# Spring Questions

## API Security

* Security of data in transit (This is typically achieved by SSL)
* Authentication and Authorization
* Token based security, JWT token (JSON Web Token)
* OAuth 2.0 — OAuth is very widely used and popular for Rest API security

HTTP Request (Verb / Action to be performed on server)

* 1. GET - Request Resource
  2. POST - Create Resource
  3. PUT - Update Resource
  4. DELETE - Delete Resource
  5. PATCH - Update partial resource - This like this, We do not want to update whole Customer, instead we want to update address of Customer. This can be treated as update partial response (i.e - PATCH Request)

HTTP Request Headers - actually request metadata, examples are,

* Content Type
* Content Length
* Authorization
* Accept
* Cookies

HTTP Response format -

* Response status code.
* Response headers
* Response Content - HTML,CSS,JavaScript,XML,JSON etc. Note that sim

What is important part of designing restful api's ?

* Design URI
* Identify headers, if any
* Request body

1. Spring IOC Container ???  
Ans - Below are the imp points of Spring IOC container,

1. Spring IOC container is the core of spring framework.
2. Spring IOC container will create objects, wire them together as per instructions provide in XML or through Java annotation.
3. Spring IOC container manages life cycle of objects, i.e from creation to destruction.
4. Objects that are created are called spring beans.
5. Spring IOC container uses dependency injection to manage the components that make up an application.
6. Spring IOC container gets the instruction on what objects to instantiate, configure and assemble by reading configuration metadata, through xml file or Java annotation.

2. What is difference between "beanfactory" and "ApplicationContext" ?  
 Ans - Below are the differences between them

1. beanfactory is a basic container while ApplicationContext is advanced.
2. BeanFactory doesn't provide support for internationalization i.e. i18n but ApplicationContext provides support for it.
3. Beanfactory Container support only two scopes (singleton & prototype) of the beans. But ApplicationContext Container supports all the beans scope.
4. Beanfactory support lazy loading while ApplicationContext support eager loading.

3. What are different spring bean scopes ?  
 Ans - Below are different scopes of a spring bean,

1. Singleton - Scopes a single bean definition to a single object instance per Spring IoC container. Default bean scope is singleton.

<bean id="accountService" class="com.foo.DefaultAccountService"/>

<!-- the following is equivalent, though redundant (singleton scope is the default); using spring-beans-2.0.dtd -->

<bean id="accountService" class="com.foo.DefaultAccountService" scope="singleton"/>

<!-- the following is equivalent and preserved for backward compatibility in spring-beans.dtd -->

<bean id="accountService" class="com.foo.DefaultAccountService" singleton="true"/>

1. Prototype - Scopes a single bean definition to any number of object instances. As a rule of thumb, you should use the prototype scope for all beans that are stateful, while the singleton scope should be used for stateless beans.

<!-- using spring-beans-2.0.dtd -->

<bean id="accountService" class="com.foo.DefaultAccountService" scope="prototype"/>

<!-- the following is equivalent and preserved for backward compatibility in spring-beans.dtd -->

<bean id="accountService" class="com.foo.DefaultAccountService" singleton="false"/>

There is one quite important thing to be aware of when deploying a bean in the prototype scope, in that the lifecycle of the bean changes slightly. Spring does not manage the complete lifecycle of a prototype bean: the container instantiates, configures, decorates and otherwise assembles a prototype object, hands it to the client and then has no further knowledge of that prototype instance. This means that while initialization lifecycle callback methods will be called on all objects regardless of scope, in the case of prototypes, any configured destruction lifecycle callbacks will not be called. It is the responsibility of the client code to clean up prototype scoped objects and release any expensive resources that the prototype bean(s) are holding onto.

3. The other scopes, namely request, session, and global session are for use only in web-based applications.

* Request -

<bean id="loginAction" class="com.foo.LoginAction" scope="request"/>

With the above bean definition in place, the Spring container will create a brand new instance of the LoginAction bean using the 'loginAction' bean definition for each and every HTTP request. That is, the 'loginAction' bean will be effectively scoped at the HTTP request level.

* Session

<bean id="userPreferences" class="com.foo.UserPreferences" scope="session"/>

With the above bean definition in place, the Spring container will create a brand new instance of the UserPreferences bean using the 'userPreferences' bean definition for the lifetime of a single HTTP Session. In other words, the 'userPreferences' bean will be effectively scoped at the HTTP Session level.

* Global session scope -

<bean id="userPreferences" class="com.foo.UserPreferences" scope="globalSession"/>

Basically we are defining a bean which is valid for all sessions, so calling it global session.

4. What are profiles in Spring boot ?

Ans - Every enterprise application has different environment. Like, DEV/TEST/PREPROD/PROD etc. For example every environment has a differnt database, so db connectivity details will be present in environment specific property file. Name of such files are

* application-dev.properties
* application-test.properties
* application-prod.properties

But remember that application.properties remains the master property file. If we over-ride any key in the profile specific property file then that will gain precedence. Remember that 1st line of application.properties will be spring.profiles.active=dev

5. What are advantages of Spring framework ?  
Ans - Below are the advantages ,

* Lightweight framework for Java EE development
* Supports good programming practice such as programming using interfaces instead of classes.
* Supports XML based, annotation based, pure java based configuration.
* It gives good support for IoC and Dependency Injection results in loose coupling.
* Spring Framework supports JDBC framework
* Spring provided support to ORM. Provides abstraction for database development.
* Spring Web MVC framework is a powerful alternative for web based applications.
* Spring boot further spimplifies and speeds up development of Spring based projects.
* The Spring Test module provides support for an easy-to-test code.