Spring Interview Questions.

------------------------------------

1. what is IOC (inversion of control) Container.

------------------------------------------------------

-> Ioc is a principle or paradigm. we have some set of rules or guidelines to develop a application in a decouple manner.

-> ioc is a collabrating the object and managing the lifecycle of those objects is called ioc container.

-> Ioc container says that you don't bother about object creation or you don't create your objects only describe how they should be created i will manage it.

-> The basic concept of the Dependency Injection or Inversion of Control is that, programmer do not need to create the objects,

instead just describe how it should be created.

Benefits of Ioc.

--------------------

-> minimize the code in our application.

-> it provides loose coupling between components in our application.

-> if any modification have to do then it doesnt effect other components.

-> it supports eager instantiation and lazy instantiation of services.

-> enhancement will be easy.

2. what is dependency injection. types of dependency injection.

--------------------------------------------------------------------

-> dependency injection is a software design pattern that deals with how components are orgnizing their dependencies.

-> it is the process of injecting the dependencies in dependent class automatically. we no need to create obj and no need to map with obj.

-> Dependency Injection means injecting the dependency between two object as per as our requirement in our application,

this help to reducing the dependency to each other.

-> Dependency Injection (DI) is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application.

Dependency Injection makes our programming code loosely coupled

-> it is internally follow strategy design pattern .

menas. favour composition over inheritance.

. always design to interface never code to implementation.

3. diff between setter injection and constructor injection.

-------------------------------------------------------------------

-> Setter Injection.

------------------------

-> to perform setter injection we will use <property> tag.

-> partial dependencies is possible. means if we have 3 dependencies like int, long, String it is not necessary to inject all values.

-> if we have more dependencies eg 15 to 20 are there in our bean class then in this case setter injection is not recomended to use as we need to

write almost 20 setters right bean lenght will be increased.

-> setter injection makes bean class object as mutable( we can change).

-> setter injection support cyclic dependencies.

Constructor injection.

---------------------------

-> to perform constructor injection <constructor-arg> tag is required.

-> partial injection of dependencies cannot be possible becoz for calling a constructor we must pass all the arguments.

-> if we have more dependencies in this case constructor injection is highly recomended to use becoz we can inject all the dependencies with in

the 3 to 4 lines (by calling one constructor).

-> constructor injection makes bean class obj is immutable( we cannot change).

-> constructor injection doesnt support cyclic dependencies.

4. diff between BeanFactory and application context.

-----------------------------------------------------------

BeanFactory

----------------

-> BeanFactory is a basic container. it can only manage a bean life cycle. but it can not provide service like transaction, security etc.

-> if we developing small scale application like mobile application embeded system then we use beanfactory.

-> Beanfactory is lazy initializer. beanfactory container will not create a bean obj upto the request time.

-> Beanfactory container supports only two scope(singeltone & prototype).

-> Beanfactory doesnt support internationalization, event handling, event processing.

Application Context

-----------------------

-> ApplicationContext is a advanced container it manage bean life cycle and also provide transaction security etc.

-> if we are developing enterprise application like(web application, distributed application) then ApplicationContext is recomended to use.

-> ApplicationContext container creates bean object of singelton bean at the time of loading only.it is eager initialzer.

-> ApplicationContext container support all the bean scope (singletone,prototype, session,request).

-> it supports internationalization, event handling, event processing also.

5. what is bean autowiring and types of autowire modes.

------------------------------------------------------------

-> injecting the dependencies between the obj is called wiring.

-> instead of telling the spring to manage the dependency by writting <property> or <constructor> tag in spring bean configuration file.

-> if we instruct the spring to automatically detect the dependencies and perform the injection between them it is called bean autowiring.

-> it is used only when rapid application development is required.

-> In Spring framework, you can wire beans automatically with auto-wiring feature. To enable it, just define the “autowire” attribute in.

-> The Spring container can autowire relationships between collaborating beans without using and elements which helps cut down on the amount of XML configuration.

<bean id="countryBean" class="org.arpit.java2blog.Country" autowire="byName">

Modes of Autowire

------------------------

1. Autowire= "byname"

--------------------------

-> if u enable autowiring byname, spring will inject the bean based on property name. it uses setter method.

-> Autowiring by property name. Spring container looks at the properties of the beans on which autowire attribute is set to byName in the XML configuration file

and it tries to match it with name of bean in xml configuration file.

2. Autowire= "bytype"

---------------------------

-> if u enable autowire bytype, spring will inject the beans based on the property type. it uses setter method.

-> Autowiring by property datatype. Spring container looks at the properties of the beans on which autowire attribute is set to byType in the XML configuration file.

-> It then tries to match and wire a property if its type matches with exactly one of the beans name in configuration file

3. Autowire= "byconstructor"

-------------------------------

-> if u enable autowire byconstructor, spring will injects the beans uses constructor.

-> byType mode in constructor argument.

4. Autowire="byautodetect"

------------------------------

-> Spring first tries to wire using autowire by constructor, if it does not work, Spring tries to autowire by byType.

6. what is bean scope. and types of bean scope. diff between singletone and prototype bean scope.

--------------------------------------------------------------------------------------------------------------

-> Beanscope is a concept which is provided by spring people. in spring when u declare a class as a bean by default the bean will be created under

the singleton scope.

Types of beanscope

-------------------------

1. singleton

----------------

-> bydefault every bean declared in the configuration file is singleton.

-> Scopes a single bean definition to a single object instance per Spring IoC container.

-> singleton is default scope of a bean in Spring. You have to explicitly change scope of a bean if you want different scope.

-> if beanscope is singleton then Ioc container creates the bean class obj and keeps in the HashMap element as value by having bean 'id' as key and

uses that obj across the multiple "factory.getBean() method.

2. Prototype

-----------------

-> prototype – Return a new bean instance each time when requested.

-> when we declare a beanscope as a prototype then Ioc container doesnt keep the created bean class object in HashMap so it returns new obj for every

factory.getBean().

3. Request

----------------

-> request – Return a single bean instance per HTTP request.

-> when we declare a beanscope as request ,for every Http request new bean instance will be injected.

4. Session

---------------

-> session – Return a single bean instance per HTTP session.

-> for every new Http session ,new bean instance will be injected.

5. Global session

---------------------

-> the global session scope is depricated in the market from spring 3.0.

-> Return a single bean instance per global HTTP session.

7. what is bean life cycle .

-------------------------------------------

-> every obj in this world have life cycle. whatever the obj is performing after the birth and before the death is known as life cycle of the obj.

-> the spring container find the beans defination from the xml file and instantiate the bean .

-> using dependency injection spring populates all of the properties as specified in the bean defination.

-> in servlet life cycle we are used following life cycle methods . 1> init() 2> service() 3> destroy().

-> spring bean allows two life cycle methods. 1> init() 2> destroy().

Spring Aop

-----------------

1. what is Aop. what is the principles of Aop. and where we apply Aop in projects.

----------------------------------------------------------------------------------------

-> Aop is not a programming language. it is a methodlogy or principles like oops. we have some set of rules or guidelines to make our application

in decouple manner.

-> Aop is the process of separating the primary logic from the secondry logic (crosscutting logic).

-> in every application there will be two types of logic, one is called primary buisness logic and other one is helper logic which makes your

primary buisness logic work better.

-> in the enterprise level application we used to add diff crosscutting functionlaties (means adding diff types of services to the application at runtime).

We can apply Aop

--------------------

-> Logging.

-> Auditing.

-> Security.

-> transaction.

-> caching.

-> performance monitoring.

Principles of Aop

----------------------

-> Aspect

-> Advice

-> joinpoint

-> pointcut

-> weaving

-> target

-> proxy

1.> Aspect.

---------------

-> it is the piece of code which will be separate from the primary logic. aspect represent secondry or crosscutting logic.

-> it is the piece of code that has to be applied across various classes of the application.

-> An Aspect is a class that implements concerns that cut across different classes such as logging. It is just a name.

2.> Advice.

----------------

-> Action taken by aspect at particular join point. For example: Before execution of getEmployeeName() method, put logging. So here, we are using before advice.

-> this principles talks about where actually we can apply that aspect.

a.> before advice.

b.> around advice.

3.> after returning advice

d.> throws advice.

3.> pointcut.

-------------------

-> Pointcut is an expression that decides execution of advice at matched joint point. Spring uses the AspectJ pointcut expression language by default.

-> it is the set of joinpoint where advice are applied to excute the aspect.

4.> joinpoint.

---------------------

-> this principles will talks about how many places we can advice the aspect .generally in spring you can apply an aspect of method execution.

-> It is a point in execution of program such as execution of method. In Spring AOP, a join point always represents a method execution.

5.> Target.

--------------------

-> the class on which you want to advice the aspect.

6.> weaving.

--------------------

-> the process of advising a target class with an aspect based on a pointcut to build proxy.

7.> proxy.

------------

-> the outcome of weaving is called proxy.

Spring MVC

---------------

1. what do you mean by MVC.

---------------------------------

-> it is a software design pattern. it is a structural (creational) design pattern comes in gang of four design pattern.

-> it provides loose coupling between model, view and controller.

model

---------

-> it is responsible for data storage related logic.

view

-------

-> it is representation for presentation logic.

Controller.

--------------

-> it is responsible for application execution logic or processing logic. it acts as a mediator between model and view.

2. what is spring MVC flow. how it works.

-----------------------------------------------

-> spring mvc is used for making a web application development faster, cost effective and flexiable.

-> in case of spring mvc user has send the request to the dispather servlet. dispatcher servlet acts as a front conroller with extension \*.mvc,\*.html etc.

-> the dispatcher servlet forward the request to the handler mapping.

-> handler mapping identify the appropriate handler or controller class for the given url request and sends to the dispatcher servlet.

-> the controller class perform operation like dao or buisness method according to the buisness requirement.then the resultant page with data will be forward to

dispatcher servlet.

-> the dispatcher servlet forward it with logical view name to view resolver.

-> view resolver resolves it and create the view obj.

-> it calls the render method on view obj then the view page will be displayed.

3. what is controller. how many types of controller. which controller u are used in project. and how to create controller class in spring mvc.

----------------------------------------------------------------------------------------------------------------------------------------------------

-> controller is a simple java class.

-> the controller class is act as a mediator between front controller to service components.

-> the controller classes are creating by application developer to communicate with service layer.

-> the controller classs method always returns model and view.

types of controller.

--------------------------

a> simple form controller.

b> abstract controller.

c> abstract command controller.

-> To create a Controller in Spring MVC, create a class and annotate it with @Controller and @RequestMapping.

@Controller

@RequestMapping("/page")

public class PersonController {

@Autowired

private IPersonService personService;

@RequestMapping("/login")

public String hello(@RequestParam(value="userId", required=false) String userId,

@RequestParam(value="location", required=false) String location,

Model model) {

model.addAttribute("msg", "Hello "+personService.getPersonName() );

model.addAttribute("userId", userId);

model.addAttribute("location", location);

return "result";

}

}

4. whtat is dispather servlet.

-----------------------------------

-> dispatcher servlet act as a front controller.

-> every incoming http request is accepting and proceesing by dispatcher servlet.

-> the dispatcher servlet we will configure in web.xml just like a normal servlet.

-> To work with Spring MVC, we need to define it in our web.xml .

<servlet>

<servlet-name>dispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

5. what is handler mapping . and which handler mapping u are used in project.

----------------------------------------------------------------------------------

-> handler mapping are used to map an incomming http request url with a controller class.

-> whenever the request coming from client.the dispatcher servlet will delegate the request obj to handler mapping to identify the appropriate controller class.

-> handler mapping beans we can configure in spring configuration class.

types of handler mapping.

------------------------------

1.> simple url handlere mapping.

2.> bean name url handler mapping.

3.>. handler interceptors.

6. what is view resolver. and which view resolver u are used in project.

-----------------------------------------------------------------------------

-> the view resolver are used to find out the actual view obj for the given logical view name.

-> after getting the model & view object, the dispatcher servlet will delegate the logical view name to view resolver to identify actual view object.

types of view resolver.

-------------------------------

a.> xml view resolver.

b.> url based view resolver.

c.> resource bundle view resolver.

7.> what is model and view.

------------------------------------

-> model and view is predefined classes. the model and view object is holding model data and logical view name.

How to handle views in Spring MVC using XML.

------------------------------------------------------

Ans: To handle views in Spring MVC, we need to configure InternalResourceViewResolver bean in spring XML where we need to define prefix and suffix of our views name.

Find the sample declaration.

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/pages/"/>

<property name="suffix" value=".jsp"/>

</bean>

What is minimum web.xml configuration to run Spring MVC.

----------------------------------------------------------------

-> To run the Spring MVC, we need to define DispatcherServlet, contextConfigLocation and ContextLoaderListener in web.xml. Find the sample web.xml.

<servlet>

<servlet-name>dispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/dispatcher-servlet.xml</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

8. which annotation u are used in project. can us list them.

-----------------------------------------------------------------

-> 1.> @Autowired

-----------------------

-> this annotation can be applied at field level,setter method level,constructor level and orbitary level .

-> when we write @Autowire then bydefault an attribute become true ie required =true, if we use @Autowire annotations then that setter or constructor or

attributes become mandatory.

2.> @Qualifier

------------------------

-> it is used to control which bean should be autowire on a field.

-> You can have more than one bean of same type in your XML configuration but you want to autowire only one of them ,

so @Qualifier removes confusion created by @Autowired by declaring exactly which bean is to autowired.

@Required

---------------------

-> This annotation simply indicates that the affected bean property must be populated at configuration time:

either through an explicit property value in a bean definition or through autowiring.

-> Suppose you have very large application and you get NullPointerExceptions because required dependency has not been injected then .

it is very hard to find out what goes wrong.So this annotation helps us in debugging the code.

Streotype Annotations

----------------------------

-> streotype annotation are used to class level. it mean we can write these annotations only at on the top of the class.

-> these annotations are used to detect classes and creates object without xml <bean> code.

-> another way to make our class as bean and place into ioc container we have to use streotype annotations.

1.> @Ccomponent (utils, validator).

-------------------------------------------

-> by using @Component we can make our class as bean and ioc container will take the class and places to the ioc container as a bean.

2.> @Service (service layer).

---------------------------------

-> @Service annotation also used for making our class as bean into ioc container.

-> it indicates a service component in the buisness layer.

3.> @Repositry (data access layer)

--------------------------------------

-> it indicates Dao component in the presentation layer.

-> @Repositry annotation also used for making our class as a bean into ioc container.

-> @Repositry indicates the class contains dao persistency logic.

4.> @Controller (presentation layer)

--------------------------------------------

-> it indicates a class has presentation logic.

-> it is also used for making our class as beans into ioc container.

-> it is Used at the class level.

-> it Tells the spring framework that the marked class acts as a controller.

@Controller

public class EmployeeController{

}

@RequestMapping

-------------------

-> it Can be used at the class level and method level in controllers.

Spring Security

----------------------

1. what is spring security. how it works.

------------------------------------------------

-> spring security provides various security features like: authentication, autherization to create secure java enterprise application.

-> authentication is the process of knowing and identifying the user that wants to access.

-> autherization is the process to allow authority to perform actions in the application.

-> username - identification.

password - verification.

-> dependencies needed developing spring boot security application.

step-1 -> add spring security dependencies in pom.xml.

--------------------------------------------------------------

1.> spring-security-core.

2.> spring-security-web.

3.> spring-security-config.

step-2 -> create a separate xml file for security configurations (security-context.xml).

<!-- we will be defining all security related confuguration in this file-->

<http use-expressions = "true">

<intercept-url pattern="/\*\* access = "isAuthenticated()">

<!--this means all url in this app will be checked if user is authenticated-->

<form-login/>

<!-- we will just use the built in form login page in spring-->

<logout logout-url = "/logout" logout-success-url="/index"/>

<!--the logout url we wil use in jsp-->

</http>

<authentication-manager>

<authentication-provider>

<user-service>

<user name = "mohan" password = "kumar" authorities = "admin, user"/>

<user name = "gandhi" password = "rakesh" authorities ="user"/>

</user-service>

<authentication-manager>

<authentication-provider>

step-3 -> configure web.xml to include spring security.

<filter>

<filter-name>springSecurityFilterChain</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

</filter>

<filter-mapping>

<filter-name>springFilterChain</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

-> all request now will go through "sprigSecurityFilterChain" filter which will apply app security.

LDAP (light weight directory access protocol) important dependencies.

-----------------------------------------------------------------------

-> spring-ldap-core.

-> spring-security-ldap.

-> unbound id-ldap sdk.

what is delagating filter proxy.

------------------------------------------

-> it provides the link between web.xml and the application context.

<filter>

<filter-name>myFilter</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

</filter>

<filter-mapping>

<filter-name>myFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

19. what is inheritance mapping in hibernate .and how many types of inheritance mapping.