Assignment Questions 9

Q1. What is Spring Framework?

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

Q2. What are the 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.

Q3. What is a Spring configuration file?

A Spring application, generally consists of following components:

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

Q4. What do you mean by 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.

Q5. What do you understand 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.

Q6. Explain the difference between constructor and setter injection?

Constructor Injection vs Setter Injection

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

Q7. What are 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.

Q8. What are the bean scopes available in Spring?

The Spring Framework supports five scopes. They are:

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

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

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

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

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

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

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