Q1.What is Spring Framework?

Ans

The Spring Framework is an open-source application framework and inversion of control container for the Java platform. It provides a comprehensive programming and configuration model for modern Java-based enterprise applications.

Spring Framework is used to create enterprise applications that are:

Loosely coupled: The Spring Framework uses dependency injection to make applications loosely coupled , which means that the components of an application do not depend on each other directly. This makes a pplications easier to understand, test, and maintain.
Highly scalable: The Spring Framework is designed to be highly scalable, so applications built on Spring c an handle large volumes of traffic.
Secure: The Spring Framework includes a number of features that help to secure applications, such as au thentication, authorization, and encryption.

Some of the key features of the Spring Framework include:

Dependency injection: Dependency injection is a technique for injecting the dependencies of a class into t hat class. This makes applications loosely coupled and easier to test.
Inversion of control (IoC): IoC is a design pattern that inverts the traditional control flow of an application. I n a traditional application, the code would explicitly create and manage the objects that it needed. With Io C, the objects are created and managed by the framework.
Aspect-oriented programming (AOP): AOP is a technique for modularizing cross-cutting concerns in an a pplication. Cross-cutting concerns are things like logging, transactions, and security. AOP makes it easier to implement these concerns in an application.
Data access: The Spring Framework provides a number of features for data access, such as JDBC, ORM , and JMS.
Web development: The Spring Framework includes a web framework that makes it easy to develop web a pplications.

Q2.What are the features of Spring Framework?

Ans

The Spring Framework has a wide range of features, including:

Dependency injection: Dependency injection is a technique for injecting the dependencies of a class into t hat class. This makes applications loosely coupled and easier to test.
Inversion of control (IoC): IoC is a design pattern that inverts the traditional control flow of an application. I n a traditional application, the code would explicitly create and manage the objects that it needed. With Io C, the objects are created and managed by the framework.
Aspect-oriented programming (AOP): AOP is a technique for modularizing cross-cutting concerns in an a pplication. Cross-cutting concerns are things like logging, transactions, and security. AOP makes it easier to implement these concerns in an application.
Data access: The Spring Framework provides a number of features for data access, such as JDBC, ORM , and JMS.
Web development: The Spring Framework includes a web framework that makes it easy to develop web a pplications.
Testing: The Spring Framework provides a number of features for testing, such as mock objects and unit t esting.
Security: The Spring Framework includes a number of features for security, such as authentication, autho

rization, and encryption.
Internationalization: The Spring Framework supports internationalization, so applications can be localized for different regions and languages.
Caching: The Spring Framework provides a caching framework that can be used to improve the performa nce of applications.
Monitoring: The Spring Framework provides a monitoring framework that can be used to track the perform ance of applications.

Q3.What is a Spring configuration file?

Ans

A Spring configuration file is a file that defines the beans in a Spring application. Beans are the componen ts of a Spring application, such as DAOs, services, and controllers. The Spring configuration file tells the Spring container how to create and manage these beans.

Spring configuration files can be written in XML or Java. XML is the more traditional way to write Spring c onfiguration files, but Java is becoming more popular.

A Spring configuration file typically contains the following elements:

Bean definitions: These elements define the beans in the application.
Property definitions: These elements define the properties of beans.
Imports: These elements import other Spring configuration files.
Events: These elements define the events that are fired in the application.
Profiles: These elements define the profiles that can be used to run the application.
Spring configuration files are an important part of Spring applications. They tell the Spring container how t o create and manage the beans in the application.

Q4.What do you mean by IoC Container?

Ans

Inversion of Control (IoC) is a design pattern that inverts the traditional control flow of an application. In a t raditional application, the code would explicitly create and manage the objects that it needed. With IoC, th e objects are created and managed by the framework.

An IoC container is a framework that implements IoC. It is responsible for creating, configuring, and mana ging the objects in an application. The IoC container provides a number of benefits, including:

Loose coupling: The IoC container makes the components of an application loosely coupled, which mean s that they do not depend on each other directly. This makes applications easier to understand, test, and maintain.
Reusability: The IoC container makes it easy to reuse components in an application. This is because the I oC container takes care of creating and managing the objects, so the code does not need to do it.
Scalability: The IoC container can be used to create scalable applications. This is because the IoC contai ner can create and manage a large number of objects.

Q5.What do you understand by Dependency Injection?

Ans

Dependency injection (DI) is a software design pattern that allows us to pass the dependencies of a class to that class. This makes applications loosely coupled and easier to test.

In a traditional application, the code would explicitly create and manage the objects that it needed. With DI, the objects are created and managed by the framework. This makes it easier to change the dependencies of a class without having to change the code.

There are two main types of dependency injection: constructor injection and setter injection.

Constructor injection: In constructor injection, the dependencies of a class are passed to the constructor of the class. This is the most common type of dependency injection.
Setter injection: In setter injection, the dependencies of a class are passed to the setter methods of the class. This is less common than constructor injection, but it can be useful in some cases.

Q6.Explain the difference between constructor and setter injection?

Ans

Constructor injection is a type of dependency injection where the dependencies of a class are passed to the constructor of the class. This is the most common type of dependency injection.

Setter injection is a type of dependency injection where the dependencies of a class are passed to the setter methods of the class. This is less common than constructor injection, but it can be useful in some cases.

There are many key differences between constructor injection and setter injection.

Partial dependency: can be injected using setter injection but it is not possible by constructor. Suppose there are 3 properties in a class, having 3 arg constructor and setters methods. In such case, if you want to pass information for only one property, it is possible by setter method only.
Overriding: Setter injection overrides the constructor injection. If we use both constructor and setter injection, IOC container will use the setter injection.
Changes: We can easily change the value by setter injection. It doesn’t create a new bean instance always like constructor. So setter injection is flexible than constructor injection.

Q7.What are Spring Beans?

Ans

A Spring bean is an object that is managed by the Spring IoC container. This means that the container creates the bean, configures it, and manages its lifecycle. Spring beans are the foundation of Spring applications, and they are used to represent all of the components in an application, such as DAOs, services, and controllers.

Q8.What are the bean scopes available in Spring?

Ans

Bean scopes available in Spring:

Singleton: This is the default scope for Spring beans. A singleton bean is created once per Spring IoC container and is shared by all of the beans in the container.

Prototype: A prototype bean is created anew each time it is requested from the Spring IoC container.
Request: A request bean is created for each HTTP request. This scope is only available in web applicatio ns.
Session: A session bean is created for each HTTP session. This scope is only available in web applicatio ns.
Global-session: A global-session bean is created for each global HTTP session. This scope is only availa ble in web applications that use Portlet containers.
The scope of a bean determines how it is created and managed by the Spring IoC container. The default scope for Spring beans is singleton. This means that a singleton bean is created once per Spring IoC cont ainer and is shared by all of the beans in the container.

If you need a bean that is created anew each time it is requested from the Spring IoC container, then you should use the prototype scope. The prototype scope is useful for beans that are expensive to create or t hat need to be customized for each request.

If you are developing a web application, then you may want to use the request or session scope. The req uest scope is useful for beans that need to be created for each HTTP request. The session scope is usefu l for beans that need to be created for each HTTP session.

The global-session scope is only available in web applications that use Portlet containers. It is similar to th e session scope, but it is created for each global HTTP session.

Q9.What is Autowiring and name the different modes of it?

Ans

Autowiring is a feature of the Spring Framework that automatically injects the dependencies of a bean into that bean. This makes applications loosely coupled and easier to test.

There are four different modes of autowiring in Spring:

No autowiring: This is the default mode. In this mode, no autowiring is done. The dependencies of a bean must be explicitly injected using the @Autowired annotation or the set method.
Autowiring by name: In this mode, Spring tries to inject the dependencies of a bean by matching the nam e of the dependency with the name of the bean. For example, if a bean has a dependency on a Logger be an, Spring will try to inject a bean with the name logger.
Autowiring by type: In this mode, Spring tries to inject the dependencies of a bean by matching the type of the dependency with the type of the bean. For example, if a bean has a dependency on a Logger bean, Spring will try to inject a bean of type Logger.
Autowiring constructor: In this mode, Spring tries to inject the dependencies of a bean by matching the typ es of the arguments in the constructor with the types of the beans. For example, if a bean has a construct or that takes a Logger bean as an argument, Spring will try to inject a bean of type Logger.

Q10.Explain Bean life cycle in Spring Bean Factory Container.

Ans

The bean life cycle in Spring Bean Factory Container is a series of steps that a bean goes through from th e time it is created to the time it is destroyed. The life cycle of a bean is managed by the Spring IoC contai ner.

The following are the steps in the bean life cycle:

Instantiation: The first step in the bean life cycle is instantiation. In this step, the Spring IoC container creates a new instance of the bean.
Configuration: The second step in the bean life cycle is configuration. In this step, the Spring IoC container configures the bean. This includes setting the bean's properties and injecting the bean's dependencies.
Initialization: The third step in the bean life cycle is initialization. In this step, the bean is initialized. This may involve calling the bean's init() method or executing some other code.
Ready to use: After the bean has been initialized, it is ready to be used.
Service: The fourth step in the bean life cycle is service. In this step, the bean is used. This may involve calling the bean's methods or accessing the bean's properties.
Destruction: The fifth and final step in the bean life cycle is destruction. In this step, the bean is destroyed. This may involve calling the bean's destroy() method or executing some other code.
The bean life cycle can be customized by implementing the InitializingBean and DisposableBean interfaces. The InitializingBean interface provides a method called afterPropertiesSet() that can be used to initialize the bean. The DisposableBean interface provides a method called destroy() that can be used to destroy the bean.

Here are some of the benefits of the bean life cycle:

It allows you to control the initialization and destruction of beans.
It allows you to implement custom initialization and destruction logic.
It allows you to decouple the bean from the container.