SPRING

1. What is a framework?

A framework is a tool that provides ready components or solutions that are customized in order to speed up the development.

1. What is Spring?

Spring is a powerful open-source, application framework created to reduce the complexity of enterprise application development.

It is light-weighted and loosely coupled.

It has layered architecture, which allows you to select the components to use, while also providing a cohesive framework for J2EE application development.

Spring framework is also called the framework of frameworks as it provides support to various other frameworks such as Struts, Hibernate, Tapestry, EJB, JSF, etc.

1. Name the different modules of the Spring framework.

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. List some of the important annotations in annotation-based Spring configuration.

The important annotations are:

@Controller – for controller classes in the 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

@ResponseBody – for sending Object as a response, usually for sending XML or JSON data as a response.

@PathVariable – for mapping dynamic values from the URI to handler method arguments.

@Autowired – for auto wiring dependencies in spring beans.

@Qualifier – with @Autowired annotation to avoid confusion when multiple instances of bean type are present.

@Service – for service classes.

@Scope – for configuring the scope of the spring bean.

@Configuration, @ComponentScan, and @Bean – for java based configurations.

AspectJ annotations for configuring aspects and advice, @Aspect, @Before, @After, @Around, @Pointcut, etc.

1. Explain Bean in Spring.

Beans are objects that form the backbone of a Spring application. They are managed by the Spring IoC container. In other words, a bean is an object that is instantiated, assembled, and managed by a Spring IoC container.

1. Different Scopes of Spring beans.

There are five Scopes defined in Spring beans:

* Singleton: 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 the same as the 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.

1. DispatcherServlet

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

1. ContextLoaderListener

ContextLoaderListener, on the other hand, is the listener to start up and shut down the WebApplicationContext in Spring root. Some of its important functions include tying up the lifecycle of Application Context to the lifecycle of the ServletContext and automating the creation of ApplicationContext.

1. Constructor injection

* No Partial Injection
* Doesn’t override the setter property
* Creates a new instance if any modification occurs
* Better for too many properties

1. Setter injection

* Partial Injection
* Overrides the constructor property if both are defined.
* Doesn’t create a new instance if you change the property value
* Better for a few properties.

1. What is auto wiring in Spring? What are the auto wiring modes?

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

Autowiring modes:

* No - this is the default mode, it means auto wiring is not enabled.
* byName - Injects the bean based on the property name. It uses the setter method.
* byType - Injects the bean based on the property type. It uses the setter method.
* Constructor - It injects the bean using the constructor

1. What are the limitations with auto wiring?

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

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

1. How to handle exceptions in Spring MVC Framework?

Spring MVC Framework provides the following ways to help us achieve 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 time we serve static pages. Spring Framework provides a HandlerExceptionResolver interface that we can implement to create a global exception handler. The reason behind this additional way to define the global exception handler is that the 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 integrate Spring and Hibernate Frameworks?

We can use the Spring ORM module to integrate Spring and Hibernate frameworks if you are using Hibernate 3+ where SessionFactory provides the current session, then you should avoid using HibernateTemplate or HibernateDaoSupport classes, and better to use the 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 hibernating boiler-plate code for transaction management.

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

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

* 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.
* Declarative transaction management: In this, transaction management is separated from the business code. Only annotations or XML-based configurations are used to manage the transactions.

1. What is a Spring configuration file?

The Spring configuration file is an XML file. This file mainly contains the class 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 big projects.

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

A Spring application generally consists of the following components:

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

1. What is an 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.

1. 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 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.

1. List some of the benefits of IoC.

Some of the benefits of IoC are:

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

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

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

* Constructor Injection
* Setter Injection
* Interface Injection

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

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

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

1. 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.

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

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

* XML-Based configuration: In Spring Framework, the dependencies and the services needed by beans are specified in configuration files which are in XML format. These configuration files usually contain a lot of bean definitions and application-specific configuration options. They generally start with a bean bag.
* 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.
* Java-based configuration: The key features in Spring Framework’s new Java-configuration support are @Configuration annotated classes and @Bean annotated methods.

1. Define Bean Wiring.

When beans are combined 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.

1. 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.

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

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

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

1. 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 consistently. This allows one to switch between the persistence technologies easily. It also allows you to code without worrying about catching exceptions that are specific to each of these technologies.

1. Describe AOP.

Aspect-oriented programming or AOP is a programming technique that 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.

1. 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 the case of client objects, the target object and the proxy object are the same.

1. What do you mean by the 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.

1. Describe DispatcherServlet.

The DispatcherServlet is the core of the 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 the view resolver in the configuration file and calls the specified view component.

1. In the Spring MVC framework, what is a 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, the controller is implemented in a very abstract way. It also enables you to create a wide variety of controllers.