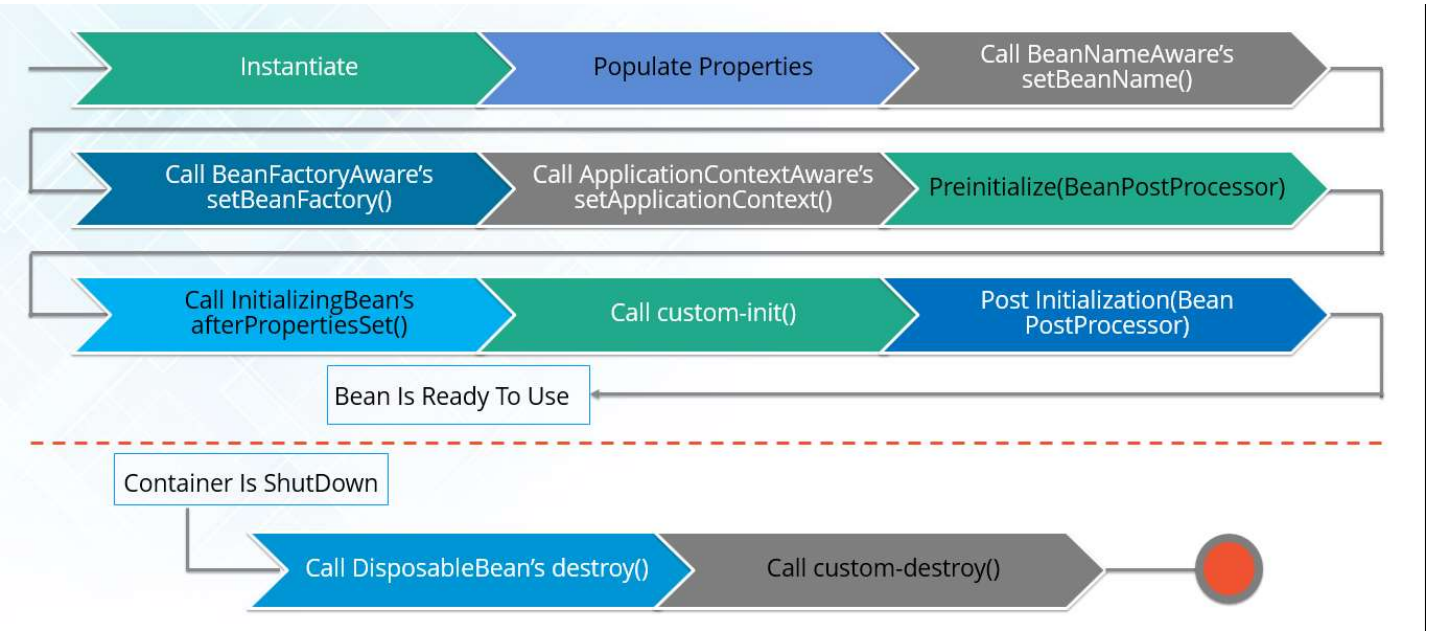
The last three are available only if the users use a web-aware ApplicationContext.

19. What is the Bean life cycle in Spring Bean Factory Container?

Bean life cycle in Spring Bean Factory Container is as follows:

1. The Spring container instantiates the bean from the bean's definition in the XML file.
2. Spring populates all of the properties using the dependency injection, as specified in the bean definition.
3. The factory calls setBeanName() by passing the bean's ID, if the bean implements the BeanNameAware interface.
4. The factory calls setBeanFactory() by passing an instance of itself, if the bean implements the BeanFactoryAware interface.
5. preProcessBeforeInitialization() methods are called if there are any BeanPostProcessors associated with the bean.
6. If an init-method is specified for the bean, then it will be called.
7. Finally, postProcessAfterInitialization() methods will be called if there are any BeanPostProcessors associated with the bean.

To understand it in better way check the below diagram:



20. Explain inner beans in Spring.

A bean can be declared as an inner bean only when it is used as a property of another bean. For defining a bean, the Spring's XML based configuration metadata provides the use of <bean> element inside the <property> or <constructor-arg>. Inner beans are always anonymous and they are always scoped as prototypes. For example, let's say we have one Student class having reference of Person class. Here we will be creating only one instance of Person class and use it inside Student.

Here's a Student class followed by bean configuration file:

Student.java

```
public class Student
{
private Person person;
//Setters and Getters
}
public class Person
{
private String name;
private String address;
//Setters and Getters
}
```

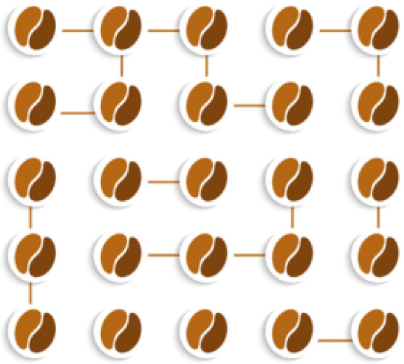
studentbean.xml

```
<bean id="StudentBean" class="com.edureka.Student">
<property name="person">
<!--This is inner bean -->
<bean class="com.edureka.Person">
<property name="name" value="Scott"></property>
<property name="address" value="Bangalore"></property>
</bean>
</property>
</bean>
```

21. Define Bean Wiring.

When beans are combined together within the Spring container, it's called wiring or bean wiring. The Spring container needs to know what beans are needed and how the container should use dependency

injection to tie the beans together, while wiring beans.



22. What do you understand by auto wiring and name the different modes of it?

The Spring container is able to autowire relationships between the collaborating beans. That is, it is possible to let Spring resolve collaborators for your bean automatically by inspecting the contents of the BeanFactory.

Different modes of bean auto-wiring are:

a. **no:** This is default setting which means no autowiring. Explicit bean reference should be used for wiring.
b. **byName:** It injects the object dependency according to name of the bean. It matches and wires its properties with the beans defined by the same names in the XML file.
c. **byType:** It injects the object dependency according to type. It matches and wires a property if its type matches with exactly one of the beans name in XML file.
d. **constructor:** It injects the dependency by calling the constructor of the class. It has a large number of parameters.
e. **autodetect:** First the container tries to wire using autowire by *constructor*, if it can't then it tries to autowire by *byType*.

23. What are the limitations with auto wiring?

Following are some of the limitations you might face with auto wiring:

- **Overriding possibility:** You can always specify dependencies using <constructor-arg> and <property> settings which will override autowiring.
- **Primitive data type:** Simple properties such as primitives, Strings and Classes can't be autowired.
- **Confusing nature:** Always prefer using explicit wiring because autowiring is less precise.

In the next section, we will discuss on *Spring Annotations Interview Questions*.

Spring Annotations – Spring Interview Questions

24. What do you mean by Annotation-based container configuration?

Instead of using XML to describe a bean wiring, the developer moves the configuration into the component class itself by using annotations on the relevant class, method, or field declaration. It acts as an alternative to XML setups. For example:

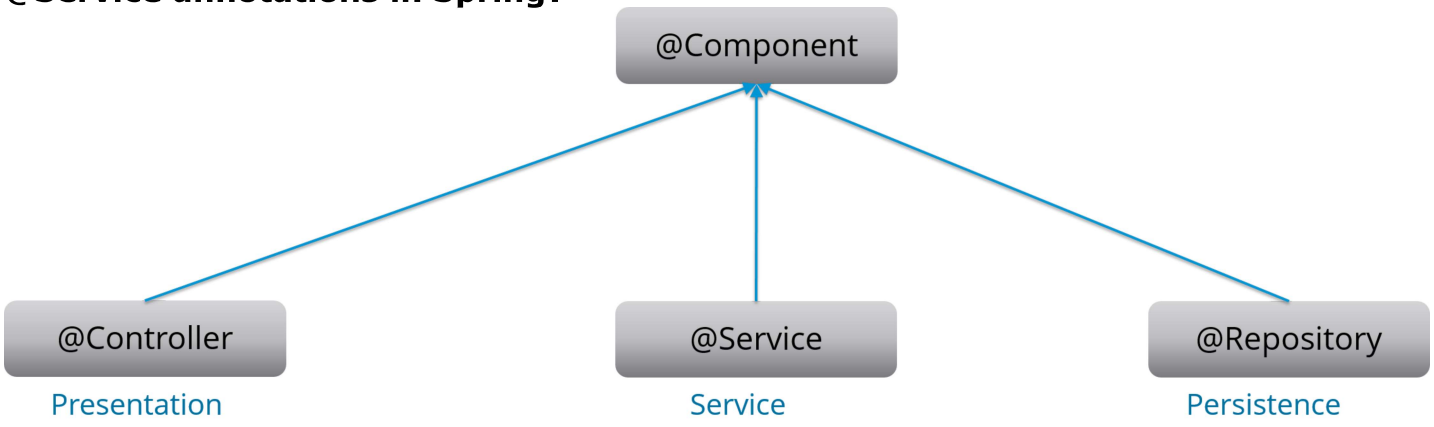
```
@Configuration
public class AnnotationConfig
{
@Bean
public MyDemo myDemo()
 { return new MyDemoImpll(); }
}
```

25. How annotation wiring can be turned on in Spring?

By default, Annotation wiring is not turned on in the Spring container. Thus, to use annotation based wiring we must enable it in our Spring configuration file by configuring **<context:annotation-config/>** element. For example:

```
<beans xmlns="http://www.springframework.org/schema/beans (http://www.springframework.org/schema/
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance (http://www.w3.org/2001/XMLSchema-instance)"
xmlns:context="http://www.springframework.org/schema/context (http://www.springframework.org/sche
<context:annotation-config/>
<beans ..... />
</beans>
```

26. What's the difference between @Component, @Controller, @Repository & @Service annotations in Spring?



@Component: This marks a java class as a bean. It is a generic stereotype for any Spring-managed component. The component-scanning mechanism of spring now can pick it up and pull it into the application context.

@Controller: This marks a class as a Spring Web MVC controller. Beans marked with it are automatically imported into the Dependency Injection container.

@Service: This annotation is a specialization of the component annotation. It doesn't provide any additional behavior over the @Component annotation. You can use @Service over @Component in service-layer classes as it specifies intent in a better way.

@Repository: This annotation is a specialization of the @Component annotation with similar use and functionality. It provides additional benefits specifically for DAOs. It imports the DAOs into the DI container and makes the unchecked exceptions eligible for translation into Spring DataAccessException.

27. What do you understand by @Required annotation?

@Required is applied to bean property setter methods. This annotation simply indicates that the

affected bean property must be populated at the configuration time with the help of an explicit property value in a bean definition or with autowiring. If the affected bean property has not been populated, the container will throw BeanInitializationException.

For example:

```
public class Employee
{
private String name;
@Required
public void setName(String name)
{this.name=name; }
public string getName()
{ return name; }
}
```

28. What do you understand by @Autowired annotation?

The **@Autowired** annotation provides more accurate control over where and how autowiring should be done. This annotation is used to autowire bean on the setter methods, constructor, a property or methods with arbitrary names or multiple arguments. By default, it is a type driven injection.

For Example:

```
public class Employee
{
private String name;
@Autowired
public void setName(String name)
{this.name=name; }
public string getName()
{ return name; }
}
```

29. What do you understand by @Qualifier annotation?

When you create more than one bean of the same type and want to wire only one of them with a property you can use the **@Qualifier** annotation along with **@Autowired** to remove the ambiguity by specifying which exact bean should be wired.

For example, here we have two classes, Employee and EmpAccount respectively. In EmpAccount, using @Qualifier its specified that bean with id emp1 must be wired.

Employee.java

```
public class Employee
{
private String name;
@Autowired
public void setName(String name)
{ this.name=name; }
public string getName()
{ return name; }
}
```

EmpAccount.java

```
public class EmpAccount
{
private Employee emp;
```

```
4  @Autowired
5  @Qualifier(emp1)
6  public void showName()
7  {
8  System.out.println("Employee name : "+emp.getName);
9  }
10 }
```

30. What do you understand by @RequestMapping annotation?

@RequestMapping annotation is used for mapping a particular HTTP request method to a specific class/ method in controller that will be handling the respective request. This annotation can be applied at both levels:

- **Class level** : Maps the URL of the request
- **Method level**: Maps the URL as well as HTTP request method

Next section of Spring Interview Questions is on *Data Access*.

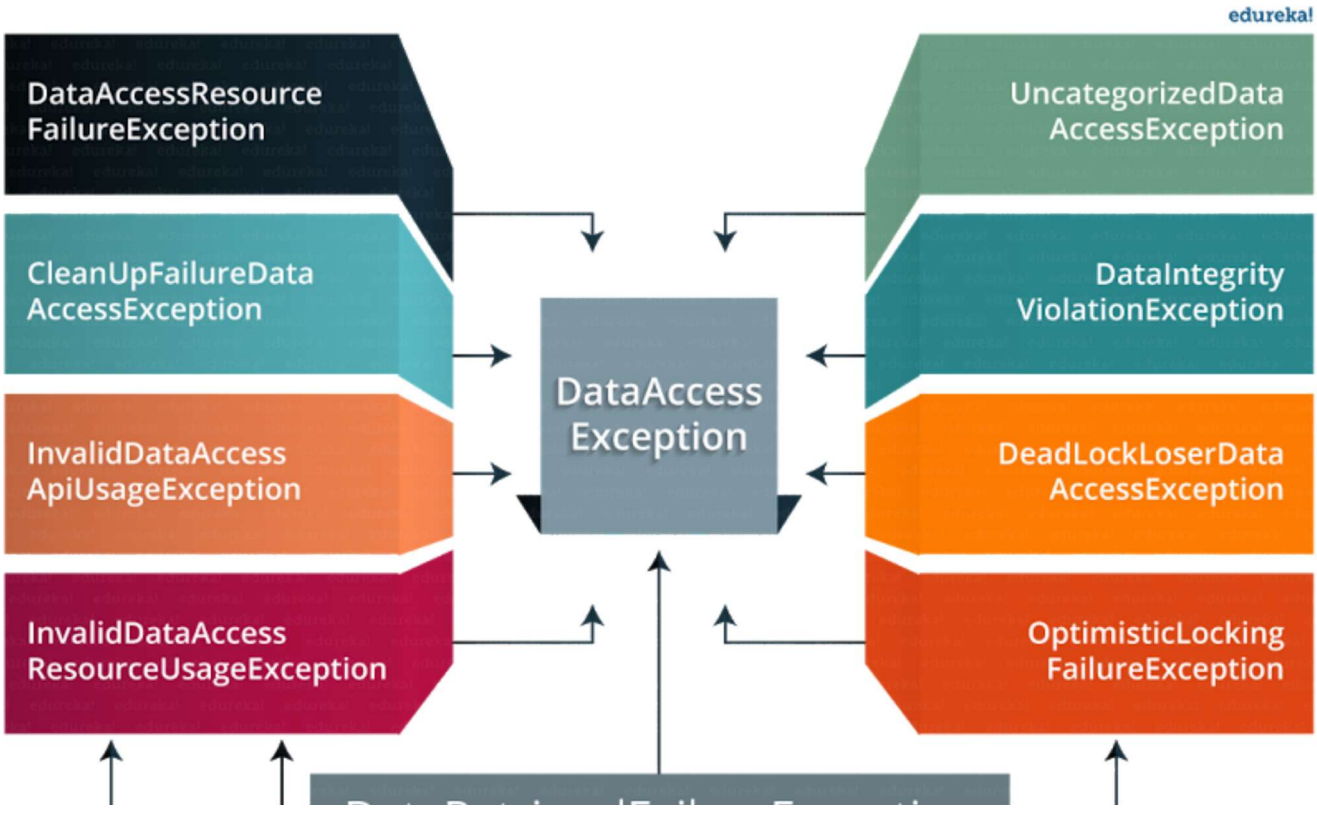
Data Access – Spring Interview Questions

31. Describe Spring DAO support?

The Data Access Object (DAO) support in Spring makes it easy to work with data access technologies like JDBC, Hibernate or JDO in a consistent way. This allows one to switch between the persistence technologies easily. It also allows you to code without worrying about catching exceptions that are specific to each of these technology.

32. Name the exceptions thrown by the Spring DAO classes.

See the below diagram, it depicts all the Spring DAO classes in the hierarchical order.



33. Which classes are present in spring JDBC API?

Classes present in JDBC API are as follows:

a. JdbcTemplate
b. SimpleJdbcTemplate
c. NamedParameterJdbcTemplate
d. SimpleJdbcInsert
e. SimpleJdbcCall

34. What are the ways by which Hibernate can be accessed using Spring?

There are two ways by which we can access Hibernate using Spring:

a. Inversion of Control with a Hibernate Template and Callback
b. Extending HibernateDAOSupport and Applying an AOP Interceptor node

35. Name the types of transaction management that Spring supports.

Two types of transaction management are supported by Spring. They are:

a. **Programmatic transaction management:** In this, the transaction is managed with the help of programming. It provides you extreme flexibility, but it is very difficult to maintain.
b. **Declarative transaction management:** In this, the transaction management is separated from the business code. Only annotations or XML based configurations are used to manage the transactions.

36. What are the different ORM's supported by Spring?

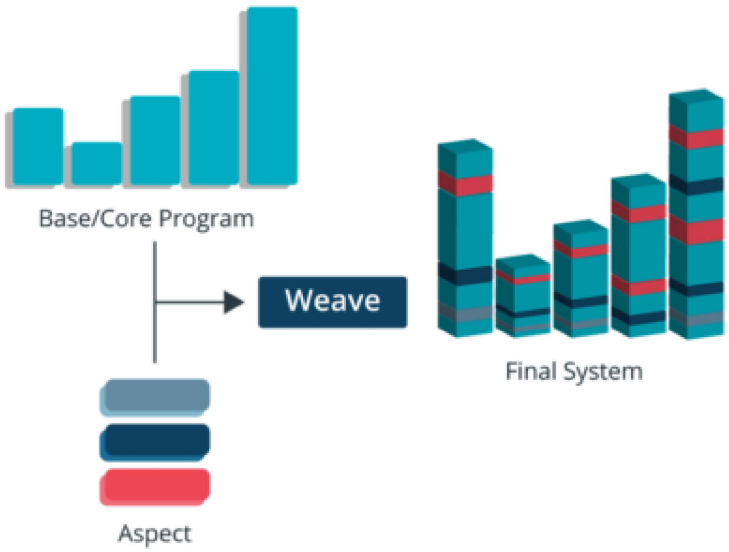
Different ORM's supported by Spring are depicted via the below diagram:

The next section of Spring interview questions discusses on *Spring AOP Interview Questions*.

Aspect Oriented Programming (AOP) – Spring Interview Questions

37. Describe AOP.

Aspect-oriented programming or AOP is a programming technique which allows programmers to modularize crosscutting concerns or behavior that cuts across the typical divisions of responsibility. Examples of cross-cutting concerns can be logging and transaction management. The core of AOP is an *aspect*. It encapsulates behaviors that can affect multiple classes into reusable modules.

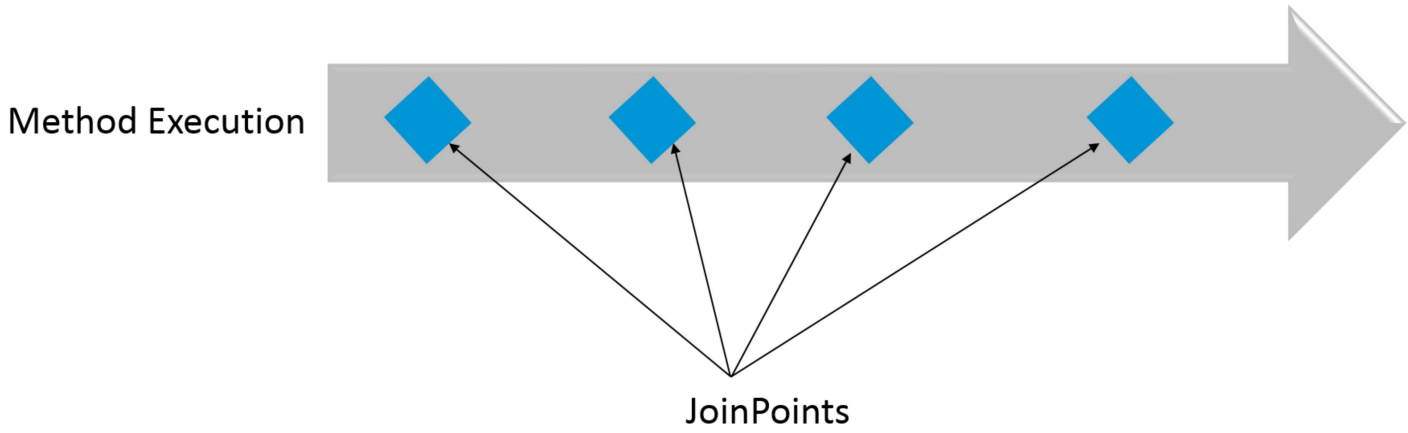


38. What do you mean by Aspect?

Aspect is a modularization of concern which cuts across multiple objects. Transaction management is a good example of a crosscutting concern in J2EE applications. Aspects are implemented using regular classes or regular classes annotated with the @Aspect annotation in Spring Framework.

39. Explain JoinPoint.

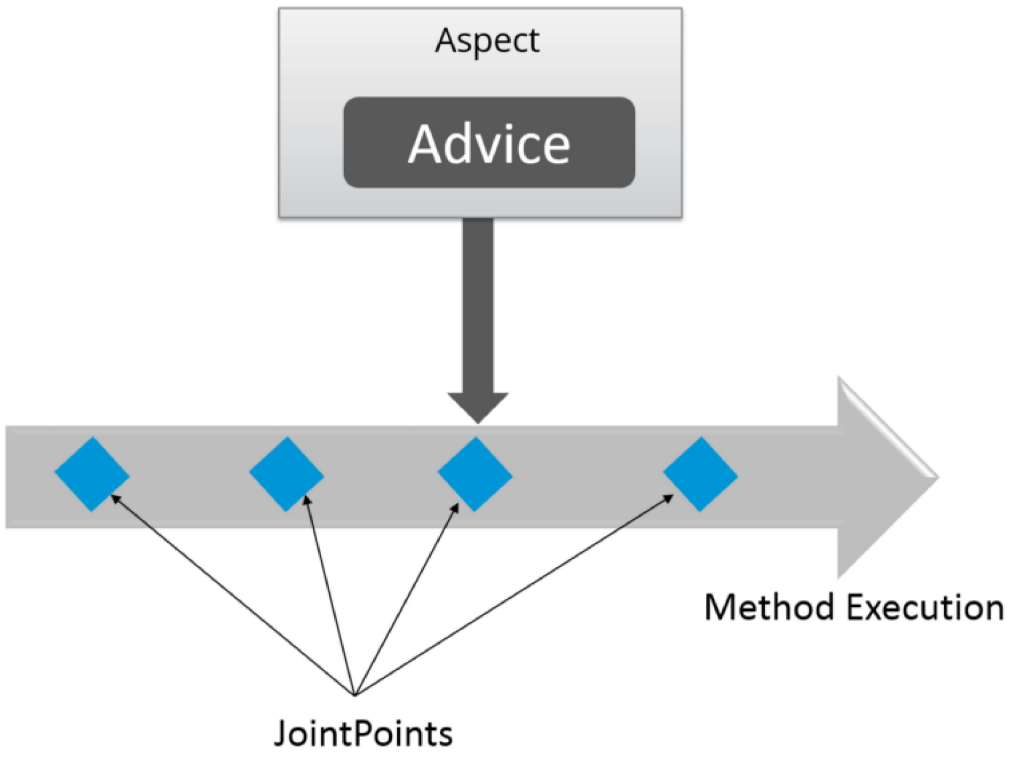
A point during the execution of a program is called JoinPoint, such as the execution of a method or the handling of an exception. In Spring AOP, a joinpoint always represents a method execution.



40. What is an Advice?

An Action taken by an aspect at a particular joinpoint is known as an Advice. Spring AOP uses an advice

An Action taken by an aspect at a particular joinpoint is known as an Advice. Spring AOP uses an advice as an interceptor, maintaining a chain of interceptors "around" the join point.



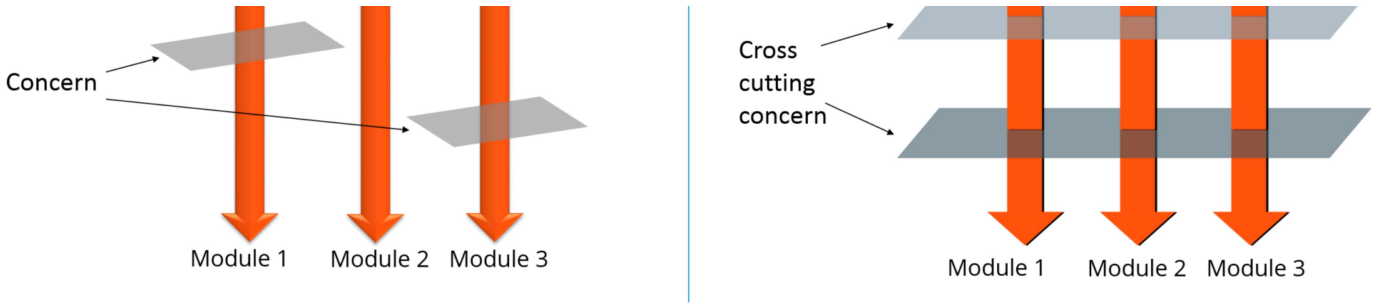
41. What are the different types of Advices?

a. **Before:** These types of advices execute before the joinpoint methods and are configured using **@Before** annotation mark.
b. **After returning:** These types of advices execute after the joinpoint methods completes executing normally and are configured using @AfterReturning annotation mark.
c. **After throwing:** These types of advices execute only if joinpoint method exits by throwing an exception and are configured using @AfterThrowing annotation mark.
d. **After (finally):** These types of advices execute after a joinpoint method, regardless of the method's exit whether normally or exceptional return and are configured using @After annotation mark.
e. **Around:** These types of advices execute before and after a joinpoint and are configured using @Around annotation mark.

42. Point out the difference between concern and cross-cutting concern in Spring AOP?

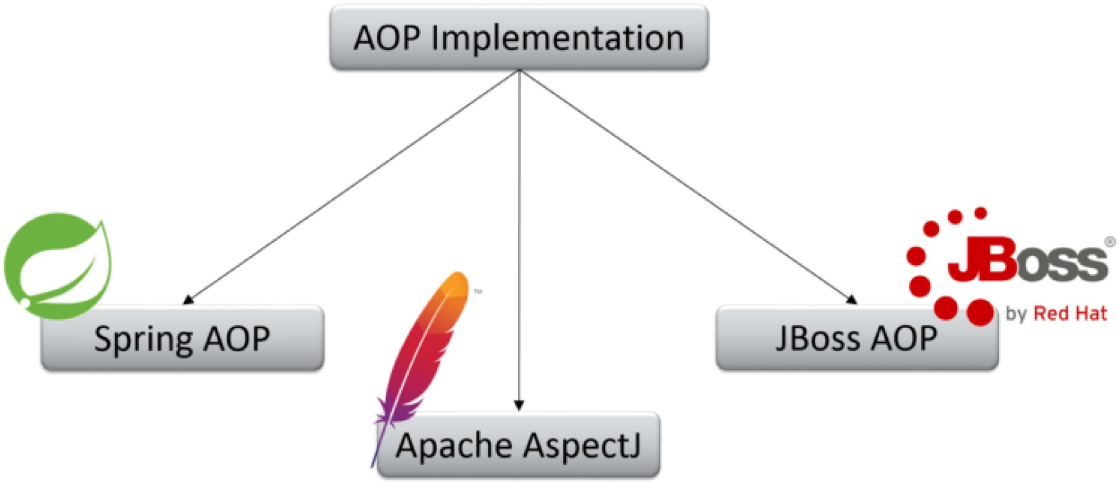
The concern is the behavior we want to have in a particular module of an application. It can be defined as a functionality we want to implement.

The cross-cutting concern is a concern which is applicable throughout the application. This affects the entire application. For example, logging, security and data transfer are the concerns needed in almost every module of an application, thus they are the cross-cutting concerns.



43. What are the different AOP implementations?

Different AOP implementations are depicted by the below diagram:



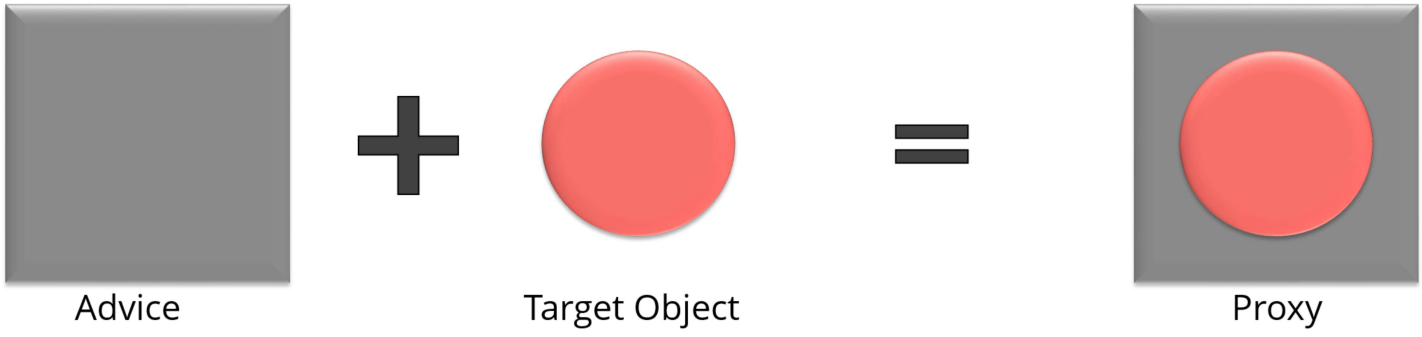
44. What are the difference between Spring AOP and AspectJ AOP?

Spring AOP vs AspectJ AOP

Spring AOP	AspectJ AOP
Runtime weaving through proxy is done	Compile time weaving through AspectJ Java tools is done
It supports only method level PointCut	It suports field level Pointcuts
It is DTD based	It is schema based and Annotation configuration

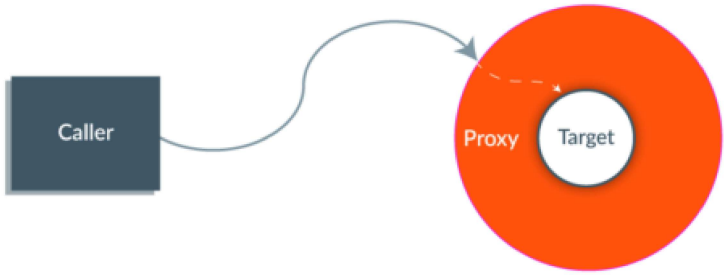
45. What do you mean by Proxy in Spring Framework?

An object which is created after applying advice to a target object is known as a Proxy. In case of client objects the target object and the proxy object are the same.



46. In Spring, what is Weaving?

The process of linking an aspect with other application types or objects to create an advised object is called Weaving. In Spring AOP, weaving is performed at runtime. Refer the below diagram:



The last section of Spring interview questions is on *Spring MVC Interview Questions*.

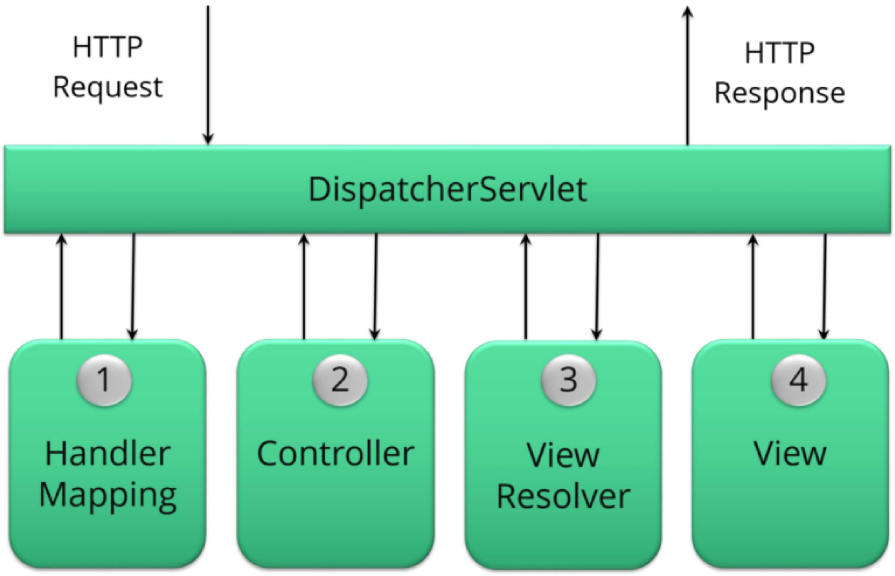
MVC (Model-View-Controller) – Spring Interview Questions

47. What do you mean by Spring MVC framework?

The Spring web MVC framework provides model-view-controller architecture and ready to use components that are used to develop flexible and loosely coupled web applications. The MVC pattern helps in separating the different aspects of the application like input logic, business logic and UI logic, while providing a loose coupling between all these elements.

48. Describe DispatcherServlet.

The DispatcherServlet is the core of Spring Web MVC framework. It handles all the HTTP requests and responses. The DispatcherServlet receives the entry of handler mapping from the configuration file and forwards the request to the controller. The controller then returns an object of Model And View. The DispatcherServlet checks the entry of view resolver in the configuration file and calls the specified view component.

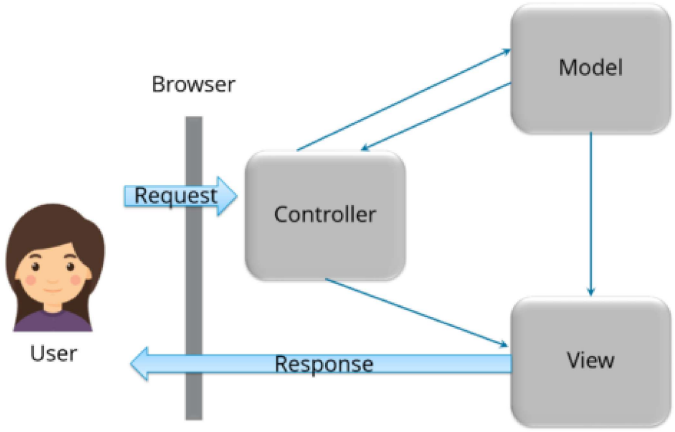


49. Explain WebApplicationContext.

The WebApplicationContext is an extension of the plain ApplicationContext. It has some extra features that are necessary for web applications. It differs from a normal ApplicationContext in terms of its capability of resolving themes and in deciding which servlet it is associated with.

50. In Spring MVC framework, what is controller?

Controllers provide access to the application behavior. These behaviors are generally defined through a service interface. Controllers interpret the user input and transform it into a model which is represented to the user by the view. In Spring, controller is implemented in a very abstract way. It also enables you to create a wide variety of controllers.



I hope this set of Spring Interview Questions and Answers will help you in preparing for your interviews. All the best!

Master Spring Framework

(https://www.edureka.co/spring-framework)

If you want to learn Spring and wish to use it while developing Java applications, then check out the ***Spring Framework Certification Training*** *(https://www.edureka.co/spring-framework) by Edureka, a trusted online learning company with a network of more than 250,000 satisfied learners spread across the globe.*

Got a question for us? Please mention it in the comments section and we will get back to you.

About Swatee Chand (15 Posts (https://www.edureka.co/blog/author/swatee-chandedureka-co/))

Research Analyst at Edureka. A techno freak who likes to explore different technologies. Likes to follow the technology trends in market and write about them.