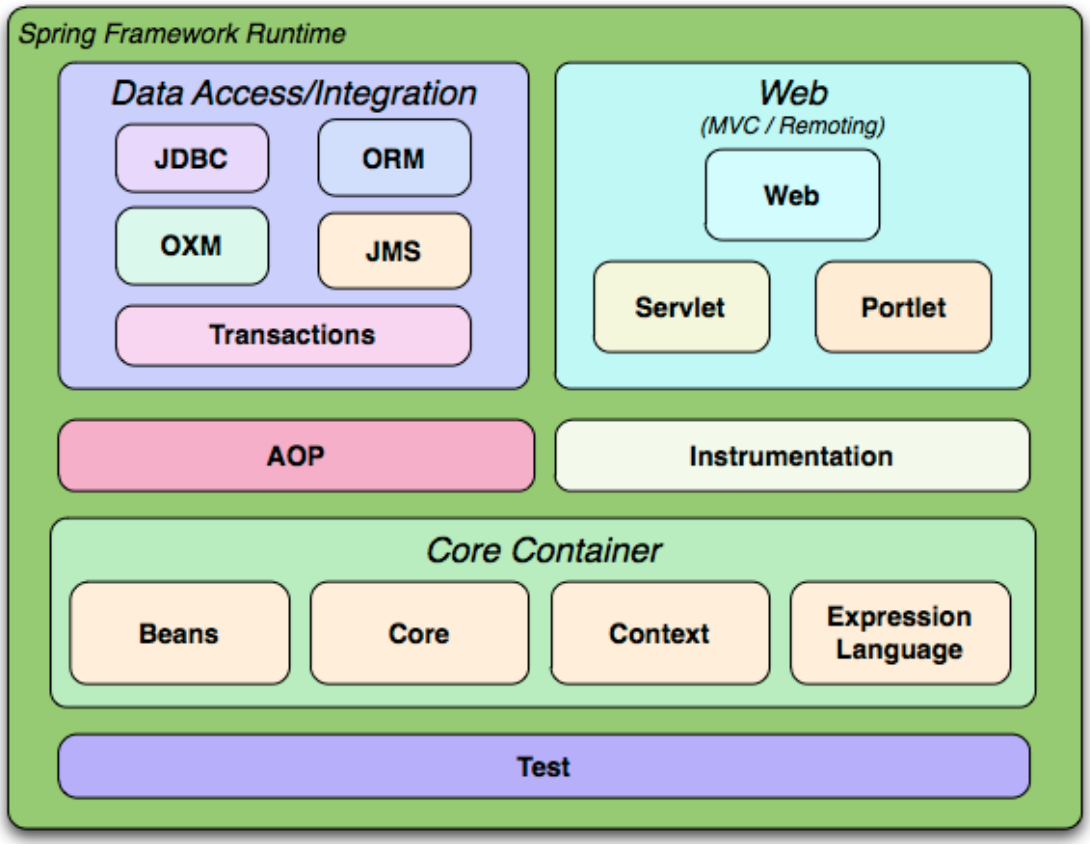
# Spring Framework Interview Q&A

*-By SivaReddy*

***1) What is Spring Framework, Features and Advantages?***

- *Spring Framework is a powerful lightweight application development framework used for Enterprise Java (JEE)*
- *Modular Framework*
- *Light Weight*
- *End to End Development*
- *Inversion of Control*
- *Aspect Oriented programming*
- *Containers*
- *MVC Framework*
- *Transaction Management*
- *Abstract Layer for JDBC*
- *Easy Integration with Framework Struts, JSF, Hibernate etc..*
- *Easy to Test*

*2) What are Spring framework modules?*




*Source: docs.spring.io*

***3) What is dependency lookup?***

*The Dependency Lookup is an approach where we get the resource after demand*

```
class Employee{
Addressaddress;
Employee(Address address){
this.address = new Address();
}
}
```

*> Tight Coupling & Not easy to test*

***4) What is dependency Injection and Inversion of Control?***

- ***Inversion of Control** (IoC) means that objects do not create other objects on which they rely to do their work. Instead, they get the objects that they need from an outside source (for example, an xml configuration file).*
- ***Dependency Injection** (DI) means that this is done without the object intervention, usually by a framework component that passes constructor parameters and set properties.*

***Loosely couple and Easy to test***

```
class Employee{
Addressaddress;
Employee(Address address){
this.address =address
}

publicvoid setAddress(Addressaddress) {
	this.address = address;
}

}
```

***5) How many ways you can create spring application meta data?***

- *XML-based metadata*
- *Annotation-based configuration(Spring 2.5)*
- *Java-based configuration(Spring 3.0)*

*6) What is the difference between BeanFactory vs. ApplicationContext?*

***Bean Factory :** Bean instantiation/wiring, Support Lazy loading*

***Application Context***

- *Bean instantiation/wiring*
- *Automatic BeanPostProcessor registration*
- *Automatic BeanFactoryPostProcessor registration*
- *Convenient MessageSource access (for i18n)*
- *ApplicationEvent publication*

***So if you need any of the points presented on the Application Context side, you should use ApplicationContext.***

***7) What are different types of DI and which one is better?***

- *Setter Based DI*
- *Example:*

*<bean id="address" class="com.sivajavatechie.spring.Address" />*

*<bean id="emp" class="com.sivajavatechie.spring.Employee">*

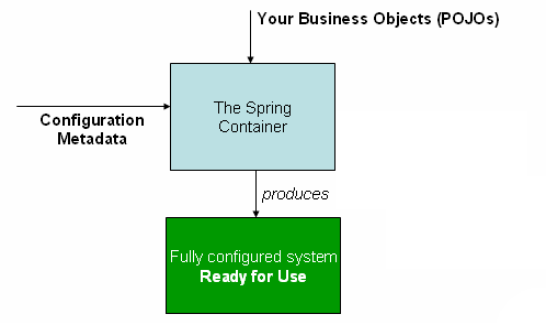
*<propertyname="addresss" ref="address"></property>*

*</bean>*

- *Constructor Based DI*

*Example:*
*<bean id="address" class="com.sivajavatechie.spring.Address" />*
*<bean id="emp" class="com.sivajavatechie.spring.Employee">*
*<constructor-arg value="addresss" ref="address"> </constructor-arg>*
*</bean>*

***8) Explain in detail about Spring IOC container?***




*Source: docs.spring.io*

- `org.springframework.context.ApplicationContext`
- org.springframework.beans.factory.BeanFactory
- Responsible for instantiating, configuring, and assembling the aforementioned beans
- The container gets its instructions on what objects to instantiate, configure, and assemble by reading configuration metadata

*ApplicationContext context = new ClassPathXmlApplicationContext(new String[] {"module.xml", "module2.xml"});*

*Employee emp = (Employee) context.getBean("emp");*

***9) What is Spring Bean and its properties?***

*What is Java Bean?*

*A Java Bean is a Java class that should follow following conventions:*

- *It should have a no-arg constructor.*
- *It should be Serializable.*
- *It should provide methods to set and get the values of the properties, known as getter and setter methods.*

*Spring Bean:*

*Any object in the Spring framework that we initialize through Spring container is called Spring Bean. Any normal Java POJO class can be a Spring Bean if it's configured to be initialized via container by providing configuration metadata information.*

***Properties:***

- *class*
- *name*
- *scope*
- *constructor arguments*
- *properties*
- *autowiring mode*
- *lazy-initialization mode*
- *initialization method*
- *destruction method*

***10) Explain Bean life cycle in Spring?***

***ApplicationContext context = new ClassPathXmlApplicationContext("spring.xml");***

- *The Spring Container reads the bean definitions from the Configuration file.*
- *The SpringContainer creates an instance of the Bean using Java Reflection API.*
- *If any properties are mentioned, then they are also applied. If the property itself is a Bean, then it is resolved and set.*
- *If the Bean class implements the BeanNameAware interface, then the setBeanName() method will be called by passing the name of the Bean.*
- *If the Bean class implements the BeanClassLoaderAware interface, then the method setBeanClassLoader()method will be called by passing an instance of the ClassLoader object that loaded this bean.*
- *If the Bean class implements the BeanFactoryAware interface, then the method setBeanFactory() will be called by passing an instance of BeanFactory object.*
- *If there are any BeanPostProcessors object associated with the BeanFactory that loaded the Bean, then the method postProcessBeforeInitialization() will be called even before the properties for the Bean are set.*

- *If the Bean class implements the InitializingBean interface, then the method afterPropertiesSet() will be called once all the Bean properties defined in the Configuration file are set.*
- *If the Bean definitions in the Configuration file contains a 'init-method' attribute, then the value for the attribute will be resolved to a method name in the Bean class and that method will be called.*
- *The postProcessAfterInitialization() method will be called if there are any Bean Post Processors attached for the Bean Factory object.*
- *If the Bean class implements the DisposableBean interface, then the method destroy() will be called when the Application no longer needs the bean reference.*
- *If the Bean definition in the Configuration file contains a 'destroy-method' attribute, then the corresponding method definition in the Bean class will be called*

***11) What are the scopes of Bean?***

| Scope | Description |
|---|---|
| Singleton | (Default) Scopes a single bean definition to a single object instance per Spring IoC container. |
| Prototype | Scopes a single bean definition to any number of object instances. |
| Request | Scopes a single bean definition to the lifecycle of a single HTTP request; that is, each HTTP request has its own instance of a bean created off the back of a single bean definition. Only valid in the context of a web-aware Spring ApplicationContext. |
| Session | Scopes a single bean definition to the lifecycle of an HTTP Session. Only valid in the context of a web-aware Spring ApplicationContext. |
| Application | Scopes a single bean definition to the lifecycle of a ServletContext. Only valid in the context of a web-aware Spring ApplicationContext. |
| Websocket | Scopes a single bean definition to the lifecycle of a WebSocket. Only valid in the context of a web-aware Spring ApplicationContext. |

***12) Explain about spring-config.xml file and does application can have multiple spring-config.xml files?***

```
<beans>
<importresource="application1.xml"/>
<importresource="resources/applicaiton2.xml"/>
<importresource="/resources/application3.xml"/>

<beanid="bean1"class="com.sivajavatechie.spring.Employee"/>
<beanid="bean2"class=" com.sivajavatechie.spring.Address"/>
</beans>
```

***13) What are bean wiring, auto wiring and different types of auto wirings in SpringFramework?***

| Mode | Explanation |
|---|---|
| No | (Default) No autowiring. Bean references must be defined via a ref element. Changing the default setting is not recommended for larger deployments, because specifying collaborators explicitly gives greater control and clarity. To some extent, it documents the structure of a system. |
| byName | Autowiring by property name. Spring looks for a bean with the same name as the property that needs to be autowired. |
| byType | Allows a property to be autowired if exactly one bean of the property type exists in the container. If more than one exists, a fatal exception is thrown, which indicates that you may not use byType autowiring for that bean. If there are no matching beans, nothing happens; the property is not set. |
| constructor | Analogous to byType, but applies to constructor arguments. If there is not exactly one bean of the constructor argument type in the container, a fatal error is raised. |

***14) What is the purpose of @Component, @Service, @Controller and @Repository?***

***@Controller****: where your* ***request mapping from presentation page*** *done i.e. Presentation layer won't go to any other file it goes directly to @Controller class and check for requested path in @RequestMapping annotation which written before method calls if necessary.*

***@Service:*** *All business logic is here i.e. Data related calculations and all. This annotation of business layer in which our user not directly call persistence method so it will call this method using this annotation.* ***It will request @Repository as per user request***

***@Repository:****This is Persistence layer (Data Access Layer) of application which used to get data from database. i.e.* ***all the Database related operations are done by repository.***

***@Component*** *- Annotate your other components with component stereotype.*
*Example:*

```
@Component
@Scope("prototype")
public @interface Bank {..}
```