**JSP**

1. What is JSP?
2. Explain implicit objects in JSP.

Objects created by web container and contain information regarding a particular request, application or page are called Implicit Objects. They are :

1)response

2)exception

3)application

4)request

5)session

6)page

7)out

8)config

9)pageContext

1. **Differentiate between response.sendRedirect(url) and <jsp:forward page = …>**

<jsp.forward> element forwards the request object from 1 JSP file to another. Target file can be HTML, servlet or another JSP file, but it should be in the same application context as forwarding JSP file.

sendRedirect send HTTP temporary redirect response to the browser. The browser then creates a new request for the redirected page. It kills the session variables.

1. How to include static files in a JSP page?

Static pages are always included using JSP include directive. This way the inclusion is performed in the translation phase once. Note that a relative URL must be supplied for file attribute. Although static resources may be included, it is not preferred as each request requires inclusion.

1. **How can a thread safe JSP page be implemented?**

It can be done by having them implemented by the SingleThreadModel Interface. Add <%@page isThreadSafe=”false” %> directive in the JSP page.

1. **How can the output of JSP or servlet page be prevented from being cached by the browser?**

Using appropriate HTTP header attributes to prevent the dynamic content output by a JSP page from being cached by the browser.

<%

response.setHeader(“Pragma”,”no-cache”); //HTTP 1.0  
response.setHeader(“Cache-Control”,”no-store”); //HTTP 1.1

%>

1. **How to restrict page errors display in a JSP page?**

By setting up an “ErrorPage”  attribute of PAGE directory to the name of the error page in the JSP page, and then in the error jsp page set “isErrorpage=”TRUE”, Errors can be stopped from getting displayed.

1. **What are JSP Actions?**

They are XML tags, which direct the server to using existing components or control behavior of JSP Engine. They consist of a typical prefix of “jsp:” and action name.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <jsp:include/>  <jsp:getProperty/>  <jsp:forward/>  <jsp:setProperty/>  <jsp:usebean/>  <jsp:plugin/> |

1. **Differentiate between <jsp:include page=…> and <%@include file=…>.**

Both these tags include information from 1 page to another.

The first tag acts as a function call between two Jsp’s. It is executed each time client page is accessed by the client. It is useful to modularize the web application. New content is included in the output.

The second tag content of file is textually embedded having similar directive. The changed content is not included in the output. It is helpful when code from one jsp is required by several jsp’s.

1. **Can constructor be used instead of init(), to initialize servlet?**

Yes, it is possible. But it is not preferred because init() was developed because earlier Java versions could not invoke constructors with arguments dynamically. So they could not assign a servletConfig. Today, however, servlet containers still call only no-arg constructor. So there is no access to servletContext or servletConfig.

1. **Explain lifecycle methods.**
2. **Define Expression**

Expression tag is used to insert Java values directly in the output. Its syntax is <%=expression%>

1. **Define JSP Scriptlet.**

It a JSP tag that encloses Java code in JSP pages. Their syntax is <% %>. Code written in scriptlet executes every time the program is run.

1. **How can information from one JSP be passed to another JSP?**
2. **Explain the uses of <jsp:usebean> tag.**
3. **Explain handling of runtime exceptions.**

We can use the errorPage attribute of the page directive to have uncaught run-time exceptions automatically forwarded to an error processing page.

Example: <%@ page errorPage="error.jsp" %>

It will redirect the browser to the JSP page error.jsp if an uncaught exception is encountered during request processing. Within error.jsp, will have to indicate that it is an error-processing page, using the directive: <%@ page isErrorPage="true" %>

1. **Explain the various scope values for <jsp:useBean> tag.**

<jsp:useBean> tag is used to use any java object in the jsp page. Some scope values are :

1)application

2)request

3)page

4)session

1. **Show the 2 types of comments in JSP.**

The 2 types are :

|  |  |
| --- | --- |
| 1  2 | <%–JSP Comment–%>  <!–HTML comment–> |

1. **Explain JSP directives.**

JSP directives are messages to JSP Engine. They serve as a message from page to container and control the processing of the entire page. They can set global values like class declaration. They do not produce output and are enclosed in <%@….%>

1. **Explain page Directives.**

Page Directives inform the JSP Engine about headers and facilities that the page receives from the environment. It is found at the top of all JSP pages. Its syntax is <%@ page attribute=”value”>

1. **Show attributes of page directives.**

1)Session : It shows if a session data is available to the page.

2)Import : it shows packages that are imported.

3)isELIgnored : It shows whether EL expressions are ignored when JSP translates into a servlet.

4)contentType : it allows the user to specify the content type of page.

1. **What is Include directive?**

The include directive statically inserts the contents of a resource into the current JSP. It helps in the reuse of code without duplication. and includes contents of the file at translation time. Its syntax is as follows <%@ include file=”Filename”%>.

1. **How can Automatic creation of session be prevented in a JSP page?**

JSP page automatically create sessions for requests. By typing the following, it can be avoided.

<%@ page session=”false”  %>

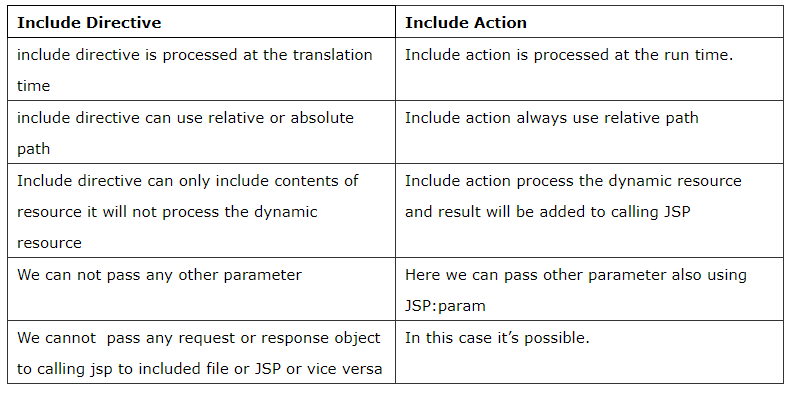
1. **Explain the jspDestroy() method.**

Whenever a JSP page is about to be destroyed, the container invokes the jspDestroy() method from the javax.servlet.jsp.JspPage interface. Servlets destroy methods are similar to it. It can be easily overridden to perform cleanup, like when closing a database connection.

Explain include Directive and include Action of JSP  
  
Syntax for include Directive is **<%@ include file="fileName" %>** which means we are including some file to our JSP Page when we use include directive contents of included file will be added to calling JSP page at translation time means when the calling JSP is converted to servlet ,all the contents are added to that page .one important thing is that any JSP page is complied if we make any changes to that particular page but if we have changed the included file or JSP page the main calling JSP page will not execute again so the output will not be according to our expectation, this one is the main disadvantage of using the include directive that why it is mostly use to add static  resources, like Header and footer .

Syntax for include action is **<jsp:include page=”relativeURL” />** it’s a runtime procedure means the result of the JSP page which is mentioned in relative URL is appended  to calling JSP at runtime on their response object at the location where we have used this tag

So any changes made to included page is being effected every time, this is the main advantage of this action but only relative URL we can use here ,because request and response object is passed between calling JSP and included JSP.



1. **Is it possible for one JSP to extend another java class if yes how?**

**Ans:** Yes

1. **Why is \_jspService() method starting with an '\_' while other life cycle methods do not?**

Ans: main JSP life cycle method are jspInit() jspDestroy() and \_jspService() ,bydefault whatever content we write in our jsp page will go inside the \_jspService() method by the container if again will try to override this method JSP compiler will give error but we can override other two life cycle method as we have implementing this two in jsp so making this difference container use \_ in jspService() method and shows that we cant override this method.

1. **What do you understand by context initialization parameters?**  
     
   The context-param element contains the declaration of a web application's servlet context initialization parameters.   
     
   < context - param >  
   < param - name > name < / param - name > < param - value > value < / param - value >  
   < / context-param >  
     
   The Context Parameters page lets you manage parameters that are accessed through the ServletContext.getInitParameterNames and ServletContext.getInitParameter methods.
2. What is the difference between <jsp:forward page = ... > and response.sendRedirect(url)?

The <jsp:forward> element forwards the request object containing the client request information from one JSP file to another file. The target file can be an HTML file, another JSP file, or a servlet, as long as it is in the same application context as the forwarding JSP file.

sendRedirect sends HTTP temporary redirect response to the browser, and browser creates a new request to go the redirected page.

1. **- What are the different JSP scripting elements?**

There are three types of scripting language elements:

* Declarations,
* Scriptlets, and
* Expressions.

**17- What is the logic behind a scriptlet in JSP?**

A scriptlet holds the executable Java code which runs whenever the JSP gets loaded. The scriptlet passes its code to the service() method while the JSP is getting compiled to a servlet. So all the scriptlet variables and methods become local to the service() method. A scriptlet is coded between the <% and %> tags and the container call it while processing the request.

**18- What is a JSP declaration?**

JSP Declarations help to declare the class variables and methods in a JSP page. They get initialized along with the initialization of the class. Everything in a <declaration> is accessible to the whole JSP page. You can encircle a declaration block within the **<%! And %>**tags.

**19- What is a JSP expression?**

A JSP expression helps to write an output without using the <out.print statement>. You can see it as a shorthand description for the scriptlets. And as usual, you delimit an expression using the **<%= and %>**tags.

Ending the expression with a semicolon is not mandatory, as it gets automatically added to the <expressions> within the expression tags.

**20- What do you understand of the JSP directives?**

* JSP directives are the instructions for the JSP container to control the processing of the whole page.
* They add the ability to set global values such as a class declaration, method definition, output data type, etc.
* They don’t send any output to the client.
* All directives should get enclosed within <%@   %> tag.

e.g. page and include directive, etc.

**21- What do you understand of the page directive?**

* It notifies the JSP container of the headers (facilities) that the page receives from the environment.
* Usually, the page directive remains at the top of a JSP page.
* A JSP page can have any number of page directives if the attribute – value pair is unique.
* The include directive syntax is: <%@ page attribute=”value”>

e.g.:<%@ include file="header.jsp" %>

**22- Explain the different attributes of a page directive?**

There are about 13-attributes available for a page directive. Some of the important ones are as follows:

* **<import>**: Signifies the packages that get queued for import.
* **<session>**: Specifies the session data available to the JSP page.
* **<contentType>:**Allows a user to update the content-type for a page.
* **<isELIgnored>**: Specifies if an EL expression gets ignored during the translation of the JSP to a servlet.

**23- What is the include directive?**

Its purpose is to attach the static resources to the current JSP page during the translation process. The include directive syntax is as follows:

<%@ include file = “File-Name” %>

* The include directive statically embeds the contents of a resource into the current JSP.
* It allows a user to reuse the code without duplicating it and inserts the data of the target file during JSP translation.
* This directive has only one attribute called <file> that specifies the name of the file to include.

**24- What is the significance of the JSP standard actions and what is their purpose?**

* The standard actions in JSP not only control the runtime behavior of theJSP page, but they do affect the response posted back to the client-side.
* We use them for the following purpose:
* Include a file at the request time,
* Locate or instantiate a JavaBean,
* Forward a request to a new page, or
* Generate a browser-specific code, etc.
* Some of them you can see in the below example.

e.g.: include, param, useBean, etc.

**25- What are the standard actions available in JSP?**

Please refer the below list:

* **<jsp: include>**: Used to specify a response from the servlet or a JSP page into the current page.
* **<jsp: forward>**: Used to send a reply from the servlet/JSP page to another page.
* **<jsp: useBean>**: Allows a JavaBean to get accessible from a page and initializes the bean.
* **<jsp: setProperty>**: Used to set the JavaBean properties.
* **<jsp: getProperty>**: Fetches the property value from a JavaBean component and appends it to the response.
* **<jsp: param>**: Used with actions like <jsp:forward> and <jsp:, or plugin> to append a parameter to the request.
* **<jsp: plugin>**: Specifies whether to add a Java applet or a JavaBean to the current JSP page.

**26- What is the purpose of the <jsp: useBean> action?**

The <jsp: useBean> standard action allows to identify an existing JavaBean or to create the one if it doesn’t exist.

It contains attributes to classify the object instance, to specify the lifetime of the bean, as well as the fully-qualified classpath and type.

**27- Define the scopes available with the <jsp: useBean>?**

* **Page Scope:** Tells the bean object is available for the entire JSP page without any external access.
* **Request Scope:** Signifies the object can link with a particular request and persist till the time request lasts.
* **Session Scope:** States that the bean object is available throughout the session.
* **Application Scope:** Specifies the bean object is available throughout the entire Web application discarding any external access.

**28- What is the purpose of the <jsp: forward> action?**

* The <jsp: forward> standard action forwards a response from a servlet or a JSP page to another page.
* The execution of the current page gets stopped, and control shifts to the forwarded page.
* The <jsp: forward> standard action uses the below synatx.

<jsp:forward page=”/targetPageTemplate” />

Here, targetPage could either be a JSP/HTML page, or a servlet within the same context.

* If anything gets written to the output stream but not buffered before the <jsp: forward>, it’ll result in an IllegalStateException.

Note : Before you use <jsp:forward> or <jsp:include> in a page, make sure the buffering is on. Though, the buffer is enabled by default.

**41- What is the purpose of the <jsp: include> standard action?**

The <jsp: include> standard action instructs the JSP page to include a static/dynamic resource at run-time. Unlike the include directive, the include action is best suited for the resources that undergo frequent changes. The resources you wish to add must be in the same context. The syntax of the <jsp: include> standard action is as follows:

<jsp:include page="targetPage" flush="true"/>

Here, the targetPage is the page to be included in the current JSP page.

**SERVLET**

**1. Can We Use The Constructor, Instead Of Init(), To Initialize Servlet?**

Yes. But you will not get the Servlet specific things from constructor. The original reason for init() was that ancient versions of Java couldn’t dynamically invoke constructors with arguments, so there was no way to give the constructor a ServletConfig. That no longer applies, but servlet containers still will only call your no-arg constructor. So you won’t have access to a ServletConfig or ServletContext.

**2.  What Is Servlet Context?**

The Servlet context is an object that contains a information about the Web application and container. Using the context, a Servlet can log events, obtain URL references to resources, and set and store attributes that other Servlet in the context can use.

**3. How Would You Create Deadlock On Your Servlet?**

Calling a doPost() method inside doGet() and doGet()method inside doPost() wouleate a deadlock for a servlet.

**4 Why Is Httpservlet Declared Abstract?**

1.The default implementations of the main service methods can not do anything and need to be overridden. This calls of the HttpServlet class to be declared as abstract.  
2.With its use the developers do not need to implement all the service methods.

**5 What Is Genericservlet Class?**

1.GenericServlet is an abstract class which implements the Servlet interface and the ServletConfig interface.  
2.Other than the methods included in above two interfaces, it also provides simple versions of the lifecycle methods init and destroy, and implements the log method declared in the ServletContext interface.  
3.Since this class is not specific to any protocol, it is known as generic servlet.

**6 How Can The Session In Servlet Be Destroyed?**

There are two ways to destroy a session:  
1. **Programatically :** By using session.invalidate() method. It makes the container abandon the session on which the method is called.  
2. When the server shuts down.

1. **What Is Lazy Loading?**

The servlets are not initialized by the container from the start. It happens when the servlet is requested for the first time. This is called lazy loading.

1. **What Are The Mechanisms Used By A Servlet Container For Maintaining Session Information?**

For maintaining session information Servlet Container uses:  
. Cookies  
. URL rewriting  
. HTTPS protocol information

1. **What Are The Uses Of Servletrequest?**

The ServletRequest gives information such as the names of the parameters passed by the client, the protocol (scheme) being used by the client, and the names of the remote host that made the request and the server that received it. The input stream, ServletInputStream.

1. **What Are The Uses Of Servletresponse Interface?**

ServletResponse allows the servlet to set the content length and MIME type of that response. It provides an output stream, ServletOutputStream and a Writer through which the servlet can send data.

1. **What Is Servlet Chaining?**

Servlet chaining is a technique in which two or more servlets can cooperate in servicing a single request. In servlet chaining, one servlet’s output is the input of next servlet. This process continues until the last servlet is reached. Its output is then sent back to the client. We are achieving Servlet Chaining with the help of RequestDispatcher.

1. **What Is The Difference Between Context Init Parameter And Servlet Init Parameter?**

Servlet init parameters are for a single servlet only. No body out side that servlet can access that. It is declared inside the <servlet> tag inside Deployment Descriptor, where as context init parameter is for the entire web application. Any servlet or JSP in that web application can access context init parameter. Context parameters are declared in a tag <context-param> directly inside the <web-app> tag. The methods for accessing context init parameter is getServletContext ().getInitParamter (“name”) where as method for accessing servlet init parameter is getServletConfig ().getInitParamter (“name”);

1. **What Are The Differences Between A Session And A Cookie?**

Session is stored in server but cookie stored in client. Session should work regardless of the settings on the client browser. There is no limit on the amount of data that can be stored on session. But it is limited in cookie. Session can store objects and cookies can store only strings. Cookies are faster than session.

1. **What's The Servlet Interface?**

The central abstraction in the Servlet API is the Servlet interface. All servlets implement this interface, either directly or, more commonly, by extending a class that implements it such as HttpServlet.

1. **What Is The Difference Between Servletcontext And Servletconfig?**

The ServletConfig gives the information about the servlet initialization parameters. The servlet engine implements the ServletConfig interface in order to pass configuration information to a servlet. The server passes an object that implements the ServletConfig interface to the servlet's init() method.  
The ServletContext gives information about the container. The ServletContext interface provides information to servlets regarding the environment in which they are running. It also provides standard way for servlets to write events to a log file.

1. When should you prefer to use doGet() over doPost()?

GET is preferred over POST in most of the situations except for the following:  
  
- When the data is sensitive.  
- When the data is greater than 1024 characters.

## Explain the life cycle of Servlet, i.e. Instantiation, Initialization, Service, Destroy

1. The Java \_\_\_\_\_\_\_\_\_\_ specification defines an application programming interface for communication between the Web server and the application program.  
   a) Servlet  
   b) Server  
   c) Program  
   d) Randomize  
   Answer:a  
   Explanation: Servlets are commonly used to generate dynamic responses to HTTP requests.
2. The doGet() method in the example extracts values of the parameter’s type and number by using \_\_\_\_\_\_\_\_\_\_  
   a) request.getParameter()  
   b) request.setParameter()  
   c) responce.getParameter()  
   d) responce.getAttribute()  
   View Answer

Answer: a  
Explanation: These methods uses these values to run a query against a database.

1. How many JDBC driver types does Sun define?  
   a) One  
   b) Two  
   c) Three  
   d) Four  
   View Answer

Answer: d  
Explanation: JBDB.DriverManager.getConnection() is used to get the connection to the database.

1. Which tag should be used to pass information from JSP to included JSP?  
   a) Using <%jsp:page> tag  
   b) Using <%jsp:param> tag  
   c) Using <%jsp:import> tag  
   d) Using <%jsp:useBean> tag  
   View Answer

Answer: a  
Explanation: <%jsp:page> tag is used to pass information from JSP to included JSP.

1. \_jspService() method of HttpJspPage class should not be overridden.  
   a) True  
   b) False  
   View Answer

Answer: a  
Explanation: \_jspService() method is created by JSP container. Hence, it should not be overridden.

1. Which option is true about session scope?  
   a) Objects are accessible only from the page in which they are created  
   b) Objects are accessible only from the pages which are in same session  
   c) Objects are accessible only from the pages which are processing the same request  
   d) Objects are accessible only from the pages which reside in same application  
   View Answer

Answer: b  
Explanation: Object data is available till session is alive.

1. Default value of autoFlush attribute is?  
   a) true  
   b) false  
   View Answer

Answer: a  
Explanation: Default value “true” depicts automatic buffer flushing.

1. Which one is the correct order of phases in JSP life cycle?  
   a) Initialization, Cleanup, Compilation, Execution  
   b) Initialization, Compilation, Cleanup, Execution  
   c) Compilation, Initialization, Execution, Cleanup  
   d) Cleanup, Compilation, Initialization, Execution  
   View Answer

Answer: c  
Explanation: The correct order is Compilation, Initialization, Execution, Cleanup.

1. “request” is instance of which one of the following classes?  
   a) Request  
   b) HttpRequest  
   c) HttpServletRequest  
   d) ServletRequest  
   View Answer

Answer: c  
Explanation: request is object of HttpServletRequest.

1. Which is not a directive?  
   a) include  
   b) page  
   c) export  
   d) useBean  
   View Answer

Answer: c  
Explanation: Export is not a directive.

1. Which is mandatory in tag?  
   a) id, class  
   b) id, type  
   c) type, property  
   d) type,id  
   View Answer

Answer: a  
Explanation: The useBean searches existing object and if not found creates an object using class.

1. Which one of the following is correct for directive in JSP?  
   a) <%@directive%>  
   b) <%!directive%>  
   c) <%directive%>  
   d) <%=directive%>  
   View Answer

Answer: a  
Explanation: Directive is declared as <%@directive%>.

1. Which of the following action variable is used to include a file in JSP?  
   a) jsp:setProperty  
   b) jsp:getProperty  
   c) jsp:include  
   d) jsp:plugin  
   View Answer

Answer: c  
Explanation: jsp:include action variable is used to include a file in JSP.

1. Which attribute uniquely identification element?  
   a) ID  
   b) Class  
   c) Name  
   d) Scope  
   View Answer

Answer: a  
Explanation: ID attribute is used to uniquely identify action element.

1. “out” is implicit object of which class?  
   a) javax.servlet.jsp.PrintWriter  
   b) javax.servlet.jsp.SessionWriter  
   c) javax.servlet.jsp.SessionPrinter  
   d) javax.servlet.jsp.JspWriter  
   View Answer

Answer: d  
Explanation: JspWriter object is referenced by the implicit variable out which is initialized automatically using methods in the PageContext object.

1. Which object stores references to the request and response objects?  
   a) sessionContext  
   b) pageContext  
   c) HttpSession  
   d) sessionAttribute  
   View Answer

Answer: b  
Explanation: pageContext object contains information about directives issued to JSP page.

1. What temporarily redirects response to the browser?  
   a) <jsp:forward>  
   b) <%@directive%>  
   c) response.sendRedirect(URL)  
   d) response.setRedirect(URL)  
   View Answer

Answer: c  
Explanation: response.sendRedirect(URL) directs response to the browser and creates a new request.

1. Which tag is used to set a value of a JavaBean?  
   a) <c:set>  
   b) <c:param>  
   c) <c:choose>  
   d) <c:forward>  
   View Answer

Answer: a  
Explanation: <c:set> is used to set a value of a java.util.Map object.

1. Can <!–comment–> and <%–comment–%> be used alternatively in JSP?  
   a) True  
   b) False  
   View Answer

Answer: b  
Explanation: <!–comment–> is an HTML comment. <%–comment–%> is JSP comment.

1. Java code is embedded under which tag in JSP?  
   a) Declaration  
   b) Scriptlet  
   c) Expression  
   d) Comment  
   View Answer

Answer: b  
Explanation: Scriptlet is used to embed java code in JSP.

1. Which of the following is not a directive in JSP?  
   a) page directive  
   b) include directive  
   c) taglib directive  
   d) command directive  
   View Answer

Answer: d  
Explanation: command directive is not a directive in JSP.

1. Which of the following method can be used to read a multiple values with same name, for example check box selections?

**A** - request.getParameter()

**B** - response.getParameter()

**C** - request.getParameterValues()

[**D** - response.getParameterValues()](javascript:void(0);)

### Answer : C

### Explaination

You call request.getParameterValues() method to get the value of a form parameter if the parameter appears more than once and returns multiple values.

1. What happens when buffer is set to a value "none"?

**A** - Servlet output is immediately directed to the response output object.

[**B** - Compilation error.](javascript:void(0);)

### Answer : A

### Explaination

When buffer is set to "none", servlet output is immediately directed to the response output object.

**Spring**

## What is Aspect Oriented Programming (AOP)?

- Basically Aspect oriented programming complements object oriented programming by providing another way of programming model structure.  
- In addition to classes, AOP gives you aspect, which enables modularization of concerns such as Transaction management or logging and can be separated out from the application business logic of the code (these kinds of concerns are termed as crosscutting concerns). AOP supports separating application business logic from System services.

## What is IOC or Dependency Injection?

- The basic concept of IOC (Dependency of Injection) is that you do not create your objects but describe how they should be created.  
- You don’t directly connect your component and services together in code but describe which services are needed by which component in configuration file.  
- You just need to describe the dependency, the Spring container is then responsible for hooking it all up.

## What is Bean Factory in Spring?

- A Bean Factory is like a factory class that contains collections of beans. The Bean Factory holds bean definition of multiple beans within itself and then instantiates the bean when asked by client.  
- Bean Factory is actual representation of the Spring IOC container that is responsible for containing and managing the configured beans.

## 4. Different Spring Bean Scope.

**1. singleton :** Return a single bean instance per Spring IOC container.  
**2. prototype :** Return a new bean instance each time when requested.  
**3. request :** Return a single bean instance per HTTP request.  
**4. session :** Return a single bean instance per HTTP session.  
**5. global session :** Return a single bean instance per global HTTP session and only valid when used in portlet context.

## 5. How you will decide when to use prototype scope and when singleton scope bean?

- You should use the prototype scope for all beans that are stateful and the singleton scope should be used for stateless beans.

## 6. What are the different types of IOC?

## Constructor, setter, name

## 7. Differentiate between BeanFactory and ApplicationContext in spring.

- With ApplicationContext more than one config files are possible while only one config file or .xml file is possible with BeanFactory.   
- ApplicationContext publishes events to beans that are registered as listeners while BeanFactory doesn't support this  
- ApplicationContext support internationalization messages, application life-cycle events, validation and many enterprise services like JNDI access, EJB integration, remoting etc. while BeanFactory doesn't support any of these.

## 8. What is the difference between singleton and prototype bean?

Mainly it is the scope of a beans which defines their existence on the application  
**Singleton :** It means single bean definition to a single object instance per Spring IOC container.  
**Prototype :** It means a single bean definition to any number of object instances.

## 9. How do beans become 'singleton' or prototype?

- Using annotation on bean

## 10. What type of transaction Management Spring support?

Spring supports two types of transaction management:   
  
1. Programmatic transaction management  
2. Declarative transaction management.

## 11. When do you use programmatic and declarative transaction management?

- Programmatic transaction management is used preferably when you have a small number of transactional operations.   
- Incase of large number of transactional operations it is better to use declarative transaction management.

## 12. What is Bean Wiring?

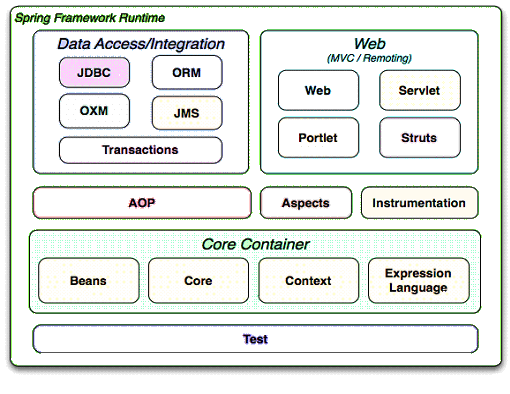
- Bean wiring means creating associations between application components i.e. beans within the spring container.

## 13. What are the various transaction manager implementations in Spring?

1. DataSourceTransactionManager : PlatformTransactionManager implementation for single JDBC data sources.   
2. HibernateTransactionManager: PlatformTransactionManager implementation for single Hibernate session factories.   
3. JdoTransactionManager : PlatformTransactionManager implementation for single JDO persistence manager factories.   
4. JtaTransactionManager : PlatformTransactionManager implementation for JTA, i.e. J2EE container transactions.

## 14. What are the different modules in spring framework?

The Spring features or organized into about 20 modules. These modules are grouped into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation and Test, as depicted below.



## 15. What is Auto Wiring in Spring?

- The Auto-wiring in spring framework can be performed by configuring in xml and Spring Auto-Wiring with Annotation @Autowired.

Byname, bytype, constructor, auto detect.

## 16. What is JdbcTemplate in Spring? And how to use it?

The JdbcTemplate class is the main class of the JDBC Core package. The JdbcTemplate (The class internally use JDBC API) helps to eliminate lot of code you write with simple JDBC API (Creating connection, closing connection, releasing resources, handling JDB Exceptions, handle transaction etc.). The JdbcTemplate handles the creation and release of resources, which helps you to avoid common error like forgetting to close connection.

## 17. What NamedParameterJdbcTemplate in Spring?

The NamedParameterJdbcTemplate allow basic set of JDBC operations, it allows named parameter in the query instead of traditional (?) placeholder, the functionality is similar to JdbcTemplate class.

## 18. What are Advice, Aspect, Join-point and point cut in spring?

**Advice :**   
An advice is an action taken by the aspect at particular join-point is called Advice.  
  
**Aspect :**   
An aspect is a subprogram which is associated with specific property of a program (Example separating logging code from the main program). An aspect is functionality or feature that cross cuts over object. AOP increase modularity of a program.  
  
**Join-Point :**   
A join point is a point used in spring AOP framework to represent a method execution. It always point during execution of program, method or exception. A join point is basically an opportunity within the code to apply aspect.  
  
**Point Cut :**   
In AOP a point cut is a set of many join points where an advice can execute. A chunk of code (known as Advice) associated with join point get executed.

## 19. What are the different types of Advice?

There are different types of Advice.  
  
**Before Advice :**   
The advice which executed before a join point called before advice. The before advice does not have the ability to prevent the execution flow proceeding at the join point (unless it throws an exception).  
  
**After Return Advice :**   
The advice which executed after a join point completed normally without any exception.  
  
**Around Advice :**   
It is responsible for choosing whether to proceeds to the join point or shortcut the advised method execution by returning its own return value or throwing an exception. This is most powerful kind of advice. With Around advice you can perform custom behavior before and after method execution.  
  
**After Throwing Advice :**   
The advice executed when a method throws an exception.  
  
**After (finally) Advice :**  
The advice is executed when program exits the join points either normally or by throwing an exception.

## 20. What is front controller in Spring MVC?

The Front Controller is basically a type of Design pattern which are being implemented in different framework (e.g. Struts and Spring MVC etc.).   
- In Spring MVC DispatcherServlet act as a Front Controller for the framework and responsible for intercepting every request and then dispatches/forwards request to appropriate controller. Configure the DispatcherServlet in the web.xml file of web application and request which we want to be handled by DispatcherServlet should be mapped using URL mapping.

## 21. Difference between FileSystemResource and ClassPathResource.

- In FileSystemResource you need to give the configuration file (i.e. spring-config.xml) relative to your project or the absolute location of the file.  
  
- In ClassPathResource spring looks for the file in the ClassPath so configuration (i.e. spring-config.xml) file should be included in the classpath. If spring-config.xml is in classpath, you can simply give the name of the file.  
  
- For Example: If your configuration file is at src/main/java/com/test/loadresource then your FileSystemResource would be:

FileSystemResource resource = new FileSystemResource("src/main/java/com/test/loadresource/spring-config.xml");

And ClassPathResource would be :

ClassPathResource resource = new ClassPathResource("com/test/loadresource /spring-config.xml");

## 22. What is inner Bean Definition?

- A bean definition added inside the property or constructor-arg elements are called inner bean.

## 23. What are the annotations supported by the spring framework?

Due to addition of some core features from the JavaConfig project to the Spring Framework,the following annotations are now directly supported :  
- @Configuration  
- @Bean  
- @DependsOn  
- @Primary  
- @Lazy  
- @Import  
- @ImportResource  
- @Value

24. How to achieve internationalization -> using MessageSource.

## 25. What is DataAccessException?

- DataAccessException is an unchecked RuntimeException. These type of exceptions are unforced by users to handle. This exception is used to handle the errors occurring when the details of the database access API in use, such as JDBC.

26. Does Spring singleton beans are thread-safe?

No, Spring singleton beans are not thread-safe. Singleton doesn't mean bean would be thread-safe.

27. **What is View Resolver pattern? how it works in Spring MVC**  
View Resolver pattern is a J2EE pattern which allows a web application to dynamically choose it's view technology e.g. HTML, JSP, Tapestry, JSF, XSLT or any other view technology.

28. **How do you create a controller in Spring? @Controller vs @RestController?**

**29. Can we use more than one configuration file for our Spring project?**  
Yes, you can use as many as you want, all you need to is import them in the main Spring configuration file which you will load from your program.

1. **What is the difference between ApplicationContext and BeanFactory in Spring framework?**

Before seeing difference between ApplicationContext and BeanFactory, let see some similarity between both of them. Spring provides two kinds of IOC container, one is BeanFactory and other is ApplicationContext. Syntactically BeanFactory and ApplicationContext both are Java interfaces and ApplicationContext extends BeanFactory. Both of them are configuration using XML configuration file. In short BeanFactory provides basic IOC and DI features while ApplicationContext provides advanced features. Apart from these, Here are few more difference between BeanFactory and ApplicationContext which is mostly based upon features supported by them.

1) BeanFactory doesn't provide support for internationalization i.e. i18n but ApplicationContext provides support for it.

2) Another difference between BeanFactory vs ApplicationContext is ability to publish event to beans that are registered as listener.

1. **What is the use of DispatcherServlet in Spring MVC?**
2. **What is the difference between @Autowired and @Inject annotation in Spring?**

The @Inject annotation also serves the same purpose, but the main difference between them is that @Inject is a **standard annotation** for dependency injection and @Autowired is **spring specific**.

1. **Difference between @RequestParam and @PathVariable in Spring MVC?**
2. **Difference between @Component, @Service, @Controller, and @Repository annotations in Spring MVC?**
3. **Is the DispatcherServlet instantiated via an application context?**  
   No, DispatcherServlet is instantiated by Servlet containers like Tomcat or Jetty. You must define DispatcherServlet into the web.xml file as shown below.  
     
   You can see that load-on-startup tag is 1 which means DispatcherServlet is instantiated when you deploy Spring MVC application to Tomcat or any other Servlet container. During instantiation, it looks for a file servlet-name-context.xml and then initializes beans defined in this file.
4. **What is the @Controller annotation used for? How can you create a controller without an annotation?**  
   The @Controller is a Spring MVC annotation to define Controller but in reality, it's just a stereotype annotation. You can even create a controller without @Controller by annotating the Spring MVC Controller classes using @Component annotation. The real job of request mapping to the handler method is done using @RequestMapping annotation.
5. **What is the ContextLoaderListener and what does it do?**   
   The ContextLoaderListener is a listener which helps to bootstrap Spring MVC. As the name suggests it loads and create ApplicationContext, so you don't have to write explicit code to do create it.  
     
   The application context is where Spring bean leaves. For a Web application, there is is a subclass called WebAppliationContext.  
     
   The ContextLoaderListener also tie the lifecycle of the ApplicationContext to the lifecycle of the ServletContext. You can get the ServletContext from WebApplicationContext using getServletContext() method.
6. What are response status?

201, 200, 401, 500, etc

1. **How can we use Spring to create Rest Web Service returning JSON response?**

For adding **JSON** support to your spring application, you will need to add **Jackson** dependency in first step.

|  |
| --- |
| <!-- Jackson JSON Processor -->  <dependency>      <groupId>com.fasterxml.jackson.core</groupId>      <artifactId>jackson-databind</artifactId>      <version>2.4.1</version>  </dependency> |

Now you are ready to return JSON response from your MVC controller. All you have to do is return JAXB annotated object from method and use @ResponseBody annotation on this return type.

|  |
| --- |
| @Controller  public class EmployeeRESTController  {      @RequestMapping(value = "/employees")      public @ResponseBody EmployeeListVO getAllEmployees()      {          EmployeeListVO employees = new EmployeeListVO();          //Add employees          return employees;      }  } |

Alternatively, you can use @RestController annotation in place of @Controller annotation. This will remove the need to using @ResponseBody.

**@RestController = @Controller + @ResponseBody**

So you can write the above controller as below.

|  |
| --- |
| @RestController  public class EmployeeRESTController  {      @RequestMapping(value = "/employees")      public EmployeeListVO getAllEmployees()      {          EmployeeListVO employees = new EmployeeListVO();          //Add employees          return employees;      }  } |

## 40. What is a MultipartResolver and when its used?

Spring comes with **MultipartResolver** to handle **file upload** in web application. There are two concrete implementations included in Spring:

1. **CommonsMultipartResolver** for Jakarta Commons FileUpload
2. **StandardServletMultipartResolver** for Servlet 3.0 Part API

## 41. How does Spring MVC provide validation support?

Spring supports validations primarily into two ways.

1. Using **JSR-303 Annotations** and any reference implementation e.g. **Hibernate Validator**
2. Using custom implementation of **org.springframework.validation.Validator** interface

42. **What are the different types of events of Listeners?**

Following are the different types of events of listeners:

* ContextClosedEvent – This event is called when the context is closed.
* ContextRefreshedEvent – This event is called when context is initialized or refreshed
* RequestHandledEvent – This event is called when the web context handles request
  1. **What are different types of Autowire?**

There are four different types of Auto wire:

* byName
* byType
* constructor
* autodetect

44. **In what points, can weaving be applied?**

Following are the points where weaving can be applied:

* Compile Time
* Class load Time
* Runtime

45. **What are the various editors used in spring work?**

The various custom editors provided by the Spring Framework are:

* PropertyEditor
* URLEditor
* ClassEditor
* CustomDateEditor
* FileEditor
* LocaleEditor
* StringArrayPropertyEditor
* StringTrimmerEditor

46. **How is Hibernate accessed using the Spring framework?**

Hibernate can be accessed in the following two ways:

* By IOC with a Callback and HibernateTemplate.
* By applying an AOP Interceptor and broadening the HibernateDaoSupport.

47. Which method is used to gracefully shutdown all the bean processes after closing the spring container?

* [**A.**](javascript:%20void(0)) shutdownHook
* [**B.**](javascript:%20void(0)) destory method
* [**C.**](javascript:%20void(0)) none of the mentioned
* [**D.**](javascript:%20void(0)) all of the mentioned

### Answer & Explanation

Answer: Option A

Explanation:

ShutdownHook method gracefully shuts down each bean process before closing the container.

48. Which attribute is used to specify classname of the bean?

* [**A.**](javascript:%20void(0)) name
* [**B.**](javascript:%20void(0)) id
* [**C.**](javascript:%20void(0)) class
* [**D.**](javascript:%20void(0)) constructor-args

### Answer & Explanation

Answer: Option C

Explanation:

Class attribute is mandatory and denotes the class used to create bean.

49. Declaring bean form object properties can be done using:-

* [**A.**](javascript:%20void(0)) PropertyPathFactoryBean
* [**B.**](javascript:%20void(0)) util:constant
* [**C.**](javascript:%20void(0)) None of the mentioned
* [**D.**](javascript:%20void(0)) All of the mentioned

### Answer & Explanation

Answer: Option A

Explanation:

To declare a bean from an object property or a property path, you can make use of either the built-in factory bean PropertyPathFactoryBean or the util:property-path tag in Spring 2.x.

50. Method used to process bean before initialization callback

* [**A.**](javascript:%20void(0)) scope
* [**B.**](javascript:%20void(0)) postProcessAfterInitialization()
* [**C.**](javascript:%20void(0)) postProcessBeforeInitialization()
* [**D.**](javascript:%20void(0)) it’s own constructor
* Answer: Option C
* Explanation:
* You can process every bean before initialization callback method by implementing the postProcessBeforeInitialization() and methods.

51. Which attribute is used to set the scope of the bean?

* [**A.**](javascript:%20void(0)) setScope
* [**B.**](javascript:%20void(0)) scope
* [**C.**](javascript:%20void(0)) getScope
* [**D.**](javascript:%20void(0)) none of the mentioned

### Answer & Explanation

Answer: Option B

Explanation:

Scope attribute defines the scope of a bean.

52. Which of the following is advice supported by Aspect Annotation?

* [**A.**](javascript:%20void(0)) @Before
* [**B.**](javascript:%20void(0)) @After
* [**C.**](javascript:%20void(0)) @AfterReturning
* [**D.**](javascript:%20void(0)) All of the mentioned

### Answer & Explanation

Answer: Option D

Explanation:

AspectJ supports five types of advice annotations: @Before, @After, @AfterReturning, @AfterThrowing, and @Around.

53. Which advice is executed only when joint point returns or throws an exception?

* [**A.**](javascript:%20void(0)) @Before
* [**B.**](javascript:%20void(0)) @After
* [**C.**](javascript:%20void(0)) @AfterReturning
* [**D.**](javascript:%20void(0)) @AfterThrowing

### Answer & Explanation

Answer: Option C

Explanation:

If you would like to perform logging only when a join point returns, you should replace the after advice with an after returning advice.

54. Special compiler used during weaving

* [**A.**](javascript:%20void(0)) jvm
* [**B.**](javascript:%20void(0)) gcc
* [**C.**](javascript:%20void(0)) ajc
* [**D.**](javascript:%20void(0)) none of the mentioned

### Answer & Explanation

Answer: Option C

Explanation:

AspectJ compile-time weaving is done through a special AspectJ compiler called ajc.

55. How to inject Spring bean into domain objcts

* [**A.**](javascript:%20void(0)) AOP
* [**B.**](javascript:%20void(0)) XML
* [**C.**](javascript:%20void(0)) AspectJ
* [**D.**](javascript:%20void(0)) Java Based

### Answer & Explanation

Answer: Option A

Explanation:

To inject a Spring bean into domain objects created outside Spring, you need the help of AOP.

56. Annotation used to declare an introduction

* [**A.**](javascript:%20void(0)) Before
* [**B.**](javascript:%20void(0)) After
* [**C.**](javascript:%20void(0)) @DeclareParents
* [**D.**](javascript:%20void(0)) None of the mentioned

### Answer & Explanation

Answer: Option C

Explanation:

In this aspect, you can declare an introduction by annotating an arbitrary field with the @DeclareParents annotation.

57.  Is the following pointcut expression correct?

execution(\* ArithmeticCalculator.\*(..))

* [**A.**](javascript:%20void(0)) Yes
* [**B.**](javascript:%20void(0)) No
* [**C.**](javascript:%20void(0)) If every target class is in same package
* [**D.**](javascript:%20void(0)) Depends where target class is located

### Answer & Explanation

Answer: Option C

Explanation:

You can omit the package name if the target class or interface is located in the same package as this aspect.

58. Validator field is not assigned to any bean, but rather a factory class of the type.

* [**A.**](javascript:%20void(0)) javax.validation.Validator
* [**B.**](javascript:%20void(0)) javax.validation.ValidatorFactory
* [**C.**](javascript:%20void(0)) javax.validation.ValidatorFactor
* [**D.**](javascript:%20void(0)) none of the mentioned

### Answer & Explanation

Answer: Option B

Explanation:

This is how JSR-303 validation works. The assignment process is done inside the controller’s constructor.

59. Attribute used to handle web flow requests.

* [**A.**](javascript:%20void(0)) servlet-mapping
* [**B.**](javascript:%20void(0)) servlet-attr
* [**C.**](javascript:%20void(0)) servlet-flow
* [**D.**](javascript:%20void(0)) servlet-requests

### Answer & Explanation

Answer: Option A

Explanation:

If you are going to use the legacy JSF request handling at the same time, you will have to specify that.

60. To access JSON in a Spring application, library used is:-

* [**A.**](javascript:%20void(0)) JSON-LIB
* [**B.**](javascript:%20void(0)) SpringJSON-lib
* [**C.**](javascript:%20void(0)) JSON-lib
* [**D.**](javascript:%20void(0)) None of the mentioned

### Answer & Explanation

Answer: Option A

Explanation:

Though it’s technically possible to access JSON in a Spring application (i.e., on the server side), using a third-party Java library like JSON-LIB

61. Handler method’s annotation.

* [**A.**](javascript:%20void(0)) @Before
* [**B.**](javascript:%20void(0)) @Controller
* [**C.**](javascript:%20void(0)) @After
* [**D.**](javascript:%20void(0)) @RequestMapping

### Answer & Explanation

Answer: Option D

Explanation:

In order to do so, a controller class’s methods are decorated with the @RequestMapping annotation, making them handler methods.

62. Alternative way to resolve locales.

* [**A.**](javascript:%20void(0)) AcceptHeaderLocale
* [**B.**](javascript:%20void(0)) AcceptHeader
* [**C.**](javascript:%20void(0)) AcceptHeaderLocaleResolver
* [**D.**](javascript:%20void(0)) SessionLocaleResolver

### Answer & Explanation

Answer: Option D

Explanation:

Another option of resolving locales is by SessionLocaleResolver.

63. View beans from a resource bundle is loaded by.

* [**A.**](javascript:%20void(0)) ResourceBundleViewResolver
* [**B.**](javascript:%20void(0)) ResourceBundleView
* [**C.**](javascript:%20void(0)) ResourceViewResolver
* [**D.**](javascript:%20void(0)) None of the mentioned

### Answer & Explanation

Answer: Option A

Explanation:

ResourceBundleViewResolver loads view beans from a resource bundle in the classpath root.

64. To configure the exception mappings in the web application context.

* [**A.**](javascript:%20void(0)) SimpleMappingExceptionResolver
* [**B.**](javascript:%20void(0)) ExceptionResolver
* [**C.**](javascript:%20void(0)) ExceptionViewResolver
* [**D.**](javascript:%20void(0)) None of the mentioned

### Answer & Explanation

Answer: Option A

Explanation:

Spring MVC comes with the exception resolver SimpleMappingExceptionResolver for you to configure the exception mappings in the web application context.

65. Proxying Spring services with EJB3s, injecting custom resources configured in Spring, or even using Spring to isolate your EJBs from acquiring references to other distributed resources such as a REST endpoint or an RMI endpoint.

* [**A.**](javascript:%20void(0)) SpringBeanAutowiringInterceptor
* [**B.**](javascript:%20void(0)) SpringBeanAutowiring
* [**C.**](javascript:%20void(0)) SpringBean
* [**D.**](javascript:%20void(0)) SpringBeanInterceptor

### Answer & Explanation

Answer: Option A

66. DAO methods require access to the session factory, which can be injected:-

* [**A.**](javascript:%20void(0)) a setter method
* [**B.**](javascript:%20void(0)) constructor argument
* [**C.**](javascript:%20void(0)) none of the mentioned
* [**D.**](javascript:%20void(0)) all of the mentioned

### Answer & Explanation

Answer: Option D

Explanation:

To use the contextual session approach, your DAO methods require access to the session factory, which can be injected via a setter method or a constructor argument.

67. PreparedStatementSetter, as its name indicates, create a PreparedStatement object on this connectionthe parameter as well as binding task of the overall update process.

* [**A.**](javascript:%20void(0)) True
* [**B.**](javascript:%20void(0)) False
* [**C.**](javascript:%20void(0)) Error
* [**D.**](javascript:%20void(0)) None of them

### Answer & Explanation

Answer: Option B

Explanation:

The second callback interface, PreparedStatementSetter, as its name indicates, performs only the parameter binding task of the overall update process.

68. Interface whose instances can be obtained from a SessionFactory instance.

* [**A.**](javascript:%20void(0)) Session
* [**B.**](javascript:%20void(0)) Session Factory
* [**C.**](javascript:%20void(0)) All of the mentioned
* [**D.**](javascript:%20void(0)) None of the mentioned

### Answer & Explanation

Answer: Option A

Explanation:

In Hibernate, the core interface for object persistence is Session, whose instances can be obtained from a SessionFactory instance.

69. SimpleJdbcTemplate offers a convenient batch update method in the form of:-

* [**A.**](javascript:%20void(0)) Vector
* [**B.**](javascript:%20void(0)) Set
* [**C.**](javascript:%20void(0)) Map
* [**D.**](javascript:%20void(0)) List

### Answer & Explanation

Answer: Option D

Explanation:

SimpleJdbcTemplate offers a convenient batch update method for you to specify a SQL statement and a batch of parameters in the form of List so that you don’t need to implement the BatchPreparedStatementSetter interface.

**Design Pattern**

Name types of Design Patterns?

Design patterns can be classified in three categories: Creational, Structural and Behavioral patterns.

* **Creational Patterns** - These design patterns provide a way to create objects while hiding the creation logic, rather than instantiating objects directly using new opreator. This gives program more flexibility in deciding which objects need to be created for a given use case.
* **Structural Patterns** - These design patterns concern class and object composition. Concept of inheritance is used to compose interfaces and define ways to compose objects to obtain new functionalities.
* **Behavioral Patterns** - These design patterns are specifically concerned with communication between objects.

**What is Factory pattern?**

Factory pattern is one of most used design pattern in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.

**What is Abstract Factory pattern?**

Abstract Factory patterns work around a super-factory which creates other factories. This factory is also called as factory of factories. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

In Abstract Factory pattern an interface is responsible for creating a factory of related objects without explicitly specifying their classes. Each generated factory can give the objects as per the Factory pattern.

**What is Singleton pattern?**

Singleton pattern is one of the simplest design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

What is the benefit of Factory pattern?

Factory pattern encapsulates the implementation details and underlying implementation can be changed without any impact on caller api.

What is Builder pattern?

Builder pattern builds a complex object using simple objects and using a step by step approach. This builder is independent of other objects.

What is Prototype pattern?

Prototype pattern refers to creating duplicate object while keeping performance in mind. This pattern involves implementing a prototype interface which tells to create a clone of the current object.

When Prototype pattern is to be used?

This pattern is used when creation of object directly is costly. For example, an object is to be created after a costly database operation. We can cache the object, returns its clone on next request and update the database as and when needed thus reducing database calls.

What is Adapter pattern?

Adapter pattern works as a bridge between two incompatible interfaces. This pattern involves a single class which is responsible to join functionalities of independent or incompatible interfaces.

Give an example of Adapter pattern.

A real life example could be a case of card reader which acts as an adapter between memory card and a laptop. You plugin the memory card into card reader and card reader into the laptop so that memory card can be read via laptop.

What is Bridge pattern?

Bridge is used when we need to decouple an abstraction from its implementation so that the two can vary independently. This type of design pattern comes under structural pattern as this pattern decouples implementation class and abstract class by providing a bridge structure between them.

This pattern involves an interface which acts as a bridge which makes the functionality of concrete classes independent from interface implementer classes. Both types of classes can be altered structurally without affecting each other.

What is Filter pattern?

Filter pattern or Criteria pattern is a design pattern that enables developers to filter a set of objects using different criteria and chaining them in a decoupled way through logical operations. This type of design pattern comes under structural pattern as this pattern combines multiple criteria to obtain single criteria.

What is Composite pattern?

Composite pattern is used where we need to treat a group of objects in similar way as a single object. Composite pattern composes objects in term of a tree structure to represent part as well as whole hierarchy. This type of design pattern comes under structural pattern as this pattern creates a tree structure of group of objects.

This pattern creates a class that contains group of its own objects. This class provides ways to modify its group of same objects.

What is Decorator pattern?

Decorator pattern allows a user to add new functionality to an existing object without altering its structure. This type of design pattern comes under structural pattern as this pattern acts as a wrapper to existing class.

This pattern creates a decorator class which wraps the original class and provides additional functionality keeping class methods signature intact.

What is Facade pattern?

Facade pattern hides the complexities of the system and provides an interface to the client using which the client can access the system. This type of design pattern comes under structural pattern as this pattern adds an interface to existing system to hide its complexities.

This pattern involves a single class which provides simplified methods required by client and delegates calls to methods of existing system classes.

What is Flyweight pattern?

Flyweight pattern is primarily used to reduce the number of objects created and to decrease memory footprint and increase performance. This type of design pattern comes under structural pattern as this pattern provides ways to decrease object count thus improving the object structure of application.

Flyweight pattern tries to reuse already existing similar kind objects by storing them and creates new object when no matching object is found.

What is Proxy pattern?

In proxy pattern, a class represents functionality of another class. This type of design pattern comes under structural pattern.

In proxy pattern, we create object having original object to interface its functionality to outer world.

What is Chain of Responsibility pattern?

As the name suggests, the chain of responsibility pattern creates a chain of receiver objects for a request. This pattern decouples sender and receiver of a request based on type of request. This pattern comes under behavioral patterns.

In this pattern, normally each receiver contains reference to another receiver. If one object cannot handle the request then it passes the same to the next receiver and so on.

What is Command pattern?

Command pattern is a data driven design pattern and falls under behavioral pattern category. A request is wrapped under an object as command and passed to invoker object. Invoker object looks for the appropriate object which can handle this command and passes the command to the corresponding object which executes the command.

What is Interpreter pattern?

Interpreter pattern provides a way to evaluate language grammar or expression. This type of pattern comes under behavioral pattern. This pattern involves implementing an expression interface which tells to interpret a particular context.

Give an example where Interpreter pattern is used?

This pattern is used in SQL parsing, symbol processing engine etc.

What is Iterator pattern?

Iterator pattern is very commonly used design pattern in Java and .Net programming environment. This pattern is used to get a way to access the elements of a collection object in sequential manner without any need to know its underlying representation. Iterator pattern falls under behavioral pattern category.

What are the entities of Service Locator pattern?

Following are the entities of this type of design pattern.

* **Service** - Actual Service which will process the request. Reference of such service is to be looked upon in JNDI server.
* **Context / Initial Context** - JNDI Context carries the reference to service used for lookup purpose.
* **Service Locator** - Service Locator is a single point of contact to get services by JNDI lookup caching the services.
* **Cache** - Cache to store references of services to reuse them.
* **Client** - Client is the object that invokes the services via ServiceLocator.

What is Mediator pattern?

Mediator pattern is used to reduce communication complexity between multiple objects or classes. This pattern provides a mediator class which normally handles all the communications between different classes and supports easy maintenance of the code by loose coupling. Mediator pattern falls under behavioral pattern category.

What is Memento pattern?

Memento pattern is used to restore state of an object to a previous state. Memento pattern falls under behavioral pattern category.

Name the actor classes used in Memento pattern.

Memento pattern uses three actor classes. Memento contains state of an object to be restored. Originator creates and stores states in Memento objects and Caretaker object is responsible to restore object state from Memento.

What is Observer pattern?

Observer pattern is used when there is one-to-many relationship between objects such as if one object is modified, its depenedent objects are to be notified automatically. Observer pattern falls under behavioral pattern category.

Name the actor classes used in Observer pattern.

Observer pattern uses three actor classes. Subject, Observer and Client. Subject is an object having methods to attach and detach observers to a client object. We have created an abstract class Observer and a concrete class Subject that is extending class Observer.

What is state pattern?

In State pattern a class behavior changes based on its state. This type of design pattern comes under behavior pattern. In State pattern, we create objects which represent various states and a context object whose behavior varies as its state object changes.

What is Null Object pattern?

In Null Object pattern, a null object replaces check of NULL object instance. Instead of putting if check for a null value, Null Object reflects a do nothing relationship. Such Null object can also be used to provide default behaviour in case data is not available.

In Null Object pattern, we create an abstract class specifying various operations to be done, concrete classes extending this class and a null object class providing do nothing implemention of this class and will be used seemlessly where we need to check null value.

What is Strategy pattern?

In Strategy pattern, a class behavior or its algorithm can be changed at run time. This type of design pattern comes under behavior pattern.

In Strategy pattern, we create objects which represent various strategies and a context object whose behavior varies as per its strategy object. The strategy object changes the executing algorithm of the context object.

What is Template pattern?

In Template pattern, an abstract class exposes defined way(s)/template(s) to execute its methods. Its subclasses can override the method implementation as per need but the invocation is to be in the same way as defined by an abstract class. This pattern comes under behavior pattern category.

What is Visitor pattern?

In Visitor pattern, we use a visitor class which changes the executing algorithm of an element class. By this way, execution algorithm of element can vary as and when visitor varies. This pattern comes under behavior pattern category. As per the pattern, element object has to accept the visitor object so that visitor object handles the operation on the element object.

What is MVC pattern?

MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

* **Model** - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
* **View** - View represents the visualization of the data that model contains.
* **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

What is Business Delegate pattern?

Business Delegate Pattern is used to decouple presentation tier and business tier. It is basically use to reduce communication or remote lookup functionality to business tier code in presentation tier code. In business tier we have following entities.

* **Client** - Presentation tier code may be JSP, servlet or UI java code.
* **Business Delegate** - A single entry point class for client entities to provide access to Business Service methods.
* **LookUp Service** - Lookup service object is responsible to get relative business implementation and provide business object access to business delegate object.
* **Business Service** - Business Service interface. Concrete classes implement this business service to provide actual business implementation logic.

What is Front Controller pattern?

The front controller design pattern is used to provide a centralized request handling mechanism so that all requests will be handled by a single handler. This handler can do the authentication/ authorization/ logging or tracking of request and then pass the requests to corresponding handlers. Following are the entities of this type of design pattern.

* **Front Controller** - Single handler for all kinds of requests coming to the application (either web based/ desktop based).
* **Dispatcher** - Front Controller may use a dispatcher object which can dispatch the request to corresponding specific handler.
* **View** - Views are the object for which the requests are made.

What is Intercepting Filter pattern?

The intercepting filter design pattern is used when we want to do some pre-processing / post-processing with request or response of the application. Filters are defined and applied on the request before passing the request to actual target application. Filters can do the authentication/ authorization/ logging or tracking of request and then pass the requests to corresponding handlers.

What are the entities of Intercepting Filter pattern?

Following are the entities of this type of design pattern.

* **Filter** - Filter which will performs certain task prior or after execution of request by request handler.
* **Filter Chain** - Filter Chain carries multiple filters and help to execute them in defined order on target.
* **Target** - Target object is the request handler.
* **Filter Manager** - Filter Manager manages the filters and Filter Chain.
* **Client** - Client is the object who sends request to the Target object.

What is Service Locator pattern?

The service locator design pattern is used when we want to locate various services using JNDI lookup. Considering high cost of looking up JNDI for a service, Service Locator pattern makes use of caching technique. For the first time a service is required, Service Locator looks up in JNDI and caches the service object. Further lookup or same service via Service Locator is done in its cache which improves the performance of application to great extent.

What is Transfer Object pattern?

The Transfer Object pattern is used when we want to pass data with multiple attributes in one shot from client to server. Transfer object is also known as Value Object. Transfer Object is a simple POJO class having getter/setter methods and is serializable so that it can be transferred over the network. It does not have any behavior. Server Side business class normally fetches data from the database and fills the POJO and send it to the client or pass it by value. For client, transfer object is read-only. Client can create its own transfer object and pass it to server to update values in database in one shot. Following are the entities of this type of design pattern.

* **Business Object** - Business Service fills the Transfer Object with data.
* **Transfer Object** - Simple POJO having methods to set/get attributes only.
* **Client** - Client either requests or sends the Transfer Object to Business Object.

1. Which of the below is not a valid design pattern?  
a) Singleton  
b) Factory  
c) Command  
d) Java  
View Answer

Answer: d  
Explanation: Design pattern is a general repeatable solution to a commonly occurring problem in software design. There are various patterns available for use in day to day coding problems.

2. Which of the below author is not a part of GOF (Gang of Four)?  
a) Erich Gamma  
b) Gang Pattern  
c) Richard Helm  
d) Ralph Johnson  
View Answer

Answer: b  
Explanation: Four authors named Richard Helm, Erich Gamma, Ralph Johnson and John Vlissides published a book on design patterns. This book initiated the concept of Design Pattern in Software development. They are known as Gang of Four (GOF).

3. Which of the below is not a valid classification of design pattern?  
a) Creational patterns  
b) Structural patterns  
c) Behavioural patterns  
d) Java patterns  
View Answer

Answer: d  
Explanation: Java patterns is not a valid classification of design patterns. The correct one is J2EE patterns.

4. Which design pattern provides a single class which provides simplified methods required by client and delegates call to those methods?  
a) Adapter pattern  
b) Builder pattern  
c) Facade pattern  
d) Prototype pattern  
View Answer

Answer: c  
Explanation: Facade pattern hides the complexities of the system and provides an interface to the client using which client can access the system.

5. Which design pattern ensures that only one object of particular class gets created?  
a) Singleton pattern  
b) Filter pattern  
c) State pattern  
d) Bridge pattern  
View Answer

Answer: a  
Explanation: Singleton pattern involves a single class which is responsible to create an object while making sure that only one object gets created. This class provides a way to access the only object which can be accessed directly without need to instantiate another object of the same class.

6. Which design pattern suggests multiple classes through which request is passed and multiple but only relevant classes carry out operations on the request?  
a) Singleton pattern  
b) Chain of responsibility pattern  
c) State pattern  
d) Bridge pattern  
View Answer

Answer: b  
Explanation: Chain of responsibility pattern creates a chain of receiver objects for a particular request. The sender and receiver of a request are decoupled based on the type of request. This pattern is one of the behavioral patterns.

7. Which design pattern represents a way to access all the objects in a collection?  
a) Iterator pattern  
b) Facade pattern  
c) Builder pattern  
d) Bridge pattern  
View Answer

Answer: a  
Explanation: Iterator pattern represents a way to access the elements of a collection object in sequential manner without the need to know its underlying representation.

8. What does MVC pattern stands for?  
a) Mock View Control  
b) Model view Controller  
c) Mock View Class  
d) Model View Class  
View Answer

Answer: b  
Explanation: Model represents an object or JAVA POJO carrying data.View represents the visualization of the data that model contains. The controller acts on both model and view. It is usually used in web development.

9. Is design pattern a logical concept.  
a) True  
b) False  
View Answer

Answer: a  
Explanation: Design pattern is a logical concept. Various classes and frameworks are provided to enable users to implement these design patterns.

10. Which design pattern works on data and action taken based on data provided?  
a) Command pattern  
b) Singleton pattern  
c) MVC pattern  
d) Facade pattern  
View Answer

Answer: a  
Explanation: Command pattern is a data driven design pattern. It is a behavioral pattern. A request is wrapped under an object as command and passed to the invoker object. The invoker object looks for the appropriate object which can handle this command and passes this command to the corresponding object which executes the command.

Q - Which of the following pattern is used when creation of object directly is costly?

A - Bridge Pattern

[B - Adapter Pattern](javascript:void(0);)

[C - Prototype Pattern](javascript:void(0);)

[D - Filter Pattern](javascript:void(0);)

### Answer : A

### Explanation

Prototype pattern is used when creation of object directly is costly.

Q - Can we create a clone of a singleton object?

A - true

[B - false](javascript:void(0);)

### Answer : A

### Explanation

true. It is possible to get a clone of singleton object. Throw exception within the body of clone() method to prevent cloning.

Q 4 - Which of the following describes the Interpreter pattern correctly?

[A - In this pattern a class represents functionality of another class.](javascript:void(0);)

[B - This pattern creates a chain of receiver objects for a request.](javascript:void(0);)

C - This pattern provides a way to evaluate language grammar or expression.

[D - In this pattern a request is wrapped under an object as command and passed to invoker object.](javascript:void(0);)

### Answer : C

### Explanation

Interpreter pattern provides a way to evaluate language grammar or expression. This type of pattern comes under behavioral pattern. This pattern involves implementing an expression interface which tells to interpret a particular context.

Q 5 - Which of the following describes the Iterator pattern correctly?

A - This pattern is used to get a way to access the elements of a collection object in sequential manner without any need to know its underlying representation.

[B - This pattern is used to reduce communication complexity between multiple objects or classes.](javascript:void(0);)

[C - This pattern is used to restore state of an object to a previous state.](javascript:void(0);)

[D - This pattern is used when there is one-to-many relationship between objects such as if one object is modified, its depenedent objects are to be notified automatically.](javascript:void(0);)

### Answer : A

### Explanation

This pattern is used to get a way to access the elements of a collection object in sequential manner without any need to know its underlying representation.

 Show Answer

Q 6 - Which of the following describes the Null Object pattern correctly?

[A - In this pattern, a class behavior changes based on its state.](javascript:void(0);)

B - In this pattern, a null object replaces check of NULL object instance.

[C - In this pattern, a class behavior or its algorithm can be changed at run time.](javascript:void(0);)

[D - In this pattern, an abstract class exposes defined way(s)/template(s) to execute its methods.](javascript:void(0);)

### Answer : B

### Explanation

In Null Object pattern, a null object replaces check of NULL object instance.

 Hide Answer

Q 7 - Which of the following describes the Service Locator pattern correctly?

[A - This pattern is used to separate low level data accessing API or operations from high level business services.](javascript:void(0);)

[B - This pattern is used to provide a centralized request handling mechanism so that all requests will be handled by a single handler.](javascript:void(0);)

[C - This pattern is used when we want to do some pre-processing / post-processing with request or response of the application.](javascript:void(0);)

D - This pattern is used when we want to locate various services using JNDI lookup.

### Answer : D

### Explanation

Service Locator pattern, is used when we want to locate various services using JNDI lookup.

 Hide Answer

Q 8 - Which of the following describes the Factory pattern correctly?

A - This pattern creates object without exposing the creation logic to the client and refer to newly created object using a common interface.

[B - In this pattern an interface is responsible for creating a factory of related objects without explicitly specifying their classes.](javascript:void(0);)

[C - This pattern involves a single class which is responsible to create an object while making sure that only single object gets created.](javascript:void(0);)

[D - This pattern is used when we want to pass data with multiple attributes in one shot from client to server.](javascript:void(0);)

### Answer : A

### Explanation

Factory pattern creates object without exposing the creation logic to the client and refer to newly created object using a common interface.

Q 10 - In MVC pattern, Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.

A - true

B - false

### Answer : A

### Explanation

True. In MVC pattern, Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.

**AOP**

### ****1. What is AOP or Aspect Oriented Programming in Spring framework?****

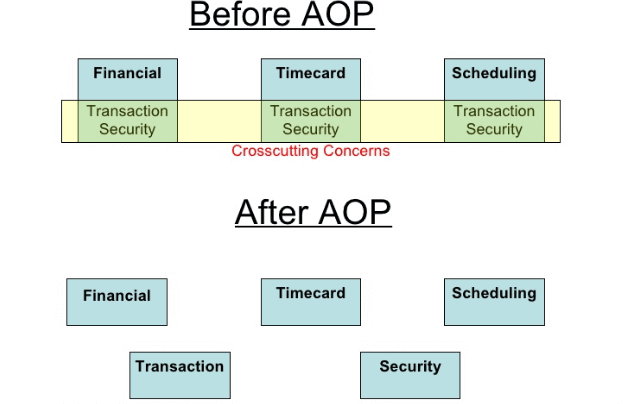
When developing an application, some program logic is common across multiple components and modules.

Most popular example for such kind of scenarios is: Logging. Maybe, we want to log the entry and exit of a method in program flow.

Other example could be: We want to authenticate every HTTP request coming to the server.

These common concerns which cut across multiple components is called **cross-cutting concern**s or an **Aspect** of an application.

Instead of applying an Aspect to each component which will create duplication, scattered piece of code to manage which has nothing to do with actual business logic, Spring provides us API to program around these Aspects.



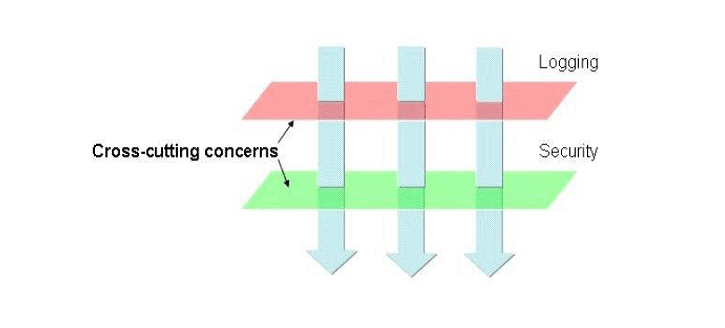
This is called **Aspect oriented programming**.

### ****2. What is the difference between concern vs cross-cutting concern in Spring AOP?****

A concern might be related to one or more components or modules.

If a concern is specific to a Single module, it is just called an application **Concern**. Typically, these are related to business domain of an application.

On that contrary, if a concern is applicable across two or more modules, it is called **cross-cutting concern**. Typical examples includes Logging, Transaction Management.



### ****3. What are some of the Spring AOP advantages?****

1. **Modularity:** It helps us to centralize the **cross-cutting concerns**.
2. **Easier Maintenance:** It is easier to make changes at one place.
3. **Pluggable:**It gets easy to add or remove **aspects**.

This is one of the must know Spring AOP interview questions.

### ****4. How do you implement cross-cutting concerns in a web application?****

We can do this with the help of Spring AOP.

Now-a-days, we can directly do it in **Spring Boot AOP**.

### ****5. When to use Spring AOP?****

There are some typical use cases where we use Spring AOP. For example:

1. Application Logging
2. Spring Security configuration
3. Transaction Management maybe.

Other than that, wherever we want to achieve some kind of cross-cutting concern, we implement that with the help of Spring AOP.

### ****6. How Spring AOP works?****

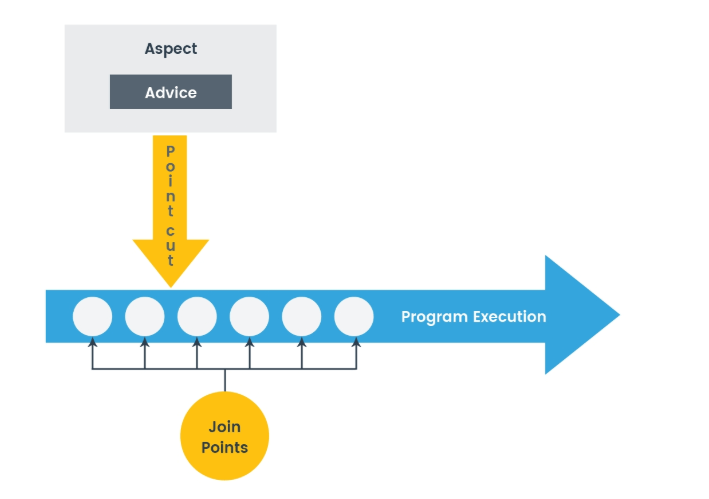
We configure Spring AOP separately from main application logic.

In this configuration, we set 2 things on broad prospective:

1. What to do?
2. and where to do?

To define these Spring provides us different concepts and components which we can use. For example: Aspect, JoinPoint, PointCut.

Once configured, it applies the defined logic in the declared locations.



**Note:** Till now we discussed the introductory Spring AOP interview questions. Now, we will start with the internals of Spring AOP concepts.

### ****7. What is Aspect in Spring AOP?****

A cross-cutting concern of an application which needs be programmed in a modular way, in separate to application’s business logic, is called an **Aspect**.

For example: Logging

### ****8. How to declare a Spring AOP Aspect?****

@Aspect annotation is used to declare an Aspect like this:

@Aspect

public class OurAspect{

}

We can also do the same with XML configuration like below:

<bean id="ourAspect" class="path.to.OurAspect">

</bean>

### ****9. Can @Aspect classes be automatically scanned with component scannning?****

No.

We need to make it annotate with @Component and then only it will be eligible for component scanning.

For example:

@Component

@Aspect

public class OurAspect{

}

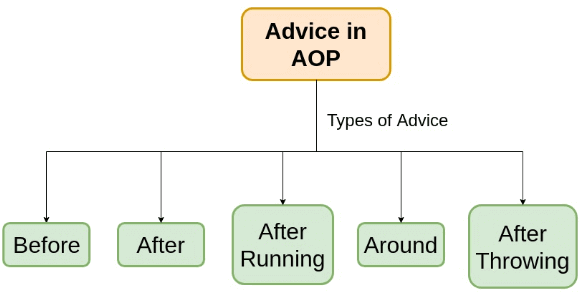
### ****10. What is Advice in Spring AOP?****

Advice is the logic we write for an Aspect.

For example: Log the method name could be an Advice for the Logging Aspect.

In terms of programming, it is achieved through writing a method for each Advice.

### ****11. What are different Spring AOP Advice types?****



Below are different Advice types supported by Spring:

1. **Before Advice:** It will run before the JoinPoint execution starts.
2. **After Returning Advice:** It will run after a JoinPoint execution is completed normally.
3. **After throwing Advice:** It will run if a JoinPoint throws Exception.
4. **After** **Advice:** It will run regardless of JoinPoint executes normally or with Exception. It is analogous to **finally** block.
5. **Spring AOP Around Advice:** It wraps the JoinPoint. It is special in the sense that it can return the result itself and choose to stop the execution to proceed to the JoinPoint.

### ****12. How to declare an Advice in Spring AOP?****

Here is an example of Before Advice:

@Component

@Aspect

public class OurLoggingAspect{

@Before("PointCutExpression")

public void log(JoinPoint joinPoint)

{

}

}

So, log method is what will get executed as advice, before the method(s), which matches PointCut expression.

### ****13. Is there any recommended way to select any one of the advice types?****

Yes. The general consideration is that:

we should pick least powerful advice type if we can accomplish our work with that.

For example:

If our task in hand can be accomplished with Before Advice or After Advice, then we need not use Around Advice.

By using the most specific advice, our program will be less prone to errors.

### ****14. What is a JoinPoint in Spring AOP?****

JoinPoint is the target location in the application where advice (method) will be applied.

In Spring AOP, a JoinPoint is always a method.

### ****15. Does Spring framework support all JoinPoints?****

A JoinPoint could be a method execution, handling of an exception or handling of an event.

Spring AOP supports **only method execution JoinPoint**.

If we want joinpoints on fields, then we can use **AspectJ**support.

### ****16. What is Pointcut in Spring AOP?****

We have **Advice** which needs to be executed and we have whole application’s classes’s methods as possible JoinPoints.

Now, where exactly the Advice will be executed? For which set of methods in the whole application?

The answer to this is: PointCut.

PointCut is a regular expression(**regex)**.

We associate a PointCut expression with each Advice. This expression will be applied to each method in the execution flow.

If it returns true, Advice will be applied for that method and if it returns false, it will not.

### ****17. What is the difference between JoinPoint and Pointcut****?

Refer [this](https://stackoverflow.com/questions/15447397/spring-aop-whats-the-difference-between-joinpoint-and-pointcut)for detailed explanation.

*This could be one of the most confusing Spring AOP interview questions sometimes.*

### ****18. Could you give an example of PointCut expression?****

Yes sure.

@Component

@Aspect

public class OurLoggingAspect{

@Before("execution(\* handleItem(..))")

public void log(JoinPoint joinPoint)

{

}

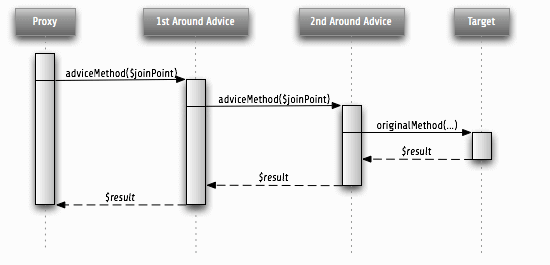
}

This PointCut expression will be true for all method named as handleItem and because of this, Advice in log method will be executed before handleItem method.

### ****19. How to implement logging using Spring AOP?****

Refer to code example in above question.

If there are multiple advices for logging, here is how it will work:



### ****20. What is “execute” in Pointcut expression?****

“execute” is called as PointCut Designator.

We can use any of the AspectJ PointCut Desidnators (PCD), and execute is just one of them.

### ****21. What are other PointCut Designators?****

We have **execution**, **within**, **target**, **args**, **this**as PointCut Designators in Spring AOP.

We have annotated variation too: **@target**, **@args**, **@within** and **@annotation**.

All these comes from AspectJ framework which we use it in Spring AOP framework.

### ****22. Can a Spring AOP pointcut have multiple expressions? Give example.****

Yes. Example being:

expression="execution(\* package1.\*.\*(..)) || execution(\* package2.\*.\*(..))"

Here we are using boolean operator to add multiple expressions in a single Spring AOP pointcut.

### ****23. Write a pointcut for all public methods.****

@Pointcut("execution(public \* \*(..))")

### ****24. Write a pointcut for all private methods.****

@Pointcut("execution(private \* \*(..))")

### ****25. What are Advice arguments?****

We can use **args()** expression, to get the JoinPoint arguments in the Advice, to customize the Advice to further level.

It will run only for JoinPoints whose arguments matches **the args()** expression.

### ****26. What is target object?****

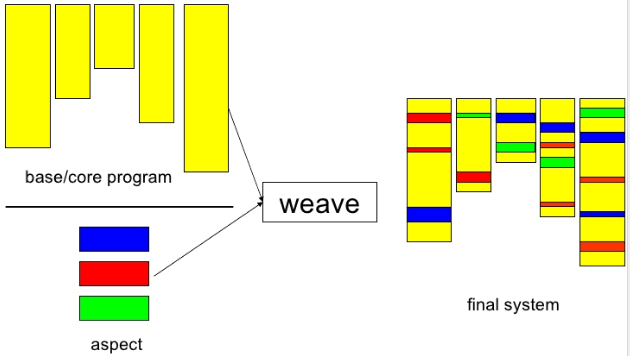
The object for which one or more Advice will be executed. It is also referred as **Advice object**.

### ****27. What is Introduction in Spring AOP?****

Introduction is a concept by which we can enhance a target object’s class in one way or another, without even touching it.

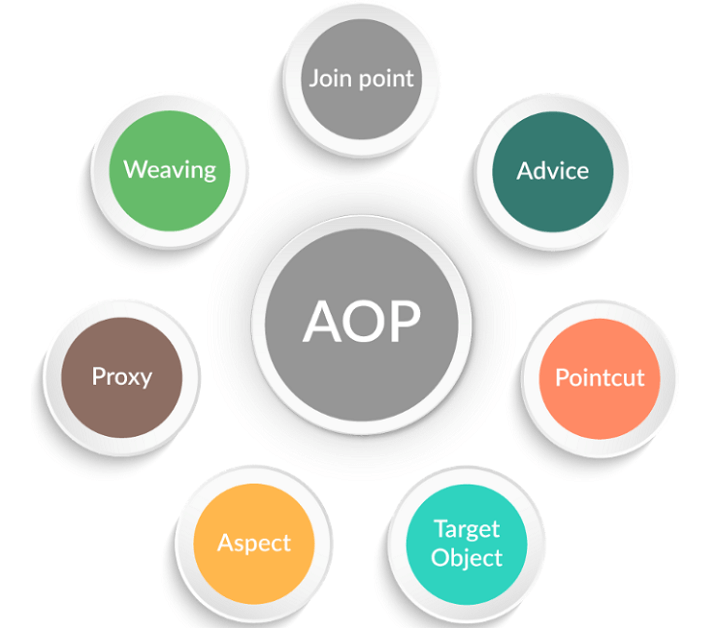
We can implement an Interface, provide it’s concrete implementation, add additional methods or attributes of the target object’s class, in the Aspect itself, in a modular fashion.

### ****28. What is Weaving?****



Weaving is a process of linking an Aspect with other application objects to create the target object.

Now, let’s revise the key Spring AOP terminologies:



**Note:** From here starts some of the trickiest of Spring AOP interview questions. So, let’s pay attention.

### ****29. When does Weaving happen in Spring AOP?****

Weaving can be done at compile time, load time or run-time.

Spring AOP does it at runtime. AspectJ does it at compile-time.

### ****30. What is Spring AOP proxy object?****

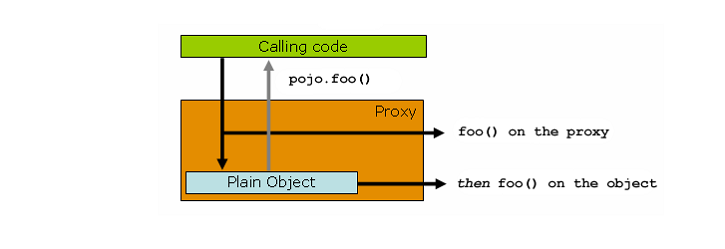
Spring implement AOP contracts with the help of proxy objects.

These are the objects created at runtime to apply the Advice.

By default, these are JDK dynamic proxies.

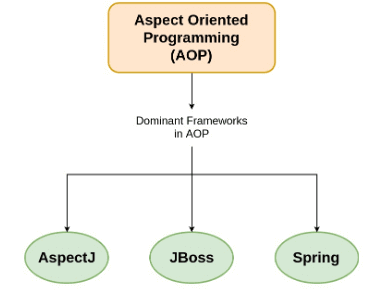
If target object does not implement any interface, Spring AOP uses CGLIB proxy.

Here is the [issue](https://stackoverflow.com/questions/14089642/original-interface-is-lost-in-spring-aop-introduction)which you might face while approaching Introduction with Spring AOP Proxy.



### ****31. What are different AOP implementations available?****

1. Spring AOP – Spring’s own implementation.
2. [AspectJ](https://www.eclipse.org/aspectj/) – now an Eclipse foundation project.
3. [JBoss AOP](https://jbossaop.jboss.org/) – maintained by JBoss community.



### ****32. Give a Spring AOP example****.

[This article](https://www.journaldev.com/2583/spring-aop-example-tutorial-aspect-advice-pointcut-joinpoint-annotations#spring-aop-example) on Journaldev has a very good summarized example on Spring AOP.

### ****33. What is the difference between Spring AOP vs AspectJ?****

Spring AOP takes care of small set of regular and typical use-cases whereas AspectJ is the most comprehensive one.

Spring AOP supports only method Pointcuts whereas AspectJ supports field level too.

Spring AOP takes care of the run-time weaving process whereas in case of AspectJ, we need to trigger the weaving process with the help of tools provided, at compile time.

### ****34. How Spring AOP works with AspectJ?****

Spring AOP uses AspectJ’s pointcut and matching library to support @AspectJ style of programming. It does not use the AspectJ compiling or weaving.

### ****35. What is the name of AspectJ pointcut and matching library?****

**aspectjweaver.jar**

This jar has to be in our classpath to support AspectJ in a Spring AOP application.

### ****36. How to enable AspectJ in a Spring AOP application?****

With XML namespace:

<aop:aspectj-autoproxy/>

With Java Configuration:

@Configuration

@EnableAspectJAutoProxy

public class CustomAOPConfig {

}

## ****Spring Boot AOP Interview Questions****

### ****37. How to enable Spring Boot AOP?****

1. Add ” spring-boot-starter-aop” starter project dependency.
2. Enable or disable AopAutoConfiguration in application.properties file. For example: spring.aop.auto=false. (It is true by default)

Our application is now ready as a Spring Boot AOP application.

### ****38. Give me a Spring Boot AOP example.****

Writing an AOP application with Spring Boot it not different than typical Spring application. We just need to enable Spring Boot as stated earlier.

Although DZone and javatpoint has examples for this but checkout [this example](https://www.javainuse.com/spring/spring-boot-aop) which uses starter project.

### ****39. How to write Spring AOP custom annotation? Give example.****

Firstly, create a custom annotation:

@Target(ElementType.METHOD)

@Retention(RetentionPolicy.RUNTIME)

public @interface CustomAnnotation{

}

The method for which we want to apply the aspect, annotate it with CustomAnnotation.

@CustomAnnotation

public void aMethod() {

//some logic

}

Now create a pointcut like this:

@Before("@annotation(CustomAnnotation)")

### ****40. Write a Spring AspectJ example for audit logging.****

Checkout this [article](https://medium.com/@wkrzywiec/moving-into-next-level-in-user-log-events-with-spring-aop-3b4435892f16) which does audit logging using AspectJ with Spring AOP. It uses Gradle as build system, not Maven.

**MCQs**

1. <http://learnerindia.com/c-dac/Advanced-java-practice-questions.php>
2. <http://learnerindia.com/Advanced-java-practice-questions.php>
3. <https://bsnlttapreparation.blogspot.com/2017/01/j2ee-20-mcq-test.html>
4. <https://www.careerride.com/test.aspx?type=Servlets>
5. <https://www.sanfoundry.com/1000-spring-questions-answers/>
6. <https://www.javatpoint.com/spring-quiz>
7. <https://www.tutorialspoint.com/spring/spring_online_quiz.htm>
8. <http://www.bullraider.com/quiz/spring-quzzies/66-quiz/321-spring-mvc-multiple-choice-questions>
9. <https://www.concretepage.com/quiz/spring-quiz/>
10. <https://www.proprofs.com/quiz-school/quizshow.php?title=3dq-how-well-do-you-know-spring-boot&q=1>
11. <https://www.springboottutorial.com/spring-boot-interview-questions>
12. <https://www.devglan.com/objectivequestionset/spring/set-1>
13. <https://www.devglan.com/objectivequestionset/spring/set-2>
14. <https://www.devglan.com/interviewquestions/hibernate>
15. <http://www.allindiaexams.in/engineering/cse/hibernate-mcq-quiz-hibernate-online-test>
16. <https://www.tutorialspoint.com/hibernate/hibernate_online_quiz.htm>
17. <https://www.onlineinterviewquestions.com/hibernate-mcq/>
18. <https://www.devglan.com/objectivequestionset/hibernate/set-1>
19. <https://career.guru99.com/hibernate-quiz/>
20. <https://www.freshersnow.com/hibernate-quiz/>
21. <https://www.concretepage.com/quiz/hibernate-quiz/>
22. <https://www.tutorialspoint.com/design_pattern/design_pattern_online_quiz.htm>
23. <https://www.freshersnow.com/design-patterns-quiz/>
24. <https://www.proprofs.com/quiz-school/story.php?title=quiz-gof-design-patterns-1>
25. <https://www.sanfoundry.com/spring-questions-answers-unit-testng-spring/>
26. <https://www.sanfoundry.com/spring-questions-answers-integration-testing/>
27. <https://www.sanfoundry.com/spring-questions-answers-jdbc-template/>
28. <https://www.sanfoundry.com/spring-quiz-questions-answers/>
29. <https://www.sanfoundry.com/spring-questions-answers-orm-frameworks/>
30. <https://www.sanfoundry.com/spring-mcqs-questions-answers/>
31. <https://www.sanfoundry.com/spring-questions-answers-transaction-management/>
32. <https://www.sanfoundry.com/spring-questions-answers-setting-transaction-attribute/>
33. <https://www.tutorialspoint.com/maven/maven_online_quiz.htm>
34. <https://career.guru99.com/top-20-maven-interview-questions/>