**What is dependency injection and what are the advantages?**

Dependency injection is the process whereby a framework or such, for example the Spring Framework, establishes the relationships between different parts of an application. This is opposed to the application code itself being responsible of establishing these relationships.

When using the Spring Framework for Java development, some of the advantages of dependency injection are:

• Reduced coupling between the parts of an application.

• Increased cohesion of the parts of an application.

• Increased testability.

• Better design of applications when using dependency injection.

• Increased reusability.

• Increased maintainability.

• Standardizes parts of application development.

• Reduces boilerplate code.

No code needs to establish relationships in domain classes. Such code or configuration separated into XML or Java configuration classes.

**What is a pattern? What is an anti-pattern?**

Design pattern is a typical solution to commonly occurring problem in software design, which considered a good programming practice.

An anti-pattern is a commonly used template that attempts to solve a type of problem but turns out to be counterproductive and inefficient. An anti-pattern may also be a habit among software developers that reduces productivity or some desirable code quality etc.

**What is an interface and what are the advantages of making use of them in Java?**

1. Reduces dependencies on implementation details, therefore making code more reusable and easier to refactor

2. Allows providing different implementations at runtime

3. Makes code easier to test

**What is meant by application-context?**

By application context is usually meant an advanced IoC container. It's called an 'advanced' because of it superset capabilities in comparison to simple bean factory. Term 'application-context' is directly related to interface which is central in Spring Framework. As related JavaDoc says, ApplicationContext offers the following functionality:

1. Bean factory

2. Resource loading

3. Events publishing to registered listeners

4. Internationalization

5. Composition of inheritance chains with other contexts

**What is the concept of a “container” and what is its lifecycle?**

The container will create the objects, wire them together, configure them, and manage their complete life cycle from creation till destruction. The Spring container uses DI to manage the components in application.

1. Spring container is created as the application is started.

2. The container reads configuration data.

3. Bean definitions are created from the configuration data.

4. Bean factory post-processors processes the bean definitions.

5. Spring beans are instantiated by the container using the bean definitions.

6. Spring beans are configured and assembled. Property values and dependencies are injected into the beans by the container.

7. Bean post-processors processes the beans in the container and any initialization callbacks are invoked on the beans.

8. The application runs.

9. Application shut down is initialized.

10. The Spring container is closed.

11. Destruction callbacks are invoked on the singleton Spring beans in the container.

**How are you going to create a new instance of an ApplicationContext?**

ClassPathXmlApplicationContext

FileSystemXmlApplicationContext

AnnotationConfigApplicationContext

--

ContextLoadListener, DispatcherServlet and web.xml (Servlets 2.0)

XmlWebApplicationContext (Servlets 3.0)

AnnotationConfigWebApplicationContext (Servlets 3.0)

**What is the preferred way to close an application context? Does Spring Boot do this for you?**

Standalone ApplicationIn a standalone non-web Spring application, there are two ways by which the Spring application context can be closed.  
• Registering a shutdown-hook by calling the method registerShutdownHook, also  
implemented in the AbstractApplicationContext class.  
This will cause the Spring application context to be closed when the Java virtual machine is shut down normally. This is the recommended way to close the application context in a nonweb application.  
• Calling the close method from the AbstractApplicationContext class.  
This will cause the Spring application context to be closed immediately.  
Web ApplicationIn a web application, closing of the Spring application context is taken care of by the  
ContextLoaderListener, which implements the ServletContextListener interface. The  
ContextLoaderListener will receive a ServletContextEvent when the web container stops the web application.

Spring Boot registers a shut-down hook and also have ContextLoadListener.