**1. What is a Spring Framework?**

Spring is a 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.****

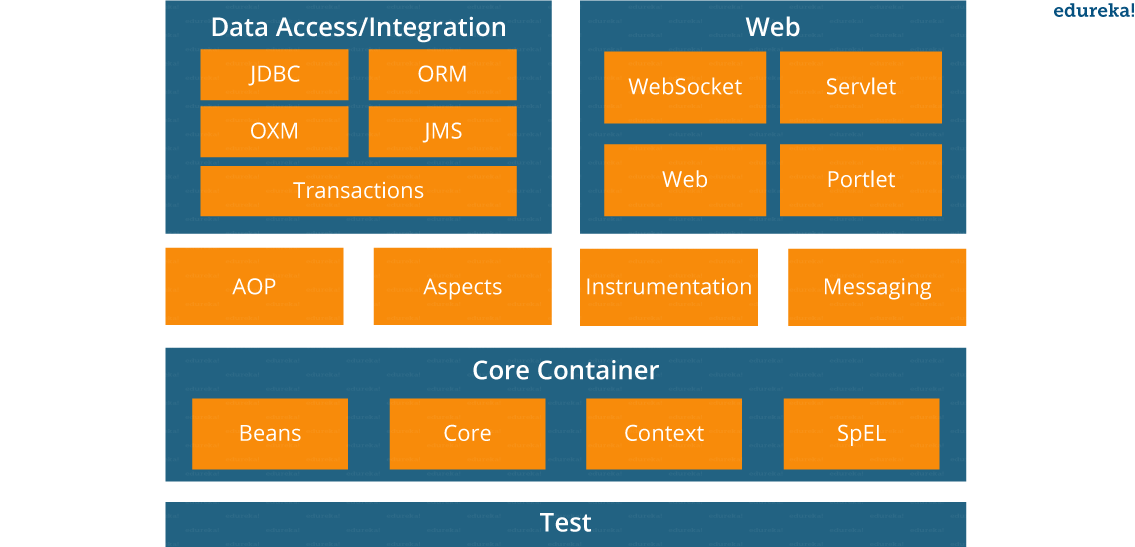
* Because of Spring Frameworks layered architecture, you can use what you need and leave which you don’t.
* Spring Framework enables POJO (Plain Old Java Object) Programming which in turn enables continuous integration and testability.
* JDBC is simplified due to Dependency Injection and Inversion of 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 :

Spring Core, Spring Bean, SpEL (Spring Expression Language) ,Spring Context

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

JDBC (Java DataBase Connectivity) , ORM (Object Relational Mapping) , OXM (Object XML Mappers) , JMS (Java Messaging Service) ,Transaction

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

Web , Web – MVC , Web – Socket , Web – Portlet

* **Aspect Oriented Programming (AOP) –** In this layer you can use Advices, Pointcuts etc., to decouplethe 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?****

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 cleaner. 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?****

* 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?****

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

**Spring 3.0:** in 2009. Itmade 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).

### ****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?****

* 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** | **Setter Injection** |
| There is no partial injection. | There can be partial injection. |
| It doesn’t override the setter property. | It overrides the constructor property. |
| create a new instance if any modification is done. | 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?****

1. **BeanFactory**: BeanFactory is like a factory class that contains a collection of beans. It instantiates the bean whenever asked for by clients.
2. **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.****

* 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.

### ****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?****

* **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

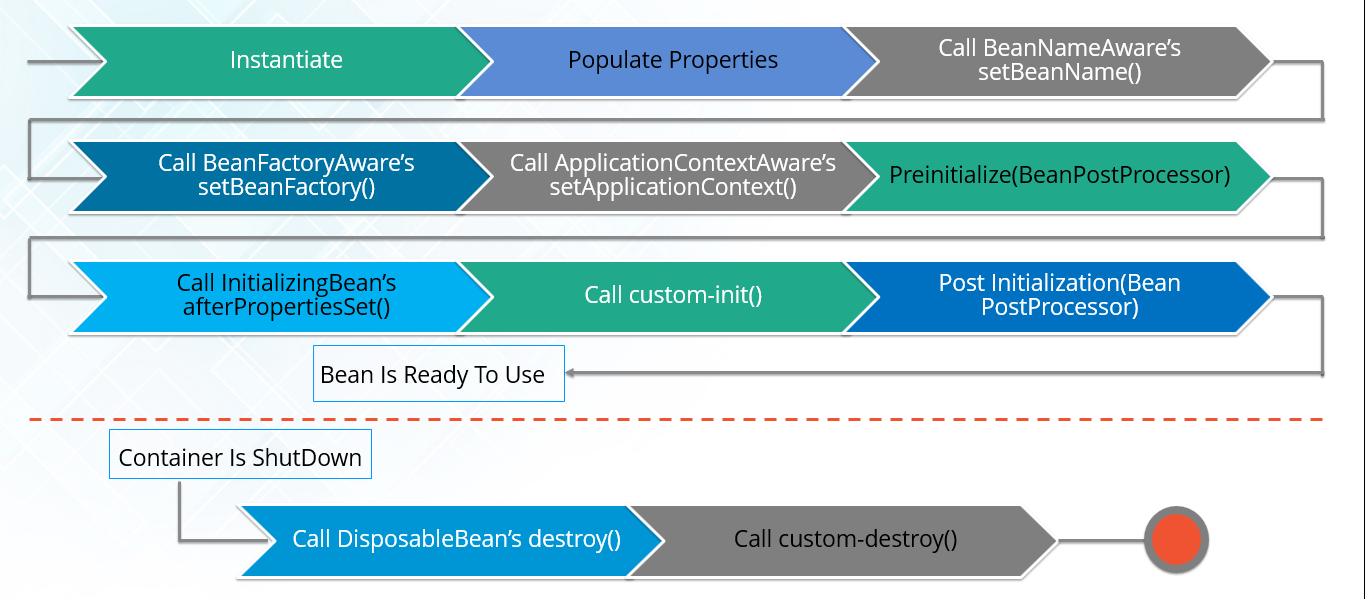
|  |  |
| --- | --- |
| 7 | @Configuration  public class StudentConfig  { @Bean  public StudentBean myStudent()  { return new StudentBean(); }} |

### ****18. How many bean scopes are supported by Spring?****

* **Singleton:** single instance per Spring IoC container.
* **Prototype:** single bean definition to have any number of object instances.
* **Request:** 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.

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

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.



### ****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.

|  |  |
| --- | --- |
| Student.Java  public class Student  {  private Person person;  //Setters and Getters}  public class Person  {  private String name;  private String address;  //Setters and Getters} | Student.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:

1. **no:** This is default setting which means no autowiring. Explicit bean reference should be used for wiring.
2. **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.
3. **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.
4. **constructor:**It injects the dependency by calling the constructor of the class. It has a large number of parameters.
5. **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?****

* **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.

### ****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.

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

### ****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 andmakes 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.

|  |  |
| --- | --- |
|  | 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.

|  |  |
| --- | --- |
|  | 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;  @Autowired  @Qualifier(emp1)  public void showName(){  System.out.println(“Employee name : ”+emp.getName);}} |

### ****30.  What do you understand by @RequestMapping annotation?****

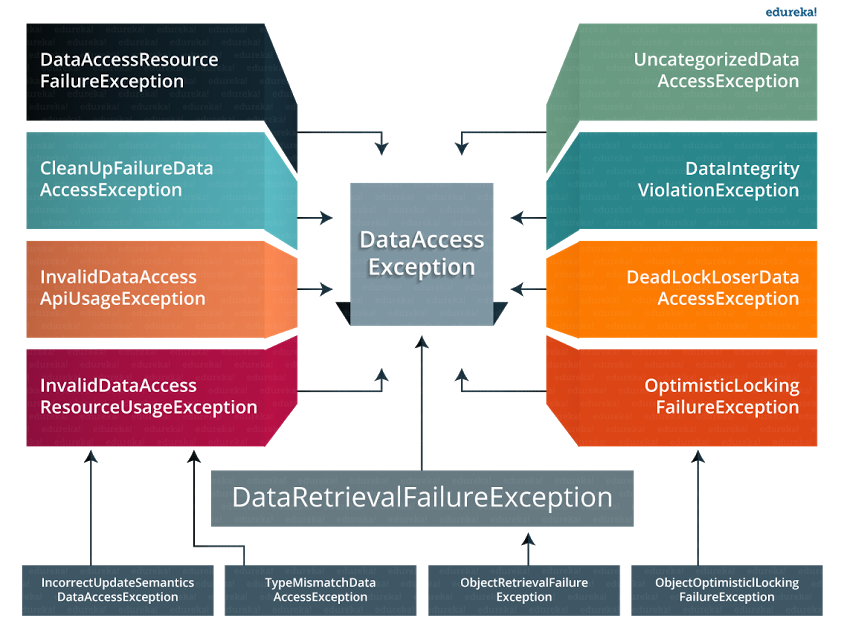
@RequestMapping annotation is used for mapping a particular HTTP request method to a specific class/ method in controller that will be handling the respective request. This annotation can be applied at both levels:

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

### ****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.

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



### ****33.  Which classes are present in spring JDBC API?****

Classes present in JDBC API are as follows:

1. JdbcTemplate
2. SimpleJdbcTemplate
3. NamedParameterJdbcTemplate
4. SimpleJdbcInsert
5. 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:

1. Inversion of Control with a Hibernate Template and Callback
2. 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:

1. **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.
2. **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:

1. JPA
2. Oracle toplink
3. JDO
4. hibernate
5. ibatis
6. OJB

### ****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 as an interceptor, maintaining a chain of interceptors “around” the join point.

### ****41. What are the different types of Advices?****

1. **Before:** These types of advices execute before the joinpoint methods and are configured using **@Before**annotation mark.
2. **After returning:**These types of advices execute after the joinpoint methods completes executing normally and are configured using @AfterReturning annotation mark.
3. **After throwing:** These types of advices execute only if joinpoint method exits by throwing an exception and are configured using @AfterThrowing annotation mark.
4. **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.
5. **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.

### ****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:

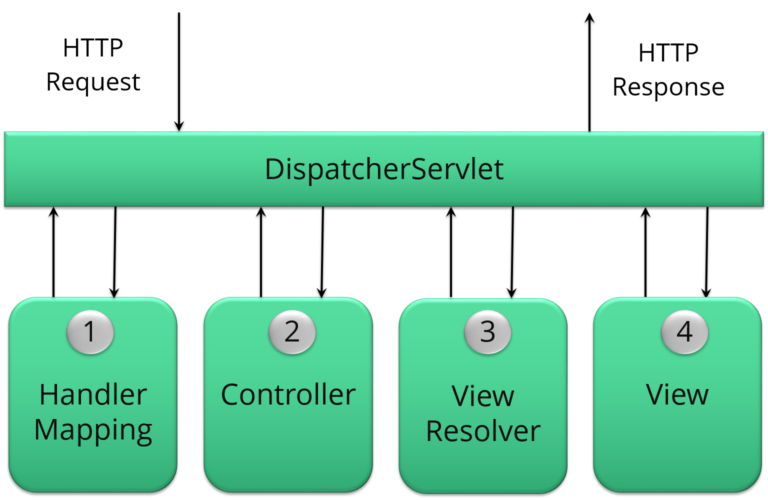
### 

### ****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.

* 1. **What are the benefits of using Spring Tool Suite?**

We can install plugins into Eclipse to get all the features of Spring Tool Suite. However STS comes with Eclipse with some other important stuffs such as Maven support, Templates for creating different types of Spring projects and tc server for better performance with Spring applications.

I like STS because it highlights the Spring components and if you are using AOP pointcuts and advices, then it clearly shows which methods will come under the specific pointcut. So rather than installing everything on our own, I prefer using STS when developing Spring based applications.

1. **Name some of the important Spring Modules?**

Some of the important Spring Framework modules are:

* + **Spring Context** – for dependency injection.
  + **Spring AOP** – for aspect oriented programming.
  + **Spring DAO** – for database operations using DAO pattern
  + **Spring JDBC** – for JDBC and DataSource support.
  + **Spring ORM** – for ORM tools support such as Hibernate
  + **Spring Web Module** – for creating web applications.
  + **Spring MVC** – Model-View-Controller implementation for creating web applications, web services etc.

1. **What do you understand by Aspect Oriented Programming?**

Enterprise applications have some common cross-cutting concerns that is applicable for different types of Objects and application modules, such as logging, transaction management, data validation, authentication etc. In Object Oriented Programming, modularity of application is achieved by Classes whereas in AOP application modularity is achieved by Aspects and they are configured to cut across different classes methods.

AOP takes out the direct dependency of cross-cutting tasks from classes that is not possible in normal object oriented programming. For example, we can have a separate class for logging but again the classes will have to call these methods for logging the data. Read more about Spring AOP support at [Spring AOP Example](https://www.journaldev.com/2583/spring-aop-example-tutorial-aspect-advice-pointcut-joinpoint-annotations).

1. **What is Aspect, Advice, Pointcut, JointPoint and Advice Arguments in AOP?**

**Aspect**: Aspect is a class that implements cross-cutting concerns, such as transaction management. Aspects can be a normal class configured and then configured in Spring Bean configuration file or we can use Spring AspectJ support to declare a class as Aspect using @Aspect annotation.

**Advice**: Advice is the action taken for a particular join point. In terms of programming, they are methods that gets executed when a specific join point with matching pointcut is reached in the application. You can think of Advices as [Spring interceptors](https://www.journaldev.com/2676/spring-mvc-interceptor-example-handlerinterceptor-handlerinterceptoradapter) or [Servlet Filters](https://www.journaldev.com/1933/java-servlet-filter-example-tutorial).

**Pointcut**: Pointcut are regular expressions that is matched with join points to determine whether advice needs to be executed or not. Pointcut uses different kinds of expressions that are matched with the join points. Spring framework uses the AspectJ pointcut expression language for determining the join points where advice methods will be applied.

**Join Point**: A join point is the specific point in the application such as method execution, exception handling, changing object variable values etc. In Spring AOP a join points is always the execution of a method.

**Advice Arguments**: We can pass arguments in the advice methods. We can use args() expression in the pointcut to be applied to any method that matches the argument pattern. If we use this, then we need to use the same name in the advice method from where argument type is determined.

These concepts seems confusing at first, but if you go through [Spring Aspect, Advice Example](https://www.journaldev.com/2583/spring-aop-example-tutorial-aspect-advice-pointcut-joinpoint-annotations) then you can easily relate to them.

1. **What is the difference between Spring AOP and AspectJ AOP?**

AspectJ is the industry-standard implementation for Aspect Oriented Programming whereas Spring implements AOP for some cases. Main differences between Spring AOP and AspectJ are:

* + Spring AOP is simpler to use than AspectJ because we don’t need to worry about the weaving process.
  + Spring AOP supports AspectJ annotations, so if you are familiar with AspectJ then working with Spring AOP is easier.
  + Spring AOP supports only proxy-based AOP, so it can be applied only to method execution join points. AspectJ support all kinds of pointcuts.
  + One of the shortcoming of Spring AOP is that it can be applied only to the beans created through Spring Context.

1. **What is Spring IoC Container?**

**Inversion of Control** (IoC) is the mechanism to achieve loose-coupling between Objects dependencies. To achieve loose coupling and dynamic binding of the objects at runtime, the objects define their dependencies that are being injected by other assembler objects. Spring IoC container is the program that injects dependencies into an object and make it ready for our use.

Spring Framework IoC container classes are part of org.springframework.beans and org.springframework.context packages and provides us different ways to decouple the object dependencies.

Some of the useful ApplicationContext implementations that we use are;

* + AnnotationConfigApplicationContext: For standalone java applications using annotations based configuration.
  + ClassPathXmlApplicationContext: For standalone java applications using XML based configuration.
  + FileSystemXmlApplicationContext: Similar to ClassPathXmlApplicationContext except that the xml configuration file can be loaded from anywhere in the file system.
  + AnnotationConfigWebApplicationContext and XmlWebApplicationContext for web applications.

1. **What is a Spring Bean?**

Any normal java class that is initialized by Spring IoC container is called Spring Bean. We use Spring ApplicationContext to get the Spring Bean instance.

Spring IoC container manages the life cycle of Spring Bean, bean scopes and injecting any required dependencies in the bean.

1. **What is the importance of Spring bean configuration file?**

We use Spring Bean configuration file to define all the beans that will be initialized by Spring Context. When we create the instance of Spring ApplicationContext, it reads the spring bean xml file and initialize all of them. Once the context is initialized, we can use it to get different bean instances.

Apart from Spring Bean configuration, this file also contains spring MVC interceptors, view resolvers and other elements to support annotations based configurations.

1. **What are different ways to configure a class as Spring Bean?**

There are three different ways to configure Spring Bean.

* + **XML Configuration**: This is the most popular configuration and we can use bean element in context file to configure a Spring Bean. For example:
  + <bean name="myBean" class="com.journaldev.spring.beans.MyBean"></bean>
  + **Java Based Configuration**: If you are using only annotations, you can configure a Spring bean using @Bean annotation. This annotation is used with @Configuration classes to configure a spring bean. Sample configuration is:
  + @Configuration
  + @ComponentScan(value="com.journaldev.spring.main")
  + public class MyConfiguration {
  + @Bean
  + public MyService getService(){
  + return new MyService(); }}

To get this bean from spring context, we need to use following code snippet:

AnnotationConfigApplicationContext ctx = new AnnotationConfigApplicationContext(

MyConfiguration.class);

MyService service = ctx.getBean(MyService.class);

* + **Annotation Based Configuration**: We can also use @Component, @Service, @Repository and @Controller annotations with classes to configure them to be as spring bean. For these, we would need to provide base package location to scan for these classes. For example:
  + <context:component-scan base-package="com.journaldev.spring" />

1. **What are different scopes of Spring Bean?**

There are five scopes defined for Spring Beans.

* + [**singleton**](https://www.journaldev.com/1377/java-singleton-design-pattern-best-practices-examples): Only one instance of the bean will be created for each container. This is the default scope for the spring beans. While using this scope, make sure spring bean doesn’t have shared instance variables otherwise it might lead to data inconsistency issues because it’s not thread-safe.
  + **prototype**: A new instance will be created every time the bean is requested.
  + **request**: This is same as prototype scope, however it’s meant to be used for web applications. A new instance of the bean will be created for each HTTP request.
  + **session**: A new bean will be created for each HTTP session by the container.
  + **global-session**: This is used to create global session beans for Portlet applications.

Spring Framework is extendable and we can create our own scopes too, however most of the times we are good with the scopes provided by the framework.

To set spring bean scopes we can use “scope” attribute in bean element or @Scope annotation for annotation based configurations.

1. **How to get ServletContext and ServletConfig object in a Spring Bean?**

There are two ways to get Container specific objects in the spring bean.

* + Implementing Spring \*Aware interfaces, for these ServletContextAware and ServletConfigAware interfaces, for complete example of these aware interfaces, please read [Spring Aware Interfaces](https://www.journaldev.com/2637/spring-bean-life-cycle)
  + Using @Autowired annotation with bean variable of type ServletContext and ServletConfig. They will work only in servlet container specific environment only though.
  + @Autowired
  + ServletContext servletContext;

1. **Does Spring Bean provide thread safety?**

The default scope of Spring bean is singleton, so there will be only one instance per context. That means that all the having a class level variable that any thread can update will lead to inconsistent data. Hence in default mode spring beans are not thread-safe.

However we can change spring bean scope to request, prototype or session to achieve thread-safety at the cost of performance. It’s a design decision and based on the project requirements.

1. **What is DispatcherServlet and ContextLoaderListener?**

DispatcherServlet is the front controller in the Spring MVC application and it loads the spring bean configuration file and initialize all the beans that are configured. If annotations are enabled, it also scans the packages and configure any bean annotated with @Component, @Controller, @Repository or @Service annotations.

ContextLoaderListener is the listener to **start up and shut down Spring’s** root WebApplicationContext. It’s important functions are to tie up the lifecycle of ApplicationContext to the lifecycle of the ServletContext and to automate the creation of ApplicationContext. We can use it to define shared beans that can be used across different spring contexts.

1. **What is ViewResolver in Spring?**

ViewResolver implementations are used to resolve the view pages by name. Usually we configure it in the spring bean configuration file. For example:

<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views directory -->

<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<beans:property name="prefix" value="/WEB-INF/views/" />

<beans:property name="suffix" value=".jsp" />

</beans:bean>

InternalResourceViewResolver is one of the implementation of ViewResolver interface and we are providing the view pages directory and suffix location through the bean properties. So if a controller handler method returns “home”, view resolver will use view page located at */WEB-INF/views/home.jsp*.

1. **What is a MultipartResolver and when its used?**

MultipartResolver interface is used for uploading files – CommonsMultipartResolver and StandardServletMultipartResolver are two implementations provided by spring framework for file uploading. By default there are no multipart resolvers configured but to use them for uploading files, all we need to define a bean named “multipartResolver” with type as MultipartResolver in spring bean configurations.

Once configured, any multipart request will be resolved by the configured MultipartResolver and pass on a wrapped HttpServletRequest. Then it’s used in the controller class to get the file and process it.

1. **How to handle exceptions in Spring MVC Framework?**

Spring MVC Framework provides following ways to help us achieving robust exception handling.

* + **Controller Based** – We can define exception handler methods in our controller classes. All we need is to annotate these methods with @ExceptionHandler annotation.
  + **Global Exception Handler** – Exception Handling is a cross-cutting concern and Spring provides @ControllerAdvice annotation that we can use with any class to define our global exception handler.
  + **HandlerExceptionResolver implementation** – For generic exceptions, most of the times we serve static pages. Spring Framework provides HandlerExceptionResolver interface that we can implement to create global exception handler. The reason behind this additional way to define global exception handler is that Spring framework also provides default implementation classes that we can define in our spring bean configuration file to get spring framework exception handling benefits.

1. **How to create ApplicationContext in a Java Program?**

There are following ways to create spring context in a standalone java program.

* + **AnnotationConfigApplicationContext**: If we are using Spring in standalone java applications and using annotations for Configuration, then we can use this to initialize the container and get the bean objects.
  + **ClassPathXmlApplicationContext**: If we have spring bean configuration xml file in standalone application, then we can use this class to load the file and get the container object.
  + **FileSystemXmlApplicationContext**: This is similar to ClassPathXmlApplicationContext except that the xml configuration file can be loaded from anywhere in the file system.

1. **Can we have multiple Spring configuration files?**

For Spring MVC applications, we can define multiple spring context configuration files through contextConfigLocation. This location string can consist of multiple locations separated by any number of commas and spaces. For example;

<servlet>

<servlet-name>appServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring/appServlet/servlet-context.xml,/WEB-INF/spring/appServlet/servlet-jdbc.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

We can also define multiple root level spring configurations and load it through context-param. For example;

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring/root-context.xml /WEB-INF/spring/root-security.xml</param-value>

</context-param>

Another option is to use import element in the context configuration file to import other configurations, for example:

<beans:import resource="spring-jdbc.xml"/>

1. **What is ContextLoaderListener?**

ContextLoaderListener is the listener class used to load root context and define spring bean configurations that will be visible to all other contexts. It’s configured in web.xml file as:

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring/root-context.xml</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

1. **What are the minimum configurations needed to create Spring MVC application?**

For creating a simple Spring MVC application, we would need to do following tasks.

* + Add spring-context and spring-webmvc dependencies in the project.
  + Configure DispatcherServlet in the web.xml file to handle requests through spring container.
  + Spring bean configuration file to define beans, if using annotations then it has to be configured here. Also we need to configure view resolver for view pages.
  + Controller class with request mappings defined to handle the client requests.

Above steps should be enough to create a simple Spring MVC Hello World application.

1. **How would you relate Spring MVC Framework to MVC architecture?**

As the name suggests Spring MVC is built on top of **Model-View-Controller** architecture. DispatcherServlet is the Front Controller in the Spring MVC application that takes care of all the incoming requests and delegate it to different controller handler methods.

Model can be any Java Bean in the Spring Framework, just like any other MVC framework Spring provides automatic binding of form data to java beans. We can set model beans as attributes to be used in the view pages.

View Pages can be JSP, static HTMLs etc. and view resolvers are responsible for finding the correct view page. Once the view page is identified, control is given back to the DispatcherServlet controller. DispatcherServlet is responsible for rendering the view and returning the final response to the client.

1. **How to achieve localization in Spring MVC applications?**

Spring provides excellent support for localization or i18n through resource bundles. Basis steps needed to make our application localized are:

* + Creating message resource bundles for different locales, such as messages\_en.properties, messages\_fr.properties etc.
  + Defining messageSource bean in the spring bean configuration file of type ResourceBundleMessageSource or ReloadableResourceBundleMessageSource.
  + For change of locale support, define localeResolver bean of type CookieLocaleResolver and configure LocaleChangeInterceptor interceptor. Example configuration can be like below:
  + <beans:bean id="messageSource"
  + class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
  + <beans:property name="basename" value="classpath:messages" />
  + <beans:property name="defaultEncoding" value="UTF-8" />
  + </beans:bean>
  + <beans:bean id="localeResolver"
  + class="org.springframework.web.servlet.i18n.CookieLocaleResolver">
  + <beans:property name="defaultLocale" value="en" />
  + <beans:property name="cookieName" value="myAppLocaleCookie"></beans:property>
  + <beans:property name="cookieMaxAge" value="3600"></beans:property>
  + </beans:bean>
  + <interceptors>
  + <beans:bean class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">
  + <beans:property name="paramName" value="locale" />
  + </beans:bean>
  + </interceptors>
  + Use spring:message element in the view pages with key names, DispatcherServlet picks the corresponding value and renders the page in corresponding locale and return as response.

For a complete example, please read [Spring Localization Example](https://www.journaldev.com/2610/spring-mvc-internationalization-i18n-and-localization-l10n-example).

1. **How can we use Spring to create Restful Web Service returning JSON response?**

We can use Spring Framework to create Restful web services that returns JSON data. Spring provides integration with [Jackson JSON](https://www.journaldev.com/2324/jackson-json-java-parser-api-example-tutorial) API that we can use to send JSON response in restful web service.

We would need to do following steps to configure our Spring MVC application to send JSON response:

* + Adding [Jackson](https://www.journaldev.com/2324/jackson-json-java-parser-api-example-tutorial) JSON dependencies, if you are using Maven it can be done with following code:
  + <!-- Jackson -->
  + <dependency>
  + <groupId>com.fasterxml.jackson.core</groupId>
  + <artifactId>jackson-databind</artifactId>
  + <version>${jackson.databind-version}</version>
  + </dependency>
  + Configure RequestMappingHandlerAdapter bean in the spring bean configuration file and set the messageConverters property to MappingJackson2HttpMessageConverter bean. Sample configuration will be:
  + <!-- Configure to plugin JSON as request and response in method handler -->
  + <beans:bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter">
  + <beans:property name="messageConverters">
  + <beans:list>
  + <beans:ref bean="jsonMessageConverter"/>
  + </beans:list>
  + </beans:property>
  + </beans:bean>
  + <!-- Configure bean to convert JSON to POJO and vice versa -->
  + <beans:bean id="jsonMessageConverter" class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter">
  + </beans:bean>
  + In the controller handler methods, return the Object as response using @ResponseBodyannotation. Sample code:
  + @RequestMapping(value = EmpRestURIConstants.GET\_EMP, method = RequestMethod.GET)
  + public @ResponseBody Employee getEmployee(@PathVariable("id") int empId) {
  + logger.info("Start getEmployee. ID="+empId);
  + return empData.get(empId);
  + }
  + You can invoke the rest service through any API, but if you want to use Spring then we can easily do it using RestTemplate class.

For a complete example, please read [Spring Restful Webservice Example](https://www.journaldev.com/2552/spring-rest-example-tutorial-spring-restful-web-services).

1. **What are some of the important Spring annotations you have used?**

Some of the Spring annotations that I have used in my project are:

* + **@Controller** – for controller classes in Spring MVC project.
  + **@RequestMapping** – for configuring URI mapping in controller handler methods. This is a very important annotation, so you should go through [Spring MVC RequestMapping Annotation Examples](https://www.journaldev.com/3358/spring-requestmapping-requestparam-pathvariable-example)
  + **@ResponseBody** – for sending Object as response, usually for sending XML or JSON data as response.
  + **@PathVariable** – for mapping dynamic values from the URI to handler method arguments.
  + **@Autowired** – for autowiring dependencies in spring beans.
  + **@Qualifier** – with @Autowired annotation to avoid confusion when multiple instances of bean type is present.
  + **@Service** – for service classes.
  + **@Scope** – for configuring scope of the spring bean.
  + **@Configuration**, **@ComponentScan** and **@Bean** – for java based configurations.
  + AspectJ annotations for configuring aspects and advices, **@Aspect**, **@Before**, **@After**, **@Around**, **@Pointcut** etc.

1. **Can we send an Object as the response of Controller handler method?**

Yes we can, using **@ResponseBody** annotation. This is how we send JSON or XML based response in restful web services.

1. **How to upload file in Spring MVC Application?**

Spring provides built-in support for uploading files through **MultipartResolver** interface implementations. It’s very easy to use and requires only configuration changes to get it working. Obviously we would need to write controller handler method to handle the incoming file and process it. For a complete example

1. **How to validate form data in Spring Web MVC Framework?**

Spring supports JSR-303 annotation based validations as well as provide Validator interface that we can implement to create our own custom validator. For using JSR-303 based validation, we need to annotate bean variables with the required validations.

For custom validator implementation, we need to configure it in the controller class.

1. **What is Spring MVC Interceptor and how to use it?**

Spring MVC Interceptors are like Servlet Filters and allow us to intercept client request and process it. We can intercept client request at three places – **preHandle**, **postHandle** and **afterCompletion**.

We can create spring interceptor by implementing HandlerInterceptor interface or by extending abstract class **HandlerInterceptorAdapter**.

We need to configure interceptors in the spring bean configuration file. We can define an interceptor to intercept all the client requests or we can configure it for specific URI mapping too.

.

1. **What is Spring JdbcTemplate class and how to use it?**

Spring Framework provides excellent integration with JDBC API and provides JdbcTemplate utility class that we can use to avoid bolier-plate code from our database operations logic such as Opening/Closing Connection, ResultSet, PreparedStatement etc.

1. **How to use Tomcat JNDI DataSource in Spring Web Application?**

For using servlet container configured JNDI DataSource, we need to configure it in the spring bean configuration file and then inject it to spring beans as dependencies. Then we can use it with JdbcTemplate to perform database operations.

Sample configuration would be:

<beans:bean id="dbDataSource" class="org.springframework.jndi.JndiObjectFactoryBean">

<beans:property name="jndiName" value="java:comp/env/jdbc/MyLocalDB"/>

</beans:bean>

1. **How would you achieve Transaction Management in Spring?**

Spring framework provides transaction management support through Declarative Transaction Management as well as programmatic transaction management. Declarative transaction management is most widely used because it’s easy to use and works in most of the cases.

We use annotate a method with @Transactional annotation for Declarative transaction management. We need to configure transaction manager for the DataSource in the spring bean configuration file.

<bean id="transactionManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource" />

</bean>

1. **What is Spring DAO?**

Spring DAO support is provided to work with data access technologies like JDBC, Hibernate in a consistent and easy way. For example we have JdbcDaoSupport, HibernateDaoSupport, JdoDaoSupport and JpaDaoSupport for respective technologies.

Spring DAO also provides consistency in exception hierarchy and we don’t need to catch specific exceptions.

1. **How to integrate Spring and Hibernate Frameworks?**

We can use Spring ORM module to integrate Spring and Hibernate frameworks, if you are using Hibernate 3+ where SessionFactory provides current session, then you should avoid using HibernateTemplate or HibernateDaoSupport classes and better to use DAO pattern with dependency injection for the integration.

Also Spring ORM provides support for using Spring declarative transaction management, so you should utilize that rather than going for hibernate boiler-plate code for transaction management.

1. **What is Spring Security?**

Spring security framework focuses on providing both authentication and authorization in java applications. It also takes care of most of the common security vulnerabilities such as CSRF attack.

It’s very beneficial and easy to use Spring security in web applications, through the use of annotations such as @EnableWebSecurity. You should go through following posts to learn how to use Spring Security framework.

1. **How to inject a java.util.Properties into a Spring Bean?**

We need to define propertyConfigurer bean that will load the properties from the given property file. Then we can use Spring EL support to inject properties into other bean dependencies. For example;

<bean id="propertyConfigurer"

class="org.springframework.context.support.PropertySourcesPlaceholderConfigurer">

<property name="location" value="/WEB-INF/application.properties" />

</bean>

<bean class="com.journaldev.spring.EmployeeDaoImpl">

<property name="maxReadResults" value="${results.read.max}"/>

</bean>

If you are using annotation to configure the spring bean, then you can inject property like below.

@Value("${maxReadResults}")

private int maxReadResults;

1. **Name some of the design patterns used in Spring Framework?**

Spring Framework is using a lot of design patterns, some of the common ones are:

* + Singleton Pattern: Creating beans with default scope.
  + [Factory Pattern](https://www.journaldev.com/1392/factory-design-pattern-in-java): Bean Factory classes
  + [Prototype Pattern](https://www.journaldev.com/1440/prototype-design-pattern-in-java): Bean scopes
  + [Adapter Pattern](https://www.journaldev.com/1487/adapter-design-pattern-java): Spring Web and Spring MVC
  + [Proxy Pattern](https://www.journaldev.com/1572/proxy-design-pattern): Spring Aspect Oriented Programming support
  + [Template Method Pattern](https://www.journaldev.com/1763/template-method-design-pattern-in-java): JdbcTemplate, HibernateTemplate etc
  + Front Controller: Spring MVC DispatcherServlet
  + Data Access Object: Spring DAO support
  + Dependency Injection and Aspect Oriented Programming

1. **What are some of the best practices for Spring Framework?**

Some of the best practices for Spring Framework are:

* + Avoid version numbers in schema reference, to make sure we have the latest configs.
  + Divide spring bean configurations based on their concerns such as spring-jdbc.xml, spring-security.xml.
  + For spring beans that are used in multiple contexts in Spring MVC, create them in the root context and initialize with listener.
  + Configure bean dependencies as much as possible, try to avoid autowiring as much as possible.
  + For application level properties, best approach is to create a property file and read it in the spring bean configuration file.
  + For smaller applications, annotations are useful but for larger applications annotations can become a pain. If we have all the configuration in xml files, maintaining it will be easier.
  + Use correct annotations for components for understanding the purpose easily. For services use @Service and for DAO beans use @Repository.
  + Spring framework has a lot of modules, use what you need. Remove all the extra dependencies that gets usually added when you create projects through Spring Tool Suite templates.
  + If you are using Aspects, make sure to keep the join pint as narrow as possible to avoid advice on unwanted methods. Consider custom annotations that are easier to use and avoid any issues.
  + Use dependency injection when there is actual benefit, just for the sake of loose-coupling don’t use it because it’s harder to maintain.

### 15. What are the common implementations of the ApplicationContext?

The **FileSystemXmlApplicationContext** container loads the definitions of the beans from an XML file. The full path of the XML bean configuration file must be provided to the constructor.  
The **ClassPathXmlApplicationContext** container also loads the definitions of the beans from an XML file. Here, you need to set CLASSPATH properly because this container will look bean configuration XML file in CLASSPATH.  
The **WebXmlApplicationContext:** container loads the XML file with definitions of all beans from within a web application.

### 16. What is the difference between Bean Factory and ApplicationContext?

Application contexts provide a means for resolving text messages, a generic way to load file resources (such as images), they can publish events to beans that are registered as listeners. In addition, operations on the container or beans in the container, which have to be handled in a programmatic fashion with a bean factory, can be handled declaratively in an application context. The application context implements MessageSource, an interface used to obtain localized messages, with the actual implementation being pluggable.

### 18. What is Dependency Injection in Spring?

[Dependency Injection](http://www.javacodegeeks.com/2014/02/dependency-injection-options-for-java.html), an aspect of Inversion of Control (IoC), is a general concept, and it can be expressed in many different ways.This concept says that you do not create your objects but describe how they should be created. You don’t directly connect your components and services together in code but describe which services are needed by which components in a configuration file. A container (the IOC container) is then responsible for hooking it all up.

### 19. What are the different types of IoC (dependency injection)?

* **Constructor-based dependency injection:** Constructor-based DI is accomplished when the container invokes a class constructor with a number of arguments, each representing a dependency on other class.
* **Setter-based dependency injection:** Setter-based DI is accomplished by the container calling setter methods on your beans after invoking a no-argument constructor or no-argument static factory method to instantiate your bean.

### 20. Which DI would you suggest Constructor-based or setter-based DI?

You can use both Constructor-based and Setter-based Dependency Injection. The best solution is using constructor arguments for mandatory dependencies and setters for optional dependencies.

## **Spring Beans**

### 21. What are Spring beans?

The [Spring Beans](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-3-bean-reference-example/) are Java Objects that form the backbone of a Spring application. They are instantiated, assembled, and managed by the Spring IoC container. These beans are created with the configuration metadata that is supplied to the container, for example, in the form of XML <bean/> definitions.

Beans defined in spring framework are singleton beans. There is an attribute in bean tag named "singleton" if specified true then bean becomes singleton and if set to false then the bean becomes a prototype bean. By default it is set to true. So, all the beans in spring framework are by default singleton beans.

### 22. What does a Spring Bean definition contain?

A Spring Bean definition contains all configuration metadata which is needed for the container to know how to create a bean, its lifecycle details and its dependencies.

### 23. How do you provide configuration metadata to the Spring Container?

There are three important methods to provide configuration metadata to the Spring Container:

* XML based configuration file.
* Annotation-based configuration
* [Java-based configuration](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-3-java-config-example/)

### 24. How do you define the scope of a bean?

When defining a <bean> in Spring, we can also declare a scope for the bean. It can be defined through the scope attribute in the bean definition. For example, when Spring has to produce a new bean instance each time one is needed, the bean’s scope attribute to be prototype. On the other hand, when the same instance of a bean must be returned by Spring every time it is needed, the the bean scope attribute must be set to singleton.

### 25. Explain the bean scopes supported by Spring

There are five scoped provided by the Spring Framework supports following five scopes:

* In **singleton** scope, Spring scopes the bean definition to a single instance per Spring IoC container.
* In **prototype** scope, a single bean definition has any number of object instances.
* In **request** scope, a bean is defined to an HTTP request. This scope is valid only in a web-aware Spring ApplicationContext.
* In **session**scope, a bean definition is scoped to an HTTP session. This scope is also valid only in a web-aware Spring ApplicationContext.
* In **global-session** scope, a bean definition is scoped to a global HTTP session. This is also a case used in a web-aware Spring ApplicationContext.

The default scope of a Spring Bean is Singleton.

### 28. Which are the important beans lifecycle methods? Can you override them?

There are two important bean lifecycle methods. The first one is setup which is called when the bean is loaded in to the container. The second method is the teardown method which is called when the bean is unloaded from the container.  
The bean tag has two important attributes (init-method and destroy-method) with which you can define your own custom initialization and destroy methods. There are also the correspondive annotations(@PostConstruct and @PreDestroy).

### 29. What are inner beans in Spring?

When a bean is only used as a property of another bean it can be declared as an inner bean. Spring’s XML-based configuration metadata provides the use of <bean/> element inside the <property/> or <constructor-arg/> elements of a bean definition, in order to define the so-called inner bean. Inner beans are always anonymous and they are always scoped as prototypes.

### 30. How can you inject a Java Collection in Spring?

Spring offers the following types of [collection configuration elements](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-collections-list-set-map-and-properties-example/):

* The <list> type is used for injecting a list of values, in the case that duplicates are allowed.
* The <set> type is used for wiring a set of values but without any duplicates.
* The <map> type is used to inject a collection of name-value pairs where name and value can be of any type.
* The <props> type can be used to inject a collection of name-value pairs where the name and value are both Strings.

### 32. What is bean auto wiring?

The Spring container is able to [autowire relationships](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-autowire-example/) between collaborating beans. This means that it is possible to automatically let Spring resolve collaborators (other beans) for a bean by inspecting the contents of the BeanFactorywithout using <constructor-arg> and <property> elements.

### 33. Explain different modes of auto wiring?

The autowiring functionality has five modes which can be used to instruct Spring container to use autowiring for dependency injection:

* **no:** This is default setting. Explicit bean reference should be used for wiring.
* **byName:** When autowiring byName, the Spring container looks at the properties of the beans on which autowireattribute is set to byName in the XML configuration file. It then tries to match and wire its properties with the beans defined by the same names in the configuration file.
* **byType:** When autowiring by datatype, the Spring container looks at the properties of the beans on which autowireattribute is set to byType in the XML configuration file. It then tries to match and wire a property if its type matches with exactly one of the beans name in configuration file. If more than one such beans exist, a fatal exception is thrown.
* **constructor:** This mode is similar to byType, but type applies to constructor arguments. If there is not exactly one bean of the constructor argument type in the container, a fatal error is raised.
* **autodetect:**Spring first tries to wire using autowire by constructor, if it does not work, Spring tries to autowire by byType.

### 34. Are there limitations with autowiring?

Limitations of autowiring are:

* **Overriding:**You can still specify dependencies using <constructor-arg> and <property> settings which will always override autowiring.
* **Primitive data types:** You cannot autowire simple properties such as primitives, Strings, and Classes.
* **Confusing nature:** Autowiring is less exact than explicit wiring, so if possible prefer using explicit wiring.

### 35. Can you inject null and empty string values in Spring?

Yes, you can.

### 36. What is Spring Java-Based Configuration? Give some annotation example.

[Java based configuration](http://www.javacodegeeks.com/2013/04/spring-java-configuration.html) option enables you to write most of your Spring configuration without XML but with the help of few Java-based annotations.  
An example is the @Configuration annotation, that indicates that the class can be used by the Spring IoC container as a source of bean definitions. Another example is the@Bean annotated method that will return an object that should be registered as a bean in the Spring application context.

### 37. What is Annotation-based container configuration?

An alternative to XML setups is provided by annotation-based configuration which relies on the bytecode metadata for wiring up components instead of angle-bracket declarations. 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.

### 38. How do you turn on annotation wiring?

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

### 39. @Required annotation

This annotation simply indicates that the affected bean property must be populated at configuration time, through an explicit property value in a bean definition or through autowiring. The container throws BeanInitializationException if the affected bean property has not been populated.

### 40. @Autowired annotation

The @Autowired annotation provides more fine-grained control over where and how autowiring should be accomplished. It can be used to autowire bean on the setter method just like @Required annotation, on the constructor, on a property or pn methods with arbitrary names and/or multiple arguments.

### 41. @Qualifier annotation

When there are more than one beans of the same type and only one is needed to be wired with a property, the @Qualifierannotation is used along with @Autowired annotation to remove the confusion by specifying which exact bean will be wired.

## **Spring Data Access**

### 42. How can JDBC be used more efficiently in the Spring framework?

When using the Spring JDBC framework the burden of resource management and error handling is reduced. So developers only need to write the statements and queries to get the data to and from the database. JDBC can be used more efficiently with the help of a template class provided by Spring framework, which is the JdbcTemplate (example [here](http://examples.javacodegeeks.com/enterprise-java/spring/jdbc/spring-jdbctemplate-example/)).

### 43. JdbcTemplate

JdbcTemplate class provides many convenience methods for doing things such as converting database data into primitives or objects, executing prepared and callable statements, and providing custom database error handling.

### 44. Spring DAO support

The [Data Access Object (DAO) support in Spring](http://www.javacodegeeks.com/2012/09/spring-dao-and-service-layer.html) is aimed at making it easy to work with data access technologies like JDBC, Hibernate or JDO in a consistent way. This allows us to switch between the persistence technologies fairly easily and to code without worrying about catching exceptions that are specific to each technology.

### 45. What are the ways to access Hibernate by using Spring?

There are two ways to access Hibernate with Spring:

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

### 47. How can we integrate Spring and Hibernate using HibernateDaoSupport?

Use Spring’s SessionFactory called LocalSessionFactory. The integration process is of 3 steps:

* Configure the Hibernate SessionFactory
* Extend a DAO Implementation from HibernateDaoSupport
* Wire in Transaction Support with AOP

### 48. Types of the transaction management Spring support

Spring supports two types of transaction management:

* **Programmatic transaction management:** This means that you have managed the transaction with the help of programming. That gives you extreme flexibility, but it is difficult to maintain.
* **Declarative transaction management:** This means you separate [transaction management from the business code](http://www.javacodegeeks.com/2011/09/spring-declarative-transactions-example.html). You only use annotations or XML based configuration to manage the transactions.

### 49. What are the benefits of the Spring Framework’s transaction management?

* It provides a consistent programming model across different transaction APIs such as JTA, JDBC, Hibernate, JPA, and JDO.
* It provides a simpler API for programmatic transaction management than a number of complex transaction APIs such as JTA.
* It supports declarative transaction management.
* It integrates very well with Spring’s various data access abstractions.

### 50. Which Transaction management type is more preferable?

Most users of the Spring Framework choose declarative transaction management because it is the option with the least impact on application code, and hence is most consistent with the ideals of a non-invasive lightweight container. Declarative transaction management is preferable over programmatic transaction management though it is less flexible than programmatic transaction management, which allows you to control transactions through your code.

## **Spring Aspect Oriented Programming (AOP)**

### 51. Explain AOP

[Aspect-oriented programming](http://www.javacodegeeks.com/2014/02/applying-aspect-oriented-programming.html), or AOP, is a programming technique that allows programmers to modularize crosscutting concerns, or behavior that cuts across the typical divisions of responsibility, such as logging and transaction management.

### 52. Aspect

The core construct of AOP is the aspect, which encapsulates behaviors affecting multiple classes into reusable modules. It ia 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. In Spring AOP, aspects are implemented using regular classes annotated with the @Aspect annotation (@AspectJ style).

### 54. Join point

The join point represents a point in an application where we can plug-in an AOP aspect. It is the actual place in the application where an action will be taken using Spring AOP framework.

### 55. Advice

The advice is the actual action that will be taken either before or after the method execution. This is actual piece of code that is invoked during the program execution by the Spring AOP framework.

Spring aspects can work with five kinds of advice:

* **before:** Run advice before the a method execution.
* **after:** Run advice after the a method execution regardless of its outcome.
* **after-returning:** Run advice after the a method execution only if method completes successfully.
* **after-throwing:** Run advice after the a method execution only if method exits by throwing an exception.
* **around:** Run advice before and after the advised method is invoked.

### 56. Pointcut

The pointcut is a set of one or more joinpoints where an advice should be executed. You can specify pointcuts using expressions or patterns.

### 57. What is Introduction?

An Introduction allows us to add new methods or attributes to existing classes.

### 58. What is Target object?

The target object is an object being advised by one or more aspects. It will always be a proxy object. It is also referred to as the advised object.

### 59. What is a Proxy?

A proxy is an object that is created after applying advice to a target object. When you think of client objects the target object and the proxy object are the same.

### 60. What are the different types of AutoProxying?

* BeanNameAutoProxyCreator
* DefaultAdvisorAutoProxyCreator
* Metadata autoproxying

### 61. What is Weaving? What are the different points where weaving can be applied?

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

### 62. Explain XML Schema-based aspect implementation?

In this implementation case, aspects are implemented using regular classes along with XML based configuration.

### 63. Explain annotation-based (@AspectJ based) aspect implementation

This implementation case (@AspectJ based implementation) refers to a style of declaring aspects as regular Java classes annotated with Java 5 annotations.

## **Spring Model View Controller (MVC)**

### 64. What is Spring MVC framework?

Spring comes with a [full-featured MVC framework for building web applications](http://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-hello-world-example/). Although Spring can easily be integrated with other MVC frameworks, such as Struts, Spring’s MVC framework uses IoC to provide a clean separation of controller logic from business objects. It also allows to declaratively bind request parameters to business objects.

### 65. DispatcherServlet

The Spring Web MVC framework is designed around a DispatcherServlet that handles all the HTTP requests and responses.

### 66. WebApplicationContext

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

### 67. What is Controller in Spring MVC framework?

Controllers provide access to the application behavior that you typically define through a service interface. Controllers interpret user input and transform it into a model that is represented to the user by the view. Spring implements a controller in a very abstract way, which enables you to create a wide variety of controllers.

### 68. @Controller annotation

The @Controller annotation indicates that a particular class serves the role of a controller. Spring does not require you to extend any controller base class or reference the Servlet API.

### 69. @RequestMapping annotation

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

#### 4. What is dependency injection(IOC) in Spring?

The basic concept of the dependency injection (also known as Inversion of Control pattern) is that you do not create your objects but describe how they should be created. You don’t directly connect your components and services together in code but describe which services are needed by which components in a configuration file. A container (in the case of the Spring framework, the IOC container) is then responsible for hooking it all up.  
i.e., Applying IoC, objects are given their dependencies at creation time by some external entity that coordinates each object in the system. That is, dependencies are injected into objects. So, IoC means an inversion of responsibility with regard to how an object obtains references to collaborating objects.

#### 5. What are ways to inject dependency in Spring?

There are two ways to do dependency injection in Spring.

* [Dependency injection via setter method](https://www.java2blog.com/2012/08/dependency-injection-via-setter-method.html)
* [Dependency injection via constructor](https://www.java2blog.com/2012/08/dependency-injection-via-constructor-in.html).

#### 6. What is Bean in Spring?

A normal POJO class managed by Spring IOC container are called Spring beans. It is core part of Spring application.

Example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <bean id="countryBean" class="org.arpit.java2blog.Country">        <property name="countryName" value="India"/>        <property name="capital" ref="CapitalBean"/>  </bean> |

#### 7. How can you configure Spring in your application?

There are 3 ways to do it.

* XML based configuration
* Java based configuration
* Annotation based configuration.

#### 8. What is Spring XML based configuration?

In Spring XML based configuration, you define all dependency in an XML file. You define all your beans with tag in XML file and all dependencies are read using this XML file.

for example :  
Sample ApplicationContext.xml file

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <!--?xml version="1.0" encoding="UTF-8"?-->  <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">      <bean id="CountryBean" class="org.arpit.javapostsforlearning.Country">        <property name="countryName" value="India"/>        <property name="capital" ref="CapitalBean"/>    </bean>    <bean id="CapitalBean" class="org.arpit.javapostsforlearning.Capital">        <property name="capitalName" value="Delhi"/>    </bean>  </beans> |

You can read this ApplicationContext.xml using:

|  |  |
| --- | --- |
|  | ApplicationContext appContext = new ClassPathXmlApplicationContext("ApplicationContext.xml") |

#### 9. What is Spring java based configuration?

In Spring Java based configuration, you inject all dependencies using java class only. You can use @Configuaration and @Bean annotations to do it.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | package org.arpit.java2blog.config;  import org.arpit.java2blog.model.Country;  import org.springframework.context.annotation.Bean;  import org.springframework.context.annotation.Configuration;    @Configuration  public class ApplicationConfiguration {    @Bean(name="countryObj")  public Country getCountry()  {    return new Country("India");  }    } |

Above file is equivalent to below spring configuration xml

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xmlns:context="http://www.springframework.org/schema/context"      xsi:schemaLocation="http://www.springframework.org/schema/beans      http://www.springframework.org/schema/beans/spring-beans-3.0.xsd      http://www.springframework.org/schema/context      http://www.springframework.org/schema/context/spring-context-3.0.xsd">  <context:annotation-config/>  <bean id="countryObj" class="org.arpit.java2blog.Country" >    <property name="countryName" value="India"/>  </bean>  </beans> |

To get above bean to application context, you need to use below code

|  |  |
| --- | --- |
| 1  2  3  4 | ApplicationContext appContext = new AnnotationConfigApplicationContext(ApplicationConfiguration.class);  Country countryObj = (Country) appContext.getBean("countryObj"); |

You can refer [Spring java based configuration](https://www.java2blog.com/2016/02/spring-java-based-configuration-example.html) for complete example.

#### 10. What is spring annotation based configuration?

You can do dependency injection via annotation also instead of XML configuration. You can define bean autowiring using annotations. You can use @Component,@Repository,@Service and @Controller annotation to configure bean in Spring application.  
Annotations wiring is not turned on by default. You need to turn it on using :

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <?xml version="1.0" encoding="UTF-8"?>  <bean>  <context:annotation-config/>  </beans> |

Once you put above code, you can start using annotation on class , fields or methods.

#### 11. What are different bean scopes in Spring?

There are 5 types of bean scopes supported in spring

* **singleton** – Scopes a single bean definition to a single object instance per Spring IoC container.
* **prototype** – Return a new bean instance each time when requested
* **request**– Return a single bean instance per HTTP request.
* **session**– Return a single bean instance per HTTP session.
* **globalSession** – Return a single bean instance per global HTTP session.

#### 12. What is default scope of bean in Spring?

singleton is default scope of a bean in Spring. You have to explicitly change scope of a bean if you want different scope.This is one of most asked spring interiew questions.

#### 13. What is ApplicationContext and what are its functions?

ApplicationContext is an central interface for providing configuration information to an application.

An ApplicationContext provides the following functionalities:

* Bean factory methods, inherited from ListableBeanFactory. This avoids the need for applications to use singletons.
* The ability to resolve messages, supporting internationalization. Inherited from the MessageSource interface.
* The ability to load file resources in a generic fashion. Inherited from the ResourceLoader interface.
* The ability to publish events. Implementations must provide a means of registering event listeners.
* Inheritance from a parent context. Definitions in a descendant context will always take priority. This means, for example, that a single parent context can be used by an entire web application, while each servlet has its own child context that is independent of that of any other servlet.

#### 14. How do you injection collection in Spring?

You can initialize collection using list and value tag as below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <bean id="CountryBean" class="org.arpit.java2blog.Country">    <property name="listOfStates">     <list>      <value>Himachal Pradesh</value>      <value>West Bengal</value>      <value>Gujrat</value>     </list>    </property>  </bean> |

#### 15. What do you mean by bean autowiring in Spring?

In Spring framework, you can wire beans automatically with auto-wiring feature. To enable it, just define the “**autowire**” attribute in .The Spring container can **autowire** relationships between collaborating beans without using and elements which helps cut down on the amount of XML configuration

|  |  |
| --- | --- |
| 1  2  3 | <bean id="countryBean" class="org.arpit.java2blog.Country" autowire="byName"> |

#### 16. What are different modes of autowiring supported by Spring?

There are following autowiring modes which can be used to instruct Spring container to use autowiring for dependency injection.

**no:**

Default, no auto wiring, set it manually via “ref” attribute as we have done in dependency injection via settor method post.

**byName:**

Autowiring by property name. Spring container looks at the properties of the beans on which *autowire* attribute is set to *byName* in the XML configuration file and it tries to match it with name of bean in xml configuration file.

**byType:**

Autowiring by property datatype. Spring container looks at the properties of the beans on which *autowire* attribute is set to *byType* in the XML configuration file. It then tries to match and wire a property if its **type** matches with exactly one of the beans name in configuration file. If more than one such beans exists, a fatal exception is thrown.

**contructor:**

byType mode in constructor argument.

**autodetect:**

Spring first tries to wire using autowire by *constructor*, if it does not work, Spring tries to autowire by *byType*.

#### 17. What is Spring AOP?

Aspect oriented Programming is programming paradigm which is analogous to object oriented programming. Key unit of object oriented programming is class, similarly key unit for AOP is Aspect. Aspect enable modularisation of concerns such as transaction management, it cut across multiple classes and types. It also refers as a crosscutting concerns.

#### 18. What is Aspect, Advice, Join point and pointcut in Spring AOP?

**Aspect:** An Aspect is a class that implements concerns that cut across different classes such as logging. It is just a name.

**Joint Point :**It is a point in execution of program such as execution of method. In Spring AOP, a join point always represents a method execution.

**Advice :**Action taken by  aspect at particular join point. For example: Before execution of getEmployeeName() method, put logging. So here, we are using before advice.

**Pointcut :** Pointcut is an expression that decides execution of advice at matched joint point. Spring uses the AspectJ pointcut expression language by default.

#### 19. What is @Qualifier annotation in Spring?

You can have more than one bean of same type in your XML configuration but you want to autowire only one of them ,so @Qualifier removes confusion created by @Autowired by declaring exactly which bean is to autowired.  
You can read about [Spring @Qualifier annotation](https://www.java2blog.com/2012/09/qualifier-annotation-in-spring.html) for more details.

#### 20. What is @Required annotation in Spring?

This annotation simply indicates that the affected bean property must be populated at configuration time: either through an explicit property value in a bean definition or through autowiring. The container will throw an exception if the affected bean property has not been populated; this allows for eager and explicit failure, avoiding NullPointerExceptions or the like later on.  
Suppose you have very large application and you get NullPointerExceptions because required dependency has not been injected then it is very hard to find out what goes wrong.So this annotation helps us in debugging the code.

**Question1: What is IOC or inversion of control?**([answer](http://javarevisited.blogspot.com/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html))

Answer: This *Spring interview question* is the first step towards Spring framework and many interviewers start Spring interview from this question. As the name implies **Inversion of control** means now we have inverted the control of creating the object from our own using new operator to container or framework. Now it’s the responsibility of container to create an object as required. We maintain one XML file where we configure our components, services, all the classes and their property. We just need to mention which service is needed by which component and container will create the object for us. This concept is known as [dependency injection](http://javarevisited.blogspot.com/2013/06/spring-helloworld-example-in-java-using-3.0-dependency-injection.html) because all object dependency (resources) is injected into it by the framework.

Example:

  <bean id="createNewStock" class="springexample.stockMarket.CreateNewStockAccont">   
        <property name="newBid"/>

  </bean>

In this example, CreateNewStockAccont class contain getter and setter for newBid and container will instantiate newBid and set the value automatically when it is used. This whole process is also called wiring in Spring and by using annotation it can be done automatically by Spring, referred as auto-wiring of bean in Spring.

**Question 2: Explain the Spring Bean-LifeCycle.**

Ans: Spring framework is based on IOC so we call it as IOC container also So Spring beans reside inside the IOC container. Spring beans are nothing but Plain old java object (POJO).

Following steps explain their life cycle inside the container.

1. The container will look the bean definition inside configuration file (e.g. bean.xml).

2 using reflection container will create the object and if any property is defined inside the bean definition then it will also be set.

3. If the bean implements the BeanNameAware interface, the factory calls setBeanName() passing the bean’s ID.  
4. If the bean implements the BeanFactoryAware interface, the factory calls setBeanFactory(), passing an instance of itself.  
5. If there are any BeanPostProcessors associated with the bean, their post-ProcessBeforeInitialization() methods will be called before the properties for the Bean are set.

6. If an init() method is specified for the bean, it will be called.  
7. If the Bean class implements the DisposableBean interface, then the method destroy() will be called when the Application no longer needs the bean reference.

8. If the Bean definition in the Configuration file contains a 'destroy-method' attribute, then the corresponding method definition in the Bean class will be called.  
To learn more about spring beans and their life cycle I also suggest you take a look at these [Spring Framework books](http://javarevisited.blogspot.com/2013/03/5-good-books-to-learn-spring-framework-mvc-java-programmer.html).

**Question 3: what is Bean Factory, have you used XMLBeanFactory?**

Ans: BeanFactory is factory Pattern which is based on IOC [design principles](http://javarevisited.blogspot.sg/2012/03/10-object-oriented-design-principles.html).it is used to make a clear separation between application configuration and dependency from actual code. The XmlBeanFactory is one of the implementations of Bean Factory which we have used in our project. The **org.springframework.beans.factory.xml.XmlBeanFactory is used to create bean instance defined in our XML file.**

BeanFactory factory = new XmlBeanFactory(new FileInputStream("beans.xml"));

Or

ClassPathResource resorce = new ClassPathResource("beans.xml");   
XmlBeanFactory factory = new XmlBeanFactory(resorce);

**Question 4: What are the difference between BeanFactory and ApplicationContext in Spring?**([answer](http://javarevisited.blogspot.com/2012/11/difference-between-beanfactory-vs-applicationcontext-spring-framework.html" \t "_blank))

Answer: This one is very popular Spring interview question and often asks in entry-level interview.ApplicationContext is the preferred way of using spring because of functionality provided by it and interviewer wanted to check whether you are familiar with it or not.

|  |  |
| --- | --- |
| **ApplicationContext.** | **BeanFactory** |
| Here we can have more than one config files possible | In this only one config file or .xml file |
| Application contexts can publish events to beans that are registered as listeners | Don't support. |
| Support internationalization (I18N) messages | It’s not |
| Support application life-cycle events, and validation. | Doesn’t support. |
| Supports many enterprise services such JNDI access, EJB integration, remoting | Doesn’t support. |

**Questions 4: What is the difference between @Controller and @RestController in Spring MVC?**([answer](http://javarevisited.blogspot.sg/2017/08/difference-between-restcontroller-and-controller-annotations-spring-mvc-rest.html))  
Even though both are used to indicate that a Spring bean is a Controller in Spring MVC setup, @RestController is better when you are developing RESTful web services using Spring MVC framework. It's a combination of @Controller + @ResponseBody annotation which allows the controller to directly write the response and bypassing the view resolution process, which is not required for RESTful web service.   
  
It also instructs DispatcherServlet to use different HttpMessageConverters to represent the response in the format client is expecting e.g. HttpMessageJackson2Convert to represent response in JSON format and JAXB based message converts to generate XML response. You can further see  **[REST with Spring](http://www.baeldung.com/rest-with-spring-course?utm_source=javarevisited&utm_medium=web&utm_campaign=rws&affcode=22136_bkwjs9xa" \t "_blank)** course by Baeldung to learn more about developing RESTful Web Services using Spring 4 and Spring 5.

**Question 6: What is the difference between singleton and prototype bean?**

Ans: This is another popular *spring interview questions* and an important concept to understand. Basically, a bean has scopes which define their existence on the application

**Singleton:** means single bean definition to a single object instance per Spring IOC container.  
**Prototype**: means a single bean definition to any number of object instances.

Whatever beans we defined in spring framework are singleton beans. There is an attribute in bean tag named ‘singleton’ if specified true then bean becomes singleton and if set to false then the bean becomes a prototype bean. By default, it is set to true. So, all the beans in spring framework are by default singleton beans.

  <bean id="createNewStock"     class="springexample.stockMarket.CreateNewStockAccont" **singleton=”false”**>  
        <property name="newBid"/>   
  </bean>

**Question 7: What is the role of DispatcherServlet in Spring MVC?**([answer](http://www.java67.com/2017/06/what-is-use-of-dispatcherservlet-in-spring-mvc.html))  
The DispatcherServlet is very important from Spring MVC perspective, it acts as a FrontController i.e. all requests pass through it. It is responsible for routing the request to controller and view resolution before sending the response to the client. When Controller returns a Model or View object, it consults all the view resolvers registered to find the correct type of ViewResolver which can render the response for clients.   
  
In case of RESTful Web Services, the DispatcherServlet is also responsible for using HttpMessageConverts to represent the response in the JSON, XML, or TEXT format, depending on the content negotiation between Client and Server e.g. if client send request with HTTP accept header as "application/json" then DispatcherServlet will ask the HttpMessageJackson2Converter to convert the response into JSON format.    
  
You can further see the free [Introduction to Spring MVC](https://pluralsight.pxf.io/c/1193463/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Fspringmvc-intro) course from Pluralsight to learn more about Spring MVC and DispatcherServlet.

**Question 8: What is AOP?**

Answer: The core construct of AOP is the aspect, which encapsulates behaviors affecting multiple classes into reusable modules. AOP is a programming technique that allows a developer to modularize crosscutting concerns,  that cuts across the typical divisions of responsibility, such as **logging and transaction management.**Spring AOP, aspects are implemented using regular classes or regular classes annotated with the @Aspect annotation. You can also check out these [Spring MVC interview questions](http://java67.blogspot.com/2012/08/spring-interview-questions-answers.html) for more focus on Java web development using Spring framework.

**Question 9: Explain Advice?**

Answer: It’s an implementation of aspect; advice is inserted into an application at join points. Different types of advice include “around,” “before” and “after” advice

**Question 10: What is joint Point and point cut?**

Ans: This is not really a spring interview questions I would say an AOP one.  Similar to [Object-oriented programming](http://javarevisited.blogspot.sg/2012/03/10-object-oriented-design-principles.html), AOP is another popular programming concept which complements OOPS. A join point is an opportunity within the code for which we can apply an aspect. In Spring AOP, a join point always represents a method execution.

**Pointcut**: a predicate that matches join points. A pointcut is something that defines at what join-points an advice should be applied.  
  
Here are few more Spring fundamental interview questions for practice  
  
**Question 11: Difference between the setter and constructor injection in Spring?** ([answer](http://javarevisited.blogspot.com/2012/11/difference-between-setter-injection-vs-constructor-injection-spring-framework.html))  
Setter injection is more flexible than constructor injection because you must remember the type and order of constructor parameter. Also, constructor injection is generally used to inject the mandatory dependency, while setter can be used to inject optional dependency.  
  
**Question 12: How to implement Role Based Access Control (RBAC) using Spring Security?**([answer](http://javarevisited.blogspot.com/2013/07/role-based-access-control-using-spring-security-ldap-authorities-mapping-mvc.html))  
Spring Security provides a couple of ways to implement Role based access control e.g. by using GrantedAuthority. See the article to learn more about it.   
  
**Question 13: How to call the stored procedure from Java using Spring Framework?** ([answer](http://javarevisited.blogspot.com/2013/04/spring-framework-tutorial-call-stored-procedures-from-java.html))  
  
**Question 14: How to Setup JDBC Database connection pool in Spring Web application?** ([answer](http://javarevisited.blogspot.com/2012/06/jdbc-database-connection-pool-in-spring.html))  
  
**Question 15: Difference between Factory Pattern and Dependency Injection in Java?**([answer](http://javarevisited.blogspot.com/2015/06/difference-between-dependency-injection.html))

Even though both allow you to reduce coupling in code, dependency injection is much more flexible and easier to test than Factory pattern.

**Question 16: What are different modules in spring?**

Answer: spring has seven core modules

1.      The Core container module

2.      Application context module

3.      AOP module (Aspect Oriented Programming)

4.      JDBC abstraction and DAO module

5.      O/R mapping integration module (Object/Relational)

6.      Web module

7.      MVC framework module

**Question 17: What type of transaction Management Spring support?**

Ans: This spring interview question is little difficult as compared to previous questions just because **transaction management** is a complex concept and not every developer familiar with it. Transaction management is critical in any applications that will interact with the database. The application has to ensure that the data is consistent and the integrity of the data is maintained.  Following two types of transaction management is supported by spring:

1. Programmatic transaction management

2. Declarative transaction management.

### ****Q1. What is Spring Framework?****

Spring is the most broadly used framework for the development of Java Enterprise Edition applications. The core features of Spring can be used in developing any Java application.

We can use its extensions for building various web applications on top of the Java EE platform, or we may just use its dependency injection provisions in simple standalone applications.

### ****Q2. What are the benefits of using Spring?****

Spring targets to make Java EE development easier. Here are the advantages of using it:

* **Lightweight:**there is a slight overhead of using the framework in development
* **Inversion of Control (IoC):** Spring container takes care of wiring dependencies of various objects, instead of creating or looking for dependent objects
* **Aspect Oriented Programming (AOP):**Spring supports AOP to separate business logic from system services
* **IoC container:** it manages Spring Bean life cycle and project specific configurations
* **MVC framework:** that is used to create web applications or RESTful web services, capable of returning XML/JSON responses
* **Transaction management:** reduces the amount of boiler-plate code in JDBC operations, file uploading, etc., either by using Java annotations or by Spring Bean XML configuration file
* **Exception Handling:**Spring provides a convenient API for translating technology-specific exceptions into unchecked exceptions

### ****Q3. What Spring sub-projects do you know? Describe them briefly.****

* **Core** – a key module that provides fundamental parts of the framework, like IoC or DI
* **JDBC** – this module enables a JDBC-abstraction layer that removes the need to do JDBC coding for specific vendor databases
* **ORM integration** – provides integration layers for popular object-relational mapping APIs, such as JPA, JDO, and Hibernate
* **Web** – a web-oriented integration module, providing multipart file upload, Servlet listeners, and web-oriented application context functionalities
* **MVC framework** – a web module implementing the Model View Controller design pattern
* **AOP module** – aspect-oriented programming implementation allowing the definition of clean method-interceptors and pointcuts

### ****Q4. What is Dependency Injection?****

Dependency Injection, an aspect of Inversion of Control (IoC), is a general concept stating that you do not create your objects manually but instead describe how they should be created. An IoC container will instantiate required classes if needed.

For more details, please refer [here](http://www.baeldung.com/inversion-control-and-dependency-injection-in-spring).

### ****Q5. How can we inject beans in Spring?****

A few different options exist:

* Setter Injection
* Constructor Injection
* Field Injection

The configuration can be done using XML files or annotations.

For more details, check [this article](http://www.baeldung.com/inversion-control-and-dependency-injection-in-spring).

### ****Q6. Which is the best way of injecting beans and why?****

The recommended approach is to use constructor arguments for mandatory dependencies and setters for optional ones. Constructor injection allows injecting values to immutable fields and makes testing easier.

### ****Q7. What is the difference between****BeanFactory****and****ApplicationContext****?****

BeanFactory is an interface representing a container that provides and manages bean instances. The default implementation instantiates beans lazily when getBean() is called.

ApplicationContext is an interface representing a container holding all information, metadata, and beans in the application. It also extends the BeanFactory interface but the default implementation instantiates beans eagerly when the application starts. This behavior can be overridden for individual beans.

For all differences, please refer to [the reference](https://docs.spring.io/spring/docs/current/spring-framework-reference/html/beans.html).

### ****Q8. What is a Spring Bean?****

The Spring Beans are Java Objects that are initialized by the Spring IoC container.

### ****Q9. What is the default bean scope in Spring framework?****

By default, a Spring Bean is initialized as a singleton.

### ****Q10. How to define the scope of a bean?****

To set Spring Bean’s scope, we can use @Scope annotation or “scope” attribute in XML configuration files. There are five supported scopes:

* **singleton**
* **prototype**
* **request**
* **session**
* **global-session**

For differences, please refer [here](https://docs.spring.io/spring/docs/3.0.0.M4/reference/html/ch03s05.html).

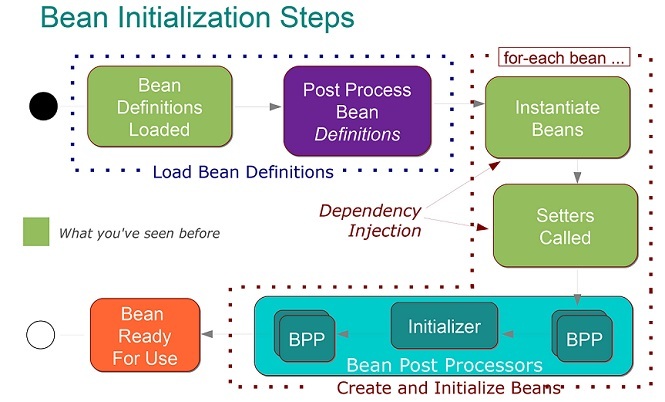
### ****Q11. Are singleton beans thread-safe?****

No, singleton beans are not thread-safe, as thread safety is about execution, whereas the singleton is a design pattern focusing on creation. Thread safety depends only on the bean implementation itself.

### ****Q12. What does the Spring bean lifecycle look like?****

First, a Spring bean needs to be instantiated, based on Java or XML bean definition. It may also be required to perform some initialization to get it into a usable state. After that, when the bean is no longer required, it will be removed from the IoC container.

The whole cycle with all initialization methods is shown on the image ([source](http://www.dineshonjava.com/2012/07/bean-lifecycle-and-callbacks.html)):

[](http://www.baeldung.com/wp-content/uploads/2017/06/Spring-Bean-Life-Cycle.jpg)

### ****Q13. What is the Spring Java-Based Configuration?****

It’s one of the ways of configuring Spring-based applications in a type-safe manner. It’s an alternative to the XML-based configuration.

Also, if you want to migrate your project from XML to Java config, please refer [to this article](http://www.baeldung.com/spring-xml-vs-java-config).

### ****Q14. Can we have multiple Spring configuration files in one project?****

Yes, in large projects, having multiple Spring configurations is recommended to increase maintainability and modularity.

You can load multiple Java-based configuration files:

|  |  |
| --- | --- |
| 1  2  3 | @Configuration  @Import({MainConfig.class, SchedulerConfig.class})  public class AppConfig { |

Or load one XML file that will contain all other configs:

|  |  |
| --- | --- |
| 1 | ApplicationContext context = new ClassPathXmlApplicationContext("spring-all.xml"); |

And inside this XML file you’ll have:

|  |  |
| --- | --- |
| 1  2 | <import resource="main.xml"/>  <import resource="scheduler.xml"/> |

### ****Q15. What is Spring Security?****

Spring Security is a separate module of the Spring framework that focuses on providing authentication and authorization methods in Java applications. It also takes care of most of the common security vulnerabilities such as CSRF attacks.

To use Spring Security in web applications, you can get started with a simple annotation: @EnableWebSecurity.

You can find the whole series of articles related to [security on Baeldung](http://www.baeldung.com/security-spring).

### ****Q16. What is Spring Boot?****

Spring Boot is a project that provides a pre-configured set of frameworks to reduce boilerplate configuration so that you can have a Spring application up and running with the smallest amount of code.

### ****Q17. Name some of the Design Patterns used in the Spring Framework?****

* **Singleton Pattern:** Singleton-scoped beans
* **Factory Pattern:** Bean Factory classes
* **Prototype Pattern:** Prototype-scoped beans
* **Adapter Pattern:** Spring Web and Spring MVC
* **Proxy Pattern:** Spring Aspect Oriented Programming support
* **Template Method Pattern:** JdbcTemplate, HibernateTemplate, etc.
* **Front Controller:** Spring MVC DispatcherServlet
* **Data Access Object:** Spring DAO support
* **Model View Controller:**Spring MVC

### ****Q18. How does the scope****Prototype****work?****

Scope prototype means that every time you call for an instance of the Bean, Spring will create a new instance and return it. This differs from the default singleton scope, where a single object instance is instantiated once per Spring IoC container.

## **3. Spring MVC**

### ****Q19. How to Get****ServletContext****and****ServletConfig****Objects in a Spring Bean?****

You can do either by:

1. Implementing Spring-aware interfaces. The complete list is available [here](http://www.buggybread.com/2015/03/spring-framework-list-of-aware.html).
2. Using @Autowired annotation on those beans:

|  |  |
| --- | --- |
| 1  2  3  4  5 | @Autowired  ServletContext servletContext;    @Autowired  ServletConfig servletConfig; |

### ****Q20. What is the role of the****@Required****annotation?****

The @Required annotation is used on setter methods, and it indicates that the bean property that has this annotation must be populated at configuration time. Otherwise, the Spring container will throw a BeanInitializationException exception.

Also, @Required differs from @Autowired – as it is limited to a setter, whereas @Autowired is not. @Autowired can be used to wire with a constructor and a field as well, while @Required only checks if the property is set.

Let’s see an example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | public class Person {      private String name;        @Required      public void setName(String name) {          this.name = name;      }  } |

Now, the name of the Person bean needs to be set in XML config like this:

|  |  |
| --- | --- |
| 1  2  3 | <bean id="person" class="com.baeldung.Person">      <property name="name" value="Joe" />  </bean> |

Please note that @Required doesn’t work with Java based @Configuration classes by default. If you need to make sure that all your properties are set, you can do so when you create the bean in the @Bean annotated methods.

### ****Q21. What is the role of the****@Autowired****annotation?****

The @Autowired annotation can be used with fields or methods for injecting a bean by type. This annotation allows Spring to resolve and inject collaborating beans into your bean.

For more details, please refer [to this tutorial](http://www.baeldung.com/spring-autowire).

### ****Q22. What is the Role of the****@Qualifier****Annotation?****

It is used simultaneously with the @Autowired annotation to avoid confusion when multiple instances of a bean type are present.

Let’s see an example. We declared two similar beans in XML config:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <bean id="person1" class="com.baeldung.Person" >      <property name="name" value="Joe" />  </bean>  <bean id="person2" class="com.baeldung.Person" >      <property name="name" value="Doe" />  </bean> |

When we try to wire the bean, we’ll get an org.springframework.beans.factory.NoSuchBeanDefinitionException. To fix it, we need to use @Qualifier to tell Spring about which bean should be wired:

|  |  |
| --- | --- |
| 1  2  3 | @Autowired  @Qualifier("person1")  private Person person; |

### ****Q23. How to handle exceptions in Spring MVC environment?****

There are three ways to handle exceptions in Spring MVC:

1. **Using @ExceptionHandler at controller level** – this approach has a major feature – the @ExceptionHandler annotated method is only active for that particular controller, not globally for the entire application
2. **Using HandlerExceptionResolver** – this will resolve any exception thrown by the application
3. **Using @ControllerAdvice**– Spring 3.2 brings support for a global @ExceptionHandler with the @ControllerAdvice annotation, which enables a mechanism that breaks away from the older MVC model and makes use of ResponseEntity along with the type safety and flexibility of @ExceptionHandler

For more detailed information on this topic, go through [this writeup](http://www.baeldung.com/exception-handling-for-rest-with-spring).

### ****Q24. How to validate if the bean was initialized using valid values?****

Spring supports [JSR-303](http://beanvalidation.org/1.0/spec/)annotation-based validations. JSR-303 is a specification of the Java API for bean validation, part of JavaEE and JavaSE, which ensures that properties of a bean meet specific criteria, using annotations such as @NotNull, @Min, and @Max. The article regarding JSR-303 is available [here](http://www.baeldung.com/javax-validation).

What’s more, Spring provides the Validator interface for creating custom validators. For example, you can have a look [here](http://www.baeldung.com/spring-mvc-custom-validator).

### ****Q25. What is Spring MVC Interceptor and how to use it?****

Spring MVC Interceptors allow us to intercept a client request and process it at three places – before handling, after handling, or after completion (when the view is rendered) of a request.

The interceptor can be used for cross-cutting concerns and to avoid repetitive handler code like logging, changing globally used parameters in Spring model, etc.

For details and various implementations, take a look [at this series](http://www.baeldung.com/spring-mvc-handlerinterceptor).

### ****Q26. What is a Controller in Spring MVC?****

Simply put, all the requests processed by the DispatcherServlet are directed to classes annotated with @Controller. Each controller class maps one or more requests to methods that process and execute the requests with provided inputs.

If you need to take a step back, we recommend having a look at the concept of the [Front Controller in the typical Spring MVC architecture](http://www.baeldung.com/spring-controllers).

## **4. Spring Web**

### ****Q27. How does the****@RequestMapping****annotation work?****

The @RequestMapping annotation is used to map web requests to Spring Controller methods. In addition to simple use cases, we can use it for mapping of HTTP headers, binding parts of the URI with @PathVariable, and working with URI parameters and the @RequestParam annotation.

More details on @RequestMapping are available [here](http://www.baeldung.com/spring-requestmapping).

### ****Q28. What’s the Difference Between****@Controller****,****@Component****,****@Repository,****and****@Service****Annotations in Spring?****

According to the official Spring documentation, @Component is a generic stereotype for any Spring-managed component. @Repository, @Service, and @Controller are specializations of @Componentfor more specific use cases, for example, in the persistence, service, and presentation layers, respectively.

Let’s take a look at specific use cases of last three:

* **@Controller** – indicates that the class serves the role of a controller, and detects @RequestMapping annotations within the class
* **@Service** – indicates that the class holds business logic and calls methods in the repository layer
* **@Repository** – indicates that the class defines a data repository; its job is to catch platform-specific exceptions and re-throw them as one of Spring’s unified unchecked exceptions

### ****Q29. What are****DispatcherServlet****and****ContextLoaderListener****?****

Simply put, in the Front Controller design pattern, a single controller is responsible for directing incoming HttpRequests to all of an application’s other controllers and handlers.

**Spring’s DispatcherServlet implements this pattern and is, therefore, responsible for correctly coordinating the HttpRequests to the right handlers.**

On the other hand, ContextLoaderListener starts up and shuts down Spring’s root WebApplicationContext. It ties the lifecycle of ApplicationContext to the lifecycle of the ServletContext. We can use it to define shared beans working across different Spring contexts.

For more details on DispatcherServler, please refer [to this tutorial](http://www.baeldung.com/spring-dispatcherservlet).

### ****Q30. What is****ViewResolver****in Spring?****

The ViewResolver enables an application to render models in the browser – without tying the implementation to a specific view technology – by mapping view names to actual views.

For a guide to the ViewResolver, have a look [here](http://www.baeldung.com/spring-mvc-view-resolver-tutorial).

### ****Q31. What is a****MultipartResolver****and when is it used?****

The MultipartResolver interface is used for uploading files. The Spring framework provides one MultipartResolver implementation for use with Commons FileUpload and another for use with Servlet 3.0 multipart request parsing.

Using these, we can support file uploads in our web applications.

## **5. Spring Data Access**

### ****Q32. What is Spring****JDBCTemplate****class and how to use it?****

The Spring JDBC template is the primary API through which we can access database operations logic that we’re interested in:

* creation and closing of connections
* executing statements and stored procedure calls
* iterating over the ResultSet and returning results

To use it, we’ll need to define the simple configuration of DataSource:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | @Configuration  @ComponentScan("org.baeldung.jdbc")  public class SpringJdbcConfig {      @Bean      public DataSource mysqlDataSource() {          DriverManagerDataSource dataSource = new DriverManagerDataSource();          dataSource.setDriverClassName("com.mysql.jdbc.Driver");          dataSource.setUrl("jdbc:<mysql://localhost:3306/springjdbc>");          dataSource.setUsername("guest\_user");          dataSource.setPassword("guest\_password");            return dataSource;      }  } |

For further explanation, you can go through [this quick article](http://www.baeldung.com/spring-jdbc-jdbctemplate).

### ****Q33. How would you enable t****ransactions****in Spring and what are their benefits?****

There are two distinct ways to configure Transactions – with annotations or by using Aspect Oriented Programming (AOP) – each with their advantages.

The benefits of using Spring Transactions, according to the [official docs](http://docs.spring.io/spring/docs/current/spring-framework-reference/html/transaction.html), are:

* Provide a consistent programming model across different transaction APIs such as JTA, JDBC, Hibernate, JPA, and JDO
* Support declarative transaction management
* Provide a simpler API for programmatic transaction management than some complex transaction APIs such as JTA
* Integrate very well with Spring’s various data access abstractions

### ****Q34. What is Spring DAO?****

Spring Data Access Object is Spring’s support provided to work with data access technologies like JDBC, Hibernate, and JPA in a consistent and easy way.

You can, of course, go more in-depth on persistence, with the [entire series](http://www.baeldung.com/persistence-with-spring-series/) discussing persistence in Spring.

## **6. Spring Aspect-Oriented Programming (AOP)**

### ****Q35. What is Aspect-Oriented Programming?****

Aspects enable the modularization of cross-cutting concerns such as transaction management that span multiple types and objects by adding extra behavior to already existing code without modifying affected classes.

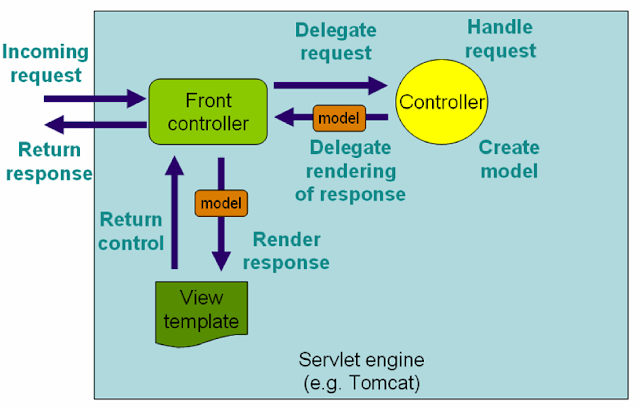
Here is the example of [aspect-based execution time logging](http://www.baeldung.com/spring-aop-annotation).

### ****Q36. What are****Aspect****,****Advice****,****Pointcut,****and****JoinPoint****in AOP?****

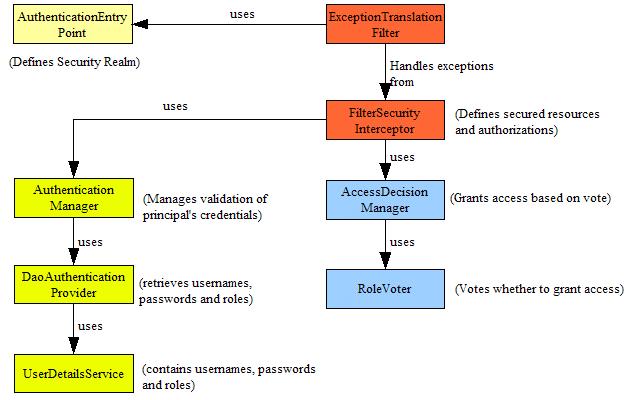
* ***Aspect***: a class that implements cross-cutting concerns, such as transaction management
* **Advice**: the methods that get executed when a specific JoinPoint with matching Pointcut is reached in the application
* ***Pointcut***: a set of regular expressions that are matched with JoinPoint to determine whether Advice needs to be executed or not
* ***JoinPoint***: a point during the execution of a program, such as the execution of a method or the handling of an exception

### ****Q37. What is****Weaving****?****

According to the [official docs](https://docs.spring.io/spring/docs/current/spring-framework-reference/html/aop.html), weaving is a process that links aspects with other application types or objects to create an advised object. This can be done at compile time, load time, or at runtime. Spring AOP, like other pure Java AOP frameworks, performs weaving at runtime

**1) What is spring framework? Why Java programmer should use Spring framework**  
Very common Spring interview question, Spring is a framework which helps Java programmer in development. Spring provides Dependency Injection and IOC container, Spring MVC flow and several useful API for Java programmer.  
  
 **2) What is default scope of bean in Spring framework? (**[**answer**](http://javarevisited.blogspot.sg/2012/05/what-is-bean-scope-in-spring-mvc.html)**)**  
The default scope of a Spring bean is the Singleton scope and in the web application default scope of a spring bean is request scope. Singleton bean means the same instance of a bean is shared with all other beans, while request scope means a bean is alive only for a request.  
  
  
  
**3) Does Spring singleton beans are thread-safe?**  
No, Spring singleton beans are not thread-safe. Singleton doesn't mean bean would be[thread-safe](http://javarevisited.blogspot.sg/2012/01/how-to-write-thread-safe-code-in-java.html).  
  
  
**4) What is Dependency Injection?**  
Dependency Injection is one of the design pattern, which allows injecting dependency on Object, instead of object resolving the dependency.  
  
  
**5) What is Inversion of Control concept, how does Spring support IOC?** ([answer](http://javarevisited.blogspot.sg/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html))  
The simple meaning of inversion of the control means that now the framework, Spring is responsible for creating objects, wiring dependencies and managing their life-cycle instead of a developer, which was the case before. That's where control is inverted from developer to framework.  
  
  
**6) What is Spring MVC? Can you explain How one request is processed?** ([answer](http://javarevisited.blogspot.sg/2017/06/how-spring-mvc-framework-works-web-flow.html))  
Spring MVC is a framework to develop Java web application. It provides an implementation of MVC or Model View Controller architecture which is built on separation of concerns and makes the development of Java web application easy. In order to use this in your project, you need to learn Spring and include its JAR file.  
  
  
**7) How do you create a controller in Spring? @Controller vs @RestController?**([answer](http://javarevisited.blogspot.sg/2017/08/difference-between-restcontroller-and-controller-annotations-spring-mvc-rest.html))  
A controller is nothing but a class, also known as a bean in Spring terminology. If you are using annotation then you can create controller by using @Controller annotation. If you are developing RESTful web service then you can also create REST controllers by using @RestController annotation. See the answer for more detailed discussion.  
  
  
**8) What is view Resolver pattern? how it works in Spring MVC**  
View Resolver pattern is a J2EE pattern which allows a web application to dynamically choose it's view technology e.g. HTML, JSP, Tapestry, JSF, XSLT or any other view technology.  
  
In this pattern, View resolver holds mapping of different views, controller return name of the view, which is then passed to View Resolver for selecting an appropriate view.  
  
Spring MVC framework also supplies inbuilt view resolver for selecting views. See [Spring Master Class - Beginner to Expert](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-tutorial-for-beginners%2F) to learn more about how Spring MVC internally works.  


**9) What is Spring Security?**  
Spring security is a project under spring framework umbrella, which provides support for security requirements of enterprise Java projects.  
  
Spring Security formerly known as aegis security provides out of box support for creating login screen, remember me cookie support, securing URL, authentication provider to authenticate the user from the database, LDAP and in memory, concurrent active session management support and much more.  
  
In order to use Spring security in a Spring MVC based project, you need to include spring-security.jar and configure it in application-Context-security.XML file, you can name it whatever you want, but make sure to supply this to ContextLoaderListener, which is responsible for creating Spring context and initializing dispatcher servlet.  
  
You can further see [Learn Spring Security Certification Class](https://courses.baeldung.com/p/learn-spring-security-the-certification-class?utm_source=javarevisited&utm_medium=web&utm_campaign=lss&affcode=22136_bkwjs9xa) learn more about authentication, authorization, and session control using Spring Security.



**10) How do you control concurrent Session on Java web application using Spring Security?**  
You can use Spring Security to control a number of active session in Java web application. Spring security framework provides this feature out of the box and when enabled, a user can only have one active session at a time. See this Spring Security example to learn more about [How to control concurrent user session using Spring security](http://javarevisited.blogspot.sg/2012/03/spring-security-example-tutorial-how-to.html)  
 **11) What types of dependency injection is supported by Spring Framework? When do you use Setter and Constructor Injection, pros and cons?**  
There are 2 types of dependency injection supported by Spring, constructor based injection, and setter-based injection.  
  
Both types have their own advantages and disadvantages, you should use Constructor injection when object's dependencies are not optional and they must be initialized with their dependencies.  
  
Also, use constructor injection if the order of initialization or dependency matters because in Setter based injection you cannot impose any order. Use setter injection when dependencies are optional. See the [difference between setter and constructor injection in Spring](http://javarevisited.blogspot.sg/2012/11/difference-between-setter-injection-vs-constructor-injection-spring-framework.html) for a more detailed answer.  
  
  
  
12) What is the difference between ApplicationContext and BeanFactory in Spring framework? ([answer](http://javarevisited.blogspot.sg/2012/11/difference-between-beanfactory-vs-applicationcontext-spring-framework.html))  
  
  
13) How do you call a stored procedure by using Spring framework? ([answer](http://javarevisited.blogspot.sg/2013/04/spring-framework-tutorial-call-stored-procedures-from-java.html))  
  
  
14) What do JdbcTemplate and JmsTemplate class offer in Spring?  
  
  
15) Can we use more than one configuration file for our Spring project?  
  
  
16) Explain Spring MVC flow with a simple example e.g. starting from Container receives a request and forward to your Java application? ([answer](http://javarevisited.blogspot.sg/2017/06/how-spring-mvc-framework-works-web-flow.html))  
  
  
  
**17) What is the difference in Spring MVC and Spring core?**  
The Spring MVC is part of Spring framework which helps you to develop Java web application using model web controller pattern, while Spring Core provides the Dependency injection and Inversion of Control. The Spring Container is part of Spring core.  
  
Both functionalities come in different JAR files. If you are developing just a core Java application using Spring then you just need Spring Core but if you are developing Web application then you need spring-mvc.jar as well. See [Introduction to Spring MVC](https://pluralsight.pxf.io/c/1193463/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Fspringmvc-intro) to learn more about Spring framework architecture and components.  
  
18) Can you use Spring MVC framework along with Struts? I have an existing Java MVC application which is based in Struts, Can I migrate that to use Spring MVC? How?  
  
  
19) What is the advantage of Spring MVC framework over Struts 1.0 or Struts 2.0 ? is it worth to convert an existing Struts application to Spring MVC?  
  
  
20) How does Spring resolve view returned by ModelAndView class?  
  
Some Spring MVC questions are tricky e.g. Struts and Spring integration and can be only answered by experienced Java program with 2 to 4-year experience in Spring MVC framework.  
  
  
**21) If a user checked in CheckBox and got a validation error in other fields and then he unchecked the CheckBox, what would be selection status in command object in Spring MVC? How do you fix this issue?**  
Since during HTTP post, if the checkbox is unchecked than HTTP does include a request parameter for checkbox, which means updated selection won't be picked up. you can use hidden form field, starting with \_ to fix this in Spring MVC. quite a tricky question to answer if you are not aware of HTTP POST behavior and Spring MVC.  
  
  
**22) What are different implementations of View interface you have used in Spring MVC?**  
ULBased View e.g. JSP, JSTLView,

**1)  What is a spring?**

Spring is set to be a framework which helps Java programmer for development of code and it provides IOC container, Dependency Injector, MVC flow and many other APIs for the java programmer.

**2) What are Advices in Spring?**

It is the execution of an aspect. Advice is like making your application learn a new trick. They are usually introduced at joinpoints.

**3) What is the default scope of bean in Spring framework?**

The default scope of bean is Singleton for Spring framework.

**4) Name the types of transaction management that are supported by Spring?**

Transaction management supported by Spring are :

* Declarative transaction management.
* Programmatic transaction management.

**5) Is Singleton beans are thread safe in Spring Framework?**

No, singleton beans are not thread-safe in Spring framework.

**6) What are the benefits of Spring Framework?**

Following are the benefits of Spring framework:

* Extensive usage of Components
* Reusability
* Decoupling
* Reduces coding effort by using pattern implementations such as singleton, factory, service locator etc.
* Removal of leaking connections
* Declarative transaction management
* Easy to integrate with third party tools and technologies.

**7) What is Bean Factory?**

Bean Factory is core of the spring framework and, it is a Lightweight container which loads bean definitions and manages your beans.  Beans are configured using XML file and manage singleton defined bean. It is also responsible for life cycle methods and injects dependencies. It also removes adhoc singletons and factories.

**8) Define Bean Wiring?**

Bean wiring is the creation of associations between application components that are between the beans in a particular spring container.

**9) What is called Spring MVC?**

A Spring MVC is a single shared controller instance and it is used to handle request type controllers, interceptors which run in the IoC container. It also allows multiple Dispatcher Servlets which can share application context interface but not class based interface.

**10) Why Spring framework is needed?**

Spring framework is needed because it is –

* Very Light Weight Container
* Framework
* IOC
* AOP

**11) Name the various modules used in spring framework?**

* AOP module (Aspect Oriented Programming)
* JDBC abstraction and DAO module
* The Core container module
* MVC framework module
* Application context module
* O/R mapping integration module (Object/Relational)
* Web module

**12) Explain the RowCallbackHandler in Spring?**

The RowCallbackHandler is called for each row in ResultSet and is used to read values from the ResultSet.

**13) Define Application context module?**

This is a very important module and supplies various necessary services like EJB integration, remoting, JNDI access and scheduling. It transforms spring into a framework. It also broadens the idea of BeanFactory by application of lifecycle events, providing support for internationalization messages and validation.

**14) Write about AOP module?**

AOP module is utilized for creating aspects for Spring applications. It also enables support for metadata programming in Spring.

**15) What is a BeanFactory Interface?**

Bean factory interface is used to provide configuration framework for object creation and basic functionality around object management.

**16) State the differences between ApplicationContext and BeanFactory in spring?**

* ApplicationContext allows more than one config files to exist while BeanFactory only permits one.
* ApplicationContext can print events to beans registered as listeners. This feature is not supported by BeanFactory.
* ApplicationContext also provides support for application of lifecycle events, internationalization messages and validation and also provides services like EJB integration, remoting, JNDI access and scheduling. These features too are not supported by Bean Factory.

**17) What is Auto Wiring?**

Autowiring is used to build relationships between the collaborating beans. Spring container can automatically resolve collaborators for beans.

**18) What are the different Modes of Autowiring?**

Autowiring has five different modes:

* no: no autowire
* byName : Autowiring that can be done by property name
* byType : property type as autowired
* constructor: It is similar to byType and it is property is in constructor
* autodetect :  Spring is allowed to select autowiring from byType or constructor

**19) How to start using spring?**

Following steps needs to be done to start with the Spring:

* Download Spring and its dependent file from spring’s site.
* Create application context xml to define beans and its dependencies
* Integrate application context xml with web.xml
* Deploy and Run the application

**20) What are the methods of bean life cycle?**

There are two important methods of Bean life cycle:

* Setup – called when bean is loaded into container
* Teardown – called when bean is unloaded into container

**21) What are the different types of events of Listeners?**

Following are the different types of events of listeners:

* ContextClosedEvent – This event is called when the context is closed.
* ContextRefreshedEvent – This event is called when context is initialized or refreshed
* RequestHandledEvent – This event is called when the web context handles request

**22) Differentiate between singleton and prototype bean?**

Singleton means only one bean is defined per object instance while Prototype means one definition to more than one object instances in Spring.

**23) What are the types of Dependency Injection?**

Two types of dependency injection are supported by spring framework:

* Setter Injection
* Constructor Injection

**24) Write about Core container module?**

Core container module is responsible for the basic functionality of the spring framework. The whole Spring framework is built with this module as a base.

**25) What is AOP module?**

This AOP module is used for spring enabled application. Support has been provided AOP alliance to ensure the interoperability between spring and other AOP frameworks.

It instructs spring to add annotations to the source code and tell how to apply aspects.

**26) What is AOP Alliance?**

AOP alliance is an open-source project which is aimed at promoting adoption of AOP. The AOP alliance’s goal is to define a common set of components and interfaces so as to improve interoperability among different AOP implementations.

**27) What is called spring configuration file?**

Spring configuration file is an XML file and it contains class information. It also describes how these classes are configured and interact with each other.

**28) What are different types of Autowire?**

There are four different types of Auto wire:

* byName
* byType
* constructor
* autodetect

**29) What are the types of the transaction management that is supported by spring?**

Following are the types of transaction management that has been supported by spring:

* declarative
* programmatically

**30) When are declarative and programmatic transaction management used?**

When only a small amount of transactional operations is there, it is advised to use Programmatic transaction management. But if there is a big amount of transactional operations to be taken care of, declarative transaction management is preferred.

**31) What is IOC?**

IOC (Inversion of Control pattern) is also known as dependency injection. IOC directs the programmers to depict how to create objects instead of actually creating them. But in this design pattern, this control has been given to assembler and assembler will instantiate required class if needed.

**32) Write about the different types of Listener related events?**

The different types of events related to listeners are:

* ContextRefreshedEvent – This gets called when the context is refreshed or initialized.
* RequestHandledEvent – This gets called when the web context is handling a request.
* ContextClosedEvent – This gets called when the context gets closed.

**33) What is an Aspect?**

Aspect is also called as logging which is required throughout the application. Logging or aspect is a cross cutting functionality in an application using AOP.

**34) What is a Joinpoint?**

The point where an aspect can be introduced in the application is known as a joinpoint. This point could be a field being modified, a method being called or even an exception being thrown. At these points, the new aspect’s code can be added to introduce a new behavior to the application.

Aspect code can be inserted at this point into normal flow of application to change the current behavior.

**35) What is called an Advice?**

Advice will tell application on new behavior and it is the implementation of an aspect. It is inserted into an application at the joinpoint.

Advice is the implementation of an aspect. It is something like telling your application of a new behavior. Generally, the advice is inserted into an application at joinpoints.

**36) What is a Pointcut?**

Pointcut is used to allow where the advice can be applied.

**37) What is weaving?**

Weaving is used to create new proxy object by applying aspects to target object.

**38) What is difference between singleton and prototype bean?**

Singleton Bean – Single bean definition to a single object instance per Spring IOC container

Prototype Bean – Single bean definition to any number of object instances per Spring IOC Container

**39) In what points, can weaving be applied?**

Following are the points where weaving can be applied:

* Compile Time
* Class load Time
* Runtime

**40) What are the different types of AutoProxying?**

Following are the different types of AutoProxying:

* BeanNameAutoProxyCreator
* DefaultAdvisorAutoProxyCreator
* Metadata autoproxying

**41) How can beans be made singleton or prototype?**

The bean tag has an attribute called ‘singleton’. The bean is singleton if its value is ‘TRUE’, otherwise the bean is a prototype.

**42) What classes are used to Control the database connection?**

Following are the classes that are used to control database connection:

* Data Source Utils
* SmartData Source
* AbstractData Source
* SingleConnection DataSource
* DriverManager DataSource
* TransactionAware DataSourceProxy
* DataSource  TransactionManager

**43) Describe about DAO in Spring framework?**

DAO is used to provide integration of Java database connectivity and Object relational mapping objects. DAO is spring framework provides connection for JDBC, hibernate, JDO, JPA, Common client interface and Oracle.

**44) What is Autoproxying?**

Autoproxying is used to create proxy automatically for the spring users. It provides following two classes to support this automatic proxy creation:

* BeanNameAutoProxyCreator
* DefaultAdvisorAutoProxyCreator

**45) What is Metadata Autoproxying?**

Metadata Autoproxying can be performed inspiring which can be driven by metadata. This is determined by source level attributes and keeps metadata inside the source code.

This maintains metadata in one place and mainly used for declarative transaction support.

**46) What is ‘Throws advice’ in Spring?**

‘Throws Advice’ define the behavior when an exception occurs. It is an interface and it has no methods which need to be implemented.

A class that implements this interface should have method with this signature:

* Void samplethrow (Throw table t)
* Void samplethrow(Method m, Object[] o, Object target, Throw tablet)

**47) What are the various editors used in spring work?**

The various custom editors provided by the Spring Framework are:

* PropertyEditor
* URLEditor
* ClassEditor
* CustomDateEditor
* FileEditor
* LocaleEditor
* StringArrayPropertyEditor
* StringTrimmerEditor

**48)  What are the advantages of spring framework?**

Following are the advantages of spring framework:

* Layered Architecture
* Enables Plain Old Java Object (POJO) Programming and it enables continuous integration and testability
* Dependency Injection and Inversion of Control that simplifies JDBC
* Open source framework which can be used for commercial purpose

**49) How is Hibernate accessed using the Spring framework?**

Hibernate can be accessed in the following two ways:

* By IOC with a Callback and HibernateTemplate.
* By applying an AOP Interceptor and broadening the HibernateDaoSupport.

**50) What are the various Channels supported by Spring 2.0?**

Following are the channels supported by spring version 2.0:

* Pollable Channel
* Subscribable Channel
* PublishSubscribe Channel
* Queue Channel
* Priority Channel
* Rendezvous Channel
* Direct Channel
* Executor Channel
* Scoped Channel

**51) Why is declarative transaction management preferred in Spring?**

Declarative transaction management has minimum impact on the application code and, therefore, is an idealistic lightweight container.

**52) Explain the concept of a BeanFactory?**

BeanFactory applies the idea of a factory pattern that utilizes IOC to separate the application’s dependencies and configuration from the actual code.

**53) What are the different scopes of spring bean?**

Scopes of spring bean are Singleton, prototype, request, session and global session.

**54) What are all the ways to access Hibernate by using Spring?**

There are two ways to access hibernate using spring:

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

**55) How struts application can be integrated with spring?**

There are two options for struts application that can be integrated with spring:

Configuration of Spring to manage beans using ContextLoader plugin and set their dependencies in a spring context file

Grab spring managed beans explicitly using agetwebapplicationcontext()

**56) What is Inversion of control (IOC)?**

Inversion of Control (IOC) is also called as dependency Injection which is nothingbut a design pattern that gives control to the assembler of classes. In general, class will instantiate another class if required.

But in this design pattern, this control has been to given to assembler and assembler will instantiate required class if needed.

**57) Write the benefits of using IOC?**

The major benefits of dependency injection or IOC are that it reduces the amount of coding required for the application. This allows the testing of the application to be done quickly and easily as no JNDI lookup mechanism or singletons are required. IOC containers also support lazy loading and eager installation of services.

**58) What is Inner bean? What is the drawback of inner bean?**

If a bean element is directly embedded in a property tag while wiring beans, then the bean is called Inner Bean. Its drawback is that it cannot be reprocessed.

**59)What are the different types of Injection in spring?**

There are three types of Injection in spring:

* Setter Injection
* Constructor Injection
* Getter or Method Injection

**60) What are the benefits of spring framework?**

Following are the benefits of spring framework:

* Light weight container when compared to j2EE containers
* Built in Web MVC framework
* Creates loosely coupled applications
* Supports aspect oriented programming like logging, transaction and security
* Configuration done in XML format which is easy to write and understand

**61) What are the types of Advice?**

There are five types of Advice:

* Before Advice
* After returning advice
* After throwing advice
* Finally advice
* Around advice

**62) What is called PreparedStatementCreator?**

PreparedStatementCreator is one of the most commonly used interfaces for writing data to the database. createPreparedStatement() is a method that can be used to create and return PreparedStatement from the Connection argument, and exception handling is automatically taken care of. When this interface is implemented, a different interface SqlProvider can also be implemented which has a method called getSql(). This method is useful for providing sql strings to the JdbcTemplate. It does not handle SQLExceptions.

**63) What is SQLProvider?**

SQLProvider has only one method called getSql()and it is implemented using PreparedStatementCreator implementers. It is mainly used for debugging.

**64) Write about BatchPreparedStatementSetter?**

BatchPreparedStatementSetter is used to update more than a single row in one go, they can use BatchPreparedStatementSetter. This interface provides two methods they are

* setValues( PreparedStatement ps, int i) throws SOL exception
* int getBatchSize

**65) What is the better method of using JDBC in Spring?**

If JDBC is used with the template class called JdbcTemplate, it gives a better performance.

**66) What exceptions do the DAO classes, use in Spring throw?**

In spring DAO classes only throws SQLException.

**67) Explain the advantages of using DAO module?**

The database code can be kept clean and simple by using the DAO module. This helps in preventing problems that rise because of poor handling of closures of database resources. Also, the DAO module utilizes the AOP module to enable objects in the Spring application to use transaction management services.

**68) Name the significant ApplicationContext implementations used in the spring framework?**

They are:

* ClassPathXmlApplicationContext
* FileSystemXmlApplicationContext
* XmlWebApplicationContext

**69) How is a bean added to a Spring application?**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <?xml version="1.0" encoding="UTF-8"?>    *<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN">*    <beans>    <bean id="foo"/>    <bean id="bar"/>    </beans> |

The bean tag has an ID attribute which stores the bean name and a class attributes which specifies the full class name.

**70) What are ORM integration modules?**

Object/relational mapping (ORM) tool is supported by Spring over straight JDBC by implementing the ORM module. Spring can join various important ORM frameworks, including JDO, iBATIS SQL Maps and Hibernate.

**71) Mention and explain the types of Advice in Spring?**

Types of advice are:

* Before advice: Advice that is executed prior to a joinpoint is called the ‘before advice’.
* After returning advice: Advice that is executed after the normal completion of a joinpoint is called the ‘after returning advice’.
* After throwing advice: Advice that is executed only if a method exits abnormally by throwing an exception, is called the ‘after throwing advice’.
* After (finally) advice: Advice that is executed irrespective of how a joinpoint exits is called ‘after finally advice’.
* Around advice: Advice that borders a joinpoint, for example, a method invocation, is called an ‘around advice’. This can be used to perform special activities before and after the invocation of method.

**72) What is the web module?**

The web module enables the creation of a web application without XML. The web.xml file needs to be configured for using the web module.

**73) What is DataAccessException?**

DataAccessException is a RuntimeException. It is an Unchecked Exception. The user cannot be forced to handle these kinds of exceptions.

**74) What is XMLBeanFactory?**

Spring includes several applications of Bean factory. Out of these, org.springframework.beans.factory.xml.XmlBeanFactory is a very important one. It loads the beans on the basis of the definitions stored in an XML file. For the creation of an XmlBeanFactory, a java.io.InputStream is passed to the constructor. The InputStream provides the XML to the factory. For example, for retrieval of the bean, the getBean() method is called by passing the name of the desired bean.



|  |  |
| --- | --- |
| 1 | MyBean helloBean = (MyBean) factory.getBean("helloBean"); |

**75) Name the Exception class which is connected to the exceptions thrown by the applications?**

It is the DataAccessException given by org.springframework.dao.DataAccessException

**76) Mention the types of lOC (dependency injection)?**

The different types of loC are: –

* Setter Injection: With the help of JavaBeans properties.
* Constructor Injection: Dependencies are given in the form of constructor parameters.
* Interface Injection: With the help of an interface, an Injection is performed.

Out of these three, only construction and setter are being used in Spring.

**77) What are the important beans lifecycle methods?**

All in all, two bean lifecycle methods are there. The first method is the setup method which is called during the loading of the bean into the container. The second is when the bean is unloaded from the container, and this method is called the teardown.

**78) How can the default lifecycle methods of beans be nullified?**

The tag, bean, has two useful attributes which can be used to define special initialization and destruction methods.

For Example, two new methods forSetup and forTeardown can be added to the Foo class in the following way:



|  |  |
| --- | --- |
| 1  2  3  4  5 | <beans>    <bean id="bar" init-method=”forSetup” destroy=”forTeardown”/>    </beans> |

**79) What is a Target?**

A target is the class that is advised. This class can either be a class to which we want to add a special behavior to or a third party class. The target class is free to center on its major concern using the AOP concepts, regardless of any advice that is being applied.

**80) Explain the term Proxy?**

The term proxy refers to an object which is produced the application of an advice to the target object.

**81) What is cross cutting concern and concern in spring AOP?**

Cross cutting concern:  It is a concern which is applicable throughout the application and it affects the       entire application. E.g Security, logging and data transfer are the concerns which are needed in almost every module of an application.

Concern: Concern is a behavior that we want to have in a module of an application. Issues in which we are interested defines our concern.

### 1) What is Spring?

It is a lightweight, loosely coupled and integrated framework for developing enterprise applications in java.

### 2) What are the advantages of spring framework?

1. Predefined Templates
2. Loose Coupling
3. Easy to test
4. Lightweight
5. Fast Development
6. Powerful Abstraction
7. Declarative support

[More details...](https://www.javatpoint.com/spring-tutorial)

### 3) What are the modules of spring framework?

1. Test
2. Spring Core Container
3. AOP, Aspects and Instrumentation
4. Data Access/Integration
5. Web

[More details...](https://www.javatpoint.com/spring-modules)

### 4) What is IOC and DI?

IOC (Inversion of Control) and DI (Dependency Injection) is a design pattern to provide loose coupling. It removes the dependency from the program.

Let's write a code without following IOC and DI.

1. **public** **class** Employee{
2. Address address;
3. Employee(){
4. address=**new** Address();//creating instance
5. }
6. }

Now, there is dependency between Employee and Address because Employee is forced to use the same address instance.

Let's write the IOC or DI code.

1. **public** **class** Employee{
2. Address address;
3. Employee(Address address){
4. **this**.address=address;//not creating instance
5. }
6. }

Now, there is no dependency between Employee and Address because Employee is not forced to use the same address instance. It can use any address instance.

### 5) What is the role of IOC container in spring?

IOC container is responsible to:

* create the instance
* configure the instance, and
* assemble the dependencies

[More details...](https://www.javatpoint.com/ioc-container)

### 6) What are the types of IOC container in spring?

There are two types of IOC containers in spring framework.

1. BeanFactory
2. ApplicationContext

[More details...](https://www.javatpoint.com/ioc-container)

### 7) What is the difference between BeanFactory and ApplicationContext?

BeanFactory is the **basic container** whereas ApplicationContext is the **advanced container**. ApplicationContext extends the BeanFactory interface. ApplicationContext provides more facilities than BeanFactory such as integration with spring AOP, message resource handling for i18n etc.

### 8) What is the difference between constructor injection and setter injection?

|  |  |  |
| --- | --- | --- |
| **No.** | **Constructor Injection** | **Setter Injection** |
| 1) | No Partial Injection | Partial Injection |
| 2) | Desn't override the setter property | Overrides the constructor property if both are defined. |
| 3) | Creates new instance if any modification occurs | Doesn't create new instance if you change the property value |
| 4) | Better for too many properties | Better for few properties. |

[More details...](https://www.javatpoint.com/difference-between-constructor-and-setter-injection)

### 9) What is autowiring in spring? What are the autowiring modes?

Autowiring enables the programmer to inject the bean automatically. We don't need to write explicit injection logic.

Let's see the code to inject bean using dependency injection.

1. <bean id="emp" **class**="com.javatpoint.Employee" autowire="byName" />

The autowiring modes are given below:

|  |  |  |
| --- | --- | --- |
| **No.** | **Mode** | **Description** |
| 1) | no | this is the default mode, it means autowiring is not enabled. |
| 2) | byName | injects the bean based on the property name. It uses setter method. |
| 3) | byType | injects the bean based on the property type. It uses setter method. |
| 4) | constructor | It injects the bean using constructor |

The "autodetect" mode is deprecated since spring 3.

### 10) What are the different bean scopes in spring?

There are 5 bean scopes in spring framework.

|  |  |  |
| --- | --- | --- |
| **No.** | **Scope** | **Description** |
| 1) | singleton | The bean instance will be only once and same instance will be returned by the IOC container. It is the default scope. |
| 2) | prototype | The bean instance will be created each time when requested. |
| 3) | request | The bean instance will be created per HTTP request. |
| 4) | session | The bean instance will be created per HTTP session. |
| 5) | globalsession | The bean instance will be created per HTTP global session. It can be used in portlet context only. |

### 11) In which scenario, you will use singleton and prototype scope?

Singleton scope should be used with EJB **stateless session bean** and prototype scope with EJB **stateful session bean**.

### 12) What are the transaction management supports provided by spring?

Spring framework provides two type of transaction management supports:

1. **Programmatic Transaction Management**: should be used for few transaction operations.
2. **Declarative Transaction Management**: should be used for many transaction operations.

## **» Spring JDBC Interview Questions**

### 13) What are the advantages of JdbcTemplate in spring?

**Less code**: By using the JdbcTemplate class, you don't need to create connection,statement,start transaction,commit transaction and close connection to execute different queries. You can execute the query directly.

[More details...](https://www.javatpoint.com/spring-JdbcTemplate-tutorial)

### 14) What are classes for spring JDBC API?

1. JdbcTemplate
2. SimpleJdbcTemplate
3. NamedParameterJdbcTemplate
4. SimpleJdbcInsert
5. SimpleJdbcCall

[More details...](https://www.javatpoint.com/spring-JdbcTemplate-tutorial)

### 15) How can you fetch records by spring JdbcTemplate?

You can fetch records from the database by the **query method of JdbcTemplate**. There are two interfaces to do this:

1. [ResultSetExtractor](https://www.javatpoint.com/ResultSetExtractor-example)
2. [RowMapper](https://www.javatpoint.com/RowMapper-example)

### 16) What is the advantage of NamedParameterJdbcTemplate?

NamedParameterJdbcTemplate class is used to pass value to the named parameter. A named parameter is better than ? (question mark of PreparedStatement).

It is **better to remember**.

[More details...](https://www.javatpoint.com/spring-NamedParameterJdbcTemplate-example)

### 17) What is the advantage of SimpleJdbcTemplate?

The **SimpleJdbcTemplate** supports the feature of var-args and autoboxing.

[More details...](https://www.javatpoint.com/spring-SimpleJdbcTemplate-example)

## **» Spring AOP Interview Questions**

### 18) What is AOP?

AOP is an acronym for Aspect Oriented Programming. It is a methodology that divides the program logic into pieces or parts or concerns.

It increases the modularity and the key unit is Aspect.

[More details...](https://www.javatpoint.com/spring-aop-tutorial)

### 19) What are the advantages of spring AOP?

AOP enables you to dynamically add or remove concern before or after the business logic. It is **pluggable** and **easy to maintain**.

[More details...](https://www.javatpoint.com/spring-aop-tutorial)

### 20) What are the AOP terminology?

AOP terminologies or concepts are as follows:

* JoinPoint
* Advice
* Pointcut
* Aspect
* Introduction
* Target Object
* Interceptor
* AOP Proxy
* Weaving

[More details...](https://www.javatpoint.com/spring-aop-tutorial)

### 21) What is JoinPoint?

JoinPoint is any point in your program such as field access, method execution, exception handling etc.

### 22) Does spring framework support all JoinPoints?

No, spring framework supports method execution joinpoint only.

### 23) What is Advice?

Advice represents action taken by aspect.

### 24) What are the types of advice in AOP?

There are 5 types of advices in spring AOP.

1. Before Advice
2. After Advice
3. After Returning Advice
4. Throws Advice
5. Around Advice

### 25) What is Pointcut?

Pointcut is expression language of Spring AOP.

### 26) What is Aspect?

Aspect is a class in spring AOP that contains advices and joinpoints.

### 27) What is Introduction?

Introduction represents introduction of new fields and methods for a type.

### 28) What is target object?

Target Object is a proxy object that is advised by one or more aspects.

### 29) What is interceptor?

Interceptor is a class like aspect that contains one advice only.

### 30) What is weaving?

Weaving is a process of linking aspect with other application.

### 31) Does spring perform weaving at compile time?

No, spring framework performs weaving at runtime.

### 32) What are the AOP implementation?

There are 3 AOP implementation.

1. Spring AOP
2. Apache AspectJ
3. JBoss AOP

## **» Spring MVC Interview Questions**

### 33) What is the front controller class of Spring MVC?

The **DispatcherServlet** class works as the front controller in Spring MVC.

[More details...](https://www.javatpoint.com/spring-3-mvc-tutorial)

### 34) What does @Controller annotation?

The **@Controller** annotation marks the class as controller class. It is applied on the class.

### 35) What does @RequestMapping annotation?

The **@RequestMapping** annotation maps the request with the method. It is applied on the method.

### 36) What does the ViewResolver class?

The **View Resolver** class resolves the view component to be invoked for the request. It defines prefix and suffix properties to resolve the view component.

### 37) Which ViewResolver class is widely used?

The **org.springframework.web.servlet.view.InternalResourceViewResolver** class is widely used.

### 38) Does spring MVC provide validation support?

Yes.

### 1. What are the main features introduced in spring 4?

Ans: Spring 4 has introduced many new features. Some of them are as follows.   
  
1. @RestController annotation has been introduced for easiness to develop spring rest web service.   
2. AsyncRestTemplate has been added to develop rest web service.   
3. Java 8 and hibernate 4.3 have been supported.   
4. Different time zone in spring MVC has been supported.   
5. Now spring supports websocket protocol.   
6. Spring messaging supports STOMP protocol.   
7. Spring security JUnit test module has been added with @WithMockUser and @WithUserDetails annotations.

### 2. What is the use of @RestController annotation in spring 4?

Ans: Spring 4 has introduced @RestController annotation that has replaced @Controller and @ResponseBody. In spring 4 rest web service development, our service methods need not to use @ResponseBody.   
  
@RestController = @Controller + @ResponseBody

@RestController

@RequestMapping("/data")

public class PersonController {}

### 3. What is the role of AsyncRestTemplate and ListenableFuture in spring 4?

Ans: AsyncRestTemplate can return the URL output asynchronously. ListenableFuture is the return type which itself will return ResponseEntity.

ListenableFuture<ResponseEntity<String>> future =

asycTemp.exchange(url, method, requestEntity, responseType);

ResponseEntity<String> entity = future.get();

### 4. What is the role of AsyncClientHttpRequestFactory and AsyncClientHttpRequest in spring 4?

Ans: AsyncClientHttpRequestFactory returns the instance of AsyncClientHttpRequest that represents client side asynchronous HTTP request. We use it as follows.

ListenableFuture<ClientHttpResponse> future = asyncClientHttpRequest.executeAsync();

### 5. How to use WebSocket in spring 4?

Ans: 1. Java configuration class implements AbstractWebSocketMessageBrokerConfigurer and we need to override its methods that are configureMessageBroker() and registerStompEndpoints().   
  
2. Java configuration class should be annotated with @EnableWebSocketMessageBroker with @Configuration.   
3. Spring controller class uses @SendTo annotation with @MessageMapping at method level to declare result URL.   
4. To work with WebSocket, other protocol and JS library such as SockJS and STOMP Protocol are required.

### 6. What is the role of @CacheConfig in spring 4?

Ans: @CacheConfig is used at class level. It is used to set common cache related settings. All the methods annotated with @Cacheable override the settings of @CacheConfig.

@Service

@CacheConfig(cacheNames="mycacheone")

public class Student {}

### 7. How to handle @Async exception in spring 4?

Ans: Spring 4 provides AsyncUncaughtExceptionHandler that caches exception thrown by the method annotated with @Async. We create a class implementing AsyncUncaughtExceptionHandler.

public class MyAsyncUncaughtExceptionHandler implements AsyncUncaughtExceptionHandler {

@Override

public void handleUncaughtException(Throwable ex, Method method, Object... params) {}

}

### 8. What is the role of @WithMockUser and @WithUserDetails annotation in spring 4 security JUnit test

Ans: @WithMockUser annotation allows mock user at server side in spring security JUnit testing. There are usernameand roles attributes in @WithMockUser annotation. We use it as follows.

@Test

@WithMockUser(username = "ram", roles={"ADMIN"})

public class SpringSecurityTest {}

@WithUserDetails annotation provides custom UserDetailsService in spring security JUnit testing and we can use it as follows.

@Test

@WithUserDetails("ram")

public void testFour() {

userService.methodFour();

}