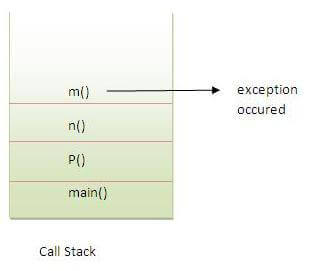
Frequently asked questions which I faced in many Interviews:  
Core java  
========  
custom exception,

exception propagation,

|  |
| --- |
| An exception is first thrown from the top of the stack and if it is not caught, it drops down the call  stack to the previous method,If not caught there, the exception again drops down to the previous  method, and so on until they are caught or until they reach the very bottom of the call stack.This  is called exception propagation. |

Rule: By default Unchecked Exceptions are forwarded in calling chain (propagated).



checked uncheked exception,

tricky progrms on exceptions.  
String StringBuffer StringBuilder difference,

### String vs StringBuffer

Since String is immutable in Java, whenever we do String manipulation like concatenation, substring etc, it generates a new String and discards the older String for garbage collection.

These are heavy operations and generate a lot of garbage in heap. So Java has provided StringBuffer and StringBuilder class that should be used for String manipulation.

StringBuffer and StringBuilder are mutable objects in java and provide append(), insert(), delete() and substring() methods for String manipulation.

### StringBuffer vs StringBuilder

StringBuffer was the only choice for String manipulation till Java 1.4 but it has one disadvantage that all of its public methods are synchronized. StringBuffer provides Thread safety but on a performance cost.

In most of the scenarios, we don’t use String in a multithreaded environment, so Java 1.5 introduced a new class StringBuilder that is similar to StringBuffer except thread safety and synchronization.

So if you are in a single threaded environment or don’t care about thread safety, you should use StringBuilder else use StringBuffer. See this post for [performance benchmarking between StringBuffer and StringBuilder](https://www.journaldev.com/137/stringbuffer-vs-stringbuilder).

### String vs StringBuffer vs StringBuilder

1. String is immutable whereas StringBuffer and StringBuider are mutable classes.
2. StringBuffer is thread safe and synchronized whereas StringBuilder is not, thats why [StringBuilder is more faster than StringBuffer](https://www.journaldev.com/137/stringbuffer-vs-stringbuilder).
3. String concat + operator internally uses StringBuffer or StringBuilder class.
4. For String manipulations in non-multi threaded environment, we should use StringBuilder else use StringBuffer class.

logical programs to reverse string,remove spaces from string,add given digits from string without using any predefined methods. Remove duplicates from array,find top 2 elements from array, reverse string using recursion algorithm.  
OOP concepts with real time example.  
Read multithreading and collection from scrach level.  
Collection is very important. Without collection u won't face any interview.  
Read all concepts of collection, differences of hashtable and hashmap, arraylist and linkedlist, hoe to sort names in ascending order,

comparator,

comparable,

how to iterate map,

1. If possible, always uses the Java 8 forEach.

Map<String, String> map = new HashMap<>();

map.forEach((key, value) -> System.out.println("[Key] : " + key + " [Value] : " + value));

Copy

2. Normal for loop in entrySet()

Map<String, String> map = new HashMap<>();

for (Map.Entry<String, String> entry : map.entrySet()) {

System.out.println("[Key] : " + entry.getKey() + " [Value] : " + entry.getValue());

}

Copy

3. Iterator, classic.

Map<String, String> map = new HashMap<>();

Iterator iter = map.entrySet().iterator();

while (iter.hasNext()) {

Map.Entry entry = (Map.Entry) iter.next();

System.out.println("[Key] : " + entry.getKey() + " [Value] : " + entry.getValue());

}

how to remove duplicates from List.

The [**LinkedHashSet**](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashSet.html) is the best approach for removing duplicate elements in an arraylist. LinkedHashSet does two things internally :

* Remove duplicate elements
* Maintain the order of elements added to it

 ArrayList<Integer> numbersList = new ArrayList<>(Arrays.asList(1, 1, 2, 3, 3, 3, 4, 5, 6, 6, 6, 7, 8));

        System.out.println(numbersList);

        LinkedHashSet<Integer> hashSet = new LinkedHashSet<>(numbersList);

        ArrayList<Integer> listWithoutDuplicates = new ArrayList<>(hashSet);

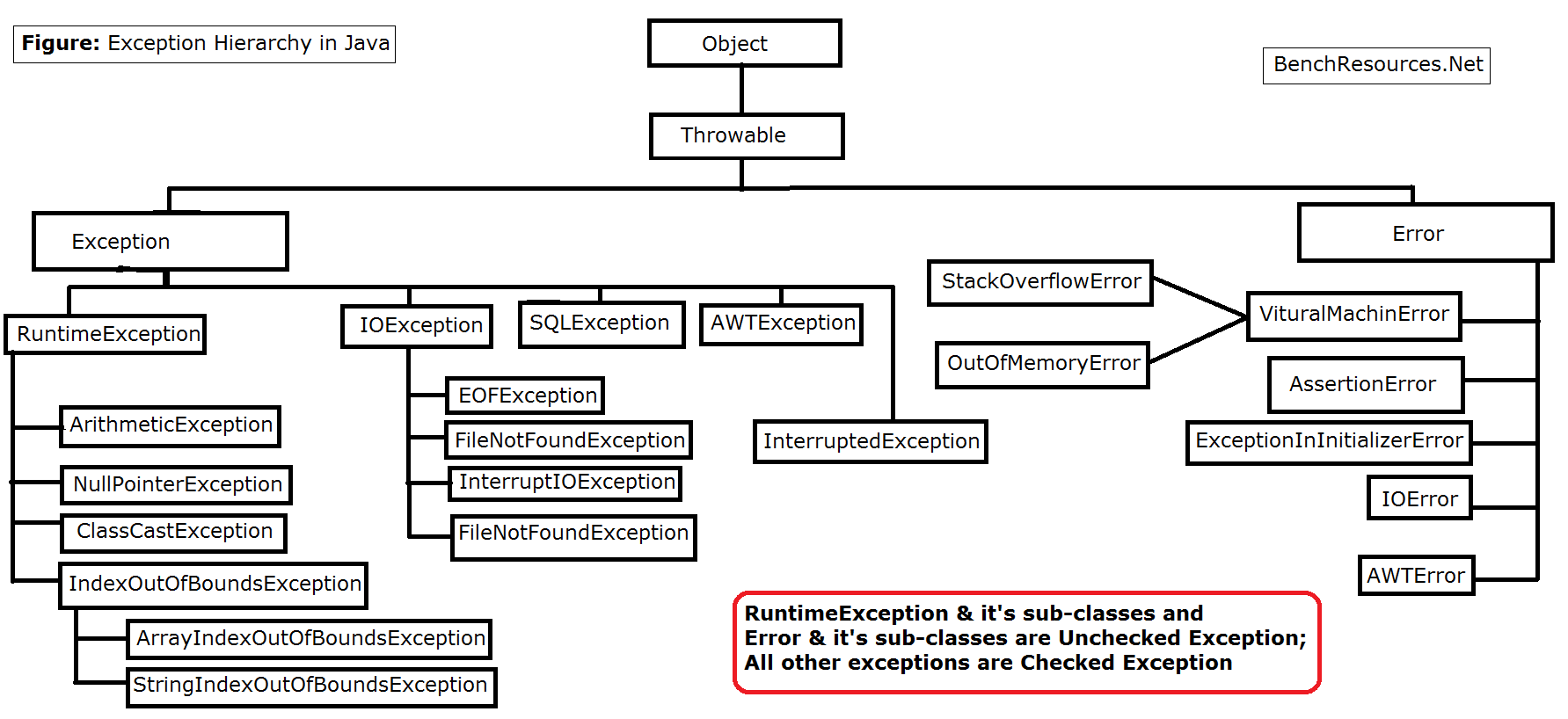
## 2. Remove duplicates in arraylist – Java 8

  ArrayList<Integer> numbersList = new ArrayList<>(Arrays.asList(1, 1, 2, 3, 3, 3, 4, 5, 6, 6, 6, 7, 8));

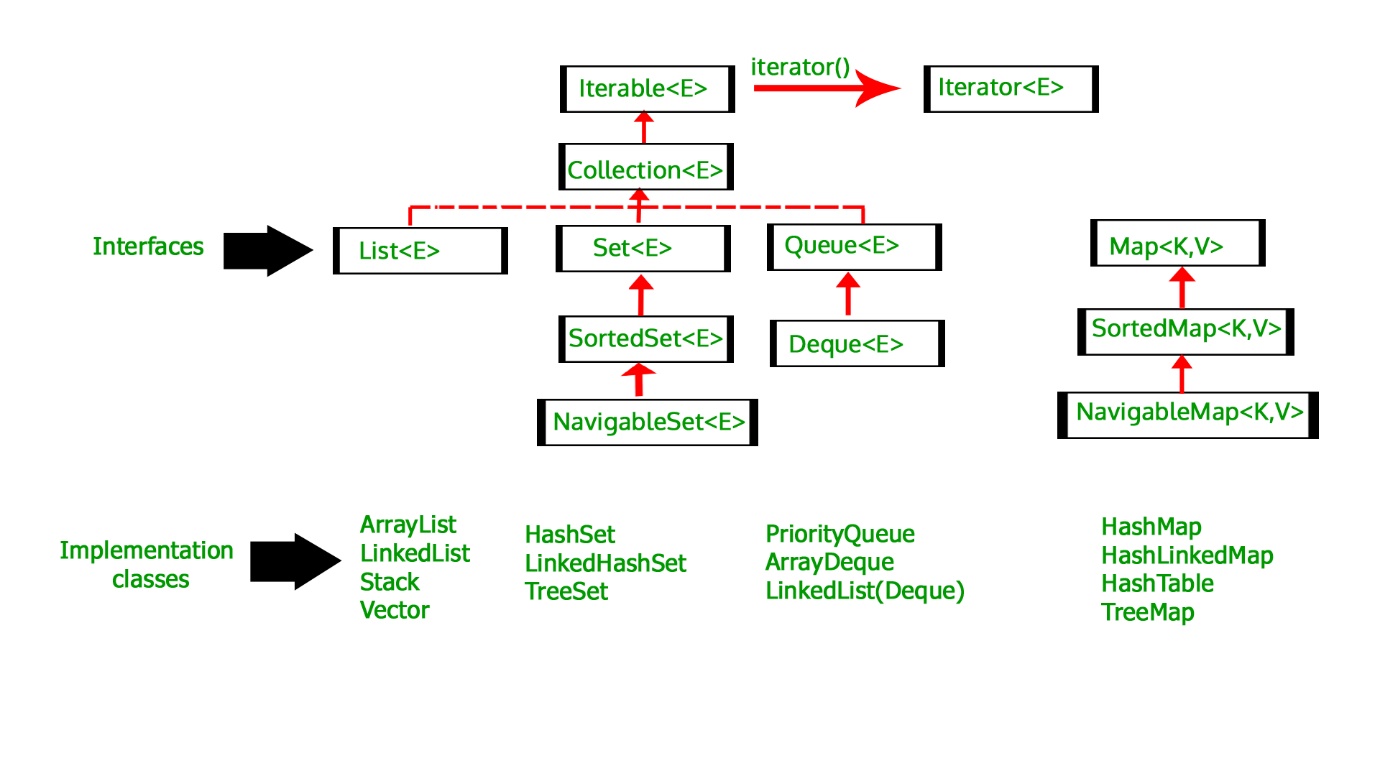
        System.out.println(numbersList);

        List<Integer> listWithoutDuplicates = numbersList.stream().distinct().collect(Collectors.toList());

Hierarchy of exceoption,



Collection.



Adv.java  
=======  
Get vs post

|  |
| --- |
| [**next →**](https://www.javatpoint.com/container)[**← prev**](https://www.javatpoint.com/http-requests) **Get vs. Post** There are many differences between the Get and Post request. Let's see these differences: |

|  |  |
| --- | --- |
| **GET** | **POST** |
| 1) In case of Get request, only **limited amount of data**can be sent because data is sent in header. | In case of post request,  **large amount of data**can be sent because  data is sent in body. |
| 2) Get request is **not secured**because data is exposed in URL bar. | Post request is **secured**because data is not  exposed in URL bar. |
| 3) Get request **can be bookmarked.** | Post request **cannot be bookmarked.** |
| 4) Get request is **idempotent**. It means second request will be ignored until response of first request is delivered | Post request is **non-idempotent.** |
| 5) Get request is **more efficient**and used more than Post. | Post request is **less efficient**and used less than get. |

Get



Some other features of GET requests are:

* It remains in the browser history
* It can be bookmarked
* It can be cached
* It have length restrictions
* It should never be used when dealing with sensitive data
* It should only be used for retrieving the data

Post

As we know, in case of post request original data is sent in message body. Let's see how information is passed to the server in case of post request.



Some other features of POST requests are:

* This requests cannot be bookmarked
* This requests have no restrictions on length of data
* This requests are never cached
* This requests do not retain in the browser history

servlet life cycle

A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.

* The servlet is initialized by calling the **init()** method.
* The servlet calls **service()** method to process a client's request.
* The servlet is terminated by calling the **destroy()** method.
* Finally, servlet is garbage collected by the garbage collector of the JVM.

## **The init() Method**

The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards. So, it is used for one-time initializations, just as with the init method of applets.

The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started.

When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. The init() method simply creates or loads some data that will be used throughout the life of the servlet.

The init method definition looks like this −

public void init() throws ServletException {

// Initialization code...

}

## **The service() Method**

The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client( browsers) and to write the formatted response back to the client.

Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

Here is the signature of this method −

public void service(ServletRequest request, ServletResponse response)

throws ServletException, IOException {

}

The service () method is called by the container and service method invokes doGet, doPost, doPut, doDelete, etc. methods as appropriate. So you have nothing to do with service() method but you override either doGet() or doPost() depending on what type of request you receive from the client.

The doGet() and doPost() are most frequently used methods with in each service request. Here is the signature of these two methods.

## **The doGet() Method**

A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// Servlet code

}

## **The doPost() Method**

A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// Servlet code

}

## **The destroy() Method**

The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

After the destroy() method is called, the servlet object is marked for garbage collection. The destroy method definition looks like this −

public void destroy() {

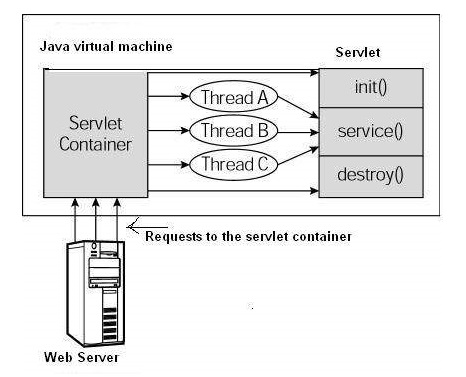
// Finalization code...

}

## **Architecture Diagram**

The following figure depicts a typical servlet life-cycle scenario.

* First the HTTP requests coming to the server are delegated to the servlet container.
* The servlet container loads the servlet before invoking the service() method.
* Then the servlet container handles multiple requests by spawning multiple threads, each thread executing the service() method of a single instance of the servlet.



session tracking mechanism

### **Session Tracking Techniques**

There are four techniques used in Session tracking:

1. **Cookies**

**Cookies** are text files stored on the client computer and they are kept for various information tracking purpose.  Browser stores this information on local machine for future use.

There are 2 types of cookies in servlets.

1. Non-persistent cookie
2. Persistent cookie

### **Non-persistent cookie**

It is **valid for single session** only. It is removed each time when user closes the browser.

### **Persistent cookie**

It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

### **Advantage of Cookies**

1. Simplest technique of maintaining the state.
2. Cookies are maintained at client side.

### **Disadvantage of Cookies**

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.

#### Note: Gmail uses cookie technique for login. If you disable the cookie, gmail won't work.

### **Cookie class**

**javax.servlet.http.Cookie** class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

### **Constructor of Cookie class**

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| Cookie() | constructs a cookie. |
| Cookie(String name, String value) | constructs a cookie with a specified  name and value. |

### **Useful Methods of Cookie class**

There are given some commonly used methods of the Cookie class.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void setMaxAge(int expiry) | Sets the maximum age of the cookie in seconds. |
| public String getName() | Returns the name of the cookie. The name cannot  be changed after creation. |
| public String getValue() | Returns the value of the cookie. |
| public void setName(String name) | changes the name of the cookie. |
| public void setValue(String value) | changes the value of the cookie. |

### **Other methods required for using Cookies**

|  |
| --- |
| For adding cookie or getting the value from the cookie, we need some methods provided by other interfaces. They are:   1. **public void addCookie(Cookie ck):**method of HttpServletResponse interface is used to add cookie in response object. 2. **public Cookie[] getCookies():**method of HttpServletRequest interface is used to return all the cookies from the browser. |

### **How to create Cookie?**

Let's see the simple code to create cookie.

1. Cookie ck=**new** Cookie("user","sonoo jaiswal");//creating cookie object
2. response.addCookie(ck);//adding cookie in the response

### **How to delete Cookie?**

Let's see the simple code to delete cookie. It is mainly used to logout or signout the user.

1. Cookie ck=**new** Cookie("user","");//deleting value of cookie
2. ck.setMaxAge(0);//changing the maximum age to 0 seconds
3. response.addCookie(ck);//adding cookie in the response

### **How to get Cookies?**

Let's see the simple code to get all the cookies.

1. Cookie ck[]=request.getCookies();
2. **for**(**int** i=0;i<ck.length;i++){
3. out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());//printing name and value of cookie
4. }
5. **Hidden Form Field**

A **hidden form field** is a technique used to store the session information for a particular client and is one of an important Session Tracking Technique.

In case of Hidden Form Field **a hidden (invisible) textfield** is used for maintaining the state of an user.

In such case, we store the information in the hidden field and get it from another servlet. This approach is better if we have to submit form in all the pages and we don't want to depend on the browser.

Let's see the code to store value in hidden field.

1. <input type="hidden" name="uname" value="Vimal Jaiswal">

Here, uname is the hidden field name and Vimal Jaiswal is the hidden field value.

### **Real application of hidden form field**

It is widely used in comment form of a website. In such case, we store page id or page name in the hidden field so that each page can be uniquely identified.

### **Advantage of Hidden Form Field**

1. It will always work whether cookie is disabled or not.

### **Disadvantage of Hidden Form Field:**

1. It is maintained at server side.
2. Extra form submission is required on each pages.
3. Only textual information can be used.
4. **URL Rewriting**

**URL rewriting** is a method in which the requested **URL** is modified to include a session ID. There are several ways to perform **URL rewriting**. You are going to look at one method that is provided by the **Servlet** API.

In URL rewriting, we append a token or identifier to the URL of the next Servlet or the next resource. We can send parameter name/value pairs using the following format:

url?name1=value1&name2=value2&??

### **Advantage of URL Rewriting**

1. It will always work whether cookie is disabled or not (browser independent).
2. Extra form submission is not required on each pages.

### **Disadvantage of URL Rewriting**

1. It will work only with links.
2. It can send Only textual information.

 String n=request.getParameter("userName");

        out.print("Welcome "+n);

        //appending the username in the query string

        out.print("<a href='servlet2?uname="+n+"'>visit</a>");

     out.close();

1. **HttpSession**

javax.servlet.http. Interface **HttpSession**. public interface **HttpSession**. Provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user. The servlet container uses this interface to create a session between an HTTP client and an HTTP server.

In such case, container creates a session id for each user.The container uses this id to identify the particular user.An object of HttpSession can be used to perform two tasks:

1. bind objects
2. view and manipulate information about a session, such as the session identifier, creation time, and last accessed time.



### **How to get the HttpSession object ?**

The HttpServletRequest interface provides two methods to get the object of HttpSession:

1. **public HttpSession getSession():**Returns the current session associated with this request, or if the request does not have a session, creates one.
2. **public HttpSession getSession(boolean create):**Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.

### **Commonly used methods of HttpSession interface**

1. **public String getId():**Returns a string containing the unique identifier value.
2. **public long getCreationTime():**Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
3. **public long getLastAccessedTime():**Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT.
4. **public void invalidate():**Invalidates this session then unbinds any objects bound to it.

 session.setAttribute("uname",n);

String n=(String)session.getAttribute("uname");

what is Cookie?  
Double posting problem

Pressing refresh button on result page from submission executes request one more time and gives side effect like duplication registration same customer and deducting amount from credit/debit card one more time and etc. To overcome this problem we can use session tokens.

**DoublePostingPreventingFilter.java**

@WebFilter(value="/\*")

public class DoublePostingPreventingFilter implements Filter{

@Override

public void init(FilterConfig fg) throws ServletException {

}

HttpSession session = null ;

HttpServletRequest hreq = null ;

int serverToken = 0 ;

int clientToken = 0 ;

//type casting

@Override

public void doFilter(ServletRequest req, ServletResponse res,

FilterChain chain) throws IOException, ServletException {

hreq = (HttpServletRequest)req;

//session = hreq.getSession(true);

//for 1st request when Session obj is not there for browser

if(hreq.getMethod().equals("GET")){

session = hreq.getSession(true);

session.setAttribute("stoken", new Random().nextInt(10000));

chain.doFilter(req, res);

}else{

//read ServerSessionToken

serverToken = (Integer)session.getAttribute("stoken");

//read clientSessionToken

System.out.println("serverToken:"+serverToken);

clientToken = Integer.parseInt(req.getParameter("ctoken"));

System.out.println("clientToken:"+clientToken);

System.out.println(clientToken+"----------"+serverToken);

if(serverToken == clientToken){

session.setAttribute("stoken", new Random().nextInt(10000));

chain.doFilter(req, res);

}else{

RequestDispatcher rd = req.getRequestDispatcher("/dbl\_post\_err.jsp");

rd.forward(req, res);

}

}

}

@Override

public void destroy() {

}

}

difference between forward and sendredirect

Simply difference between Forward(ServletRequest request, ServletResponse response) and sendRedirect(String url) is

**forward():**

1. The forward() method is executed in the server side.
2. The request is transfer to other resource within same server.
3. It does not depend on the client’s request protocol since the forward () method is provided by the servlet container.
4. The request is shared by the target resource.
5. Only one call is consumed in this method.
6. It can be used within server.
7. We cannot see forwarded message, it is transparent.
8. The forward() method is faster than sendRedirect() method.
9. It is declared in RequestDispatcher interface.

**sendRedirect():**

1. The sendRedirect() method is executed in the client side.
2. The request is transfer to other resource to different server.
3. The sendRedirect() method is provided under HTTP so it can be used only with HTTP clients.
4. New request is created for the destination resource.
5. Two request and response calls are consumed.
6. It can be used within and outside the server.
7. We can see redirected address, it is not transparent.
8. The sendRedirect() method is slower because when new request is created old request object is lost.
9. It is declared in HttpServletResponse.

session config vs session context

ServletConfig  
  
1.ServletConfig available in javax.servlet.\*; package  
2.ServletConfig object is one per servlet class  
3.destroyed once the servlet execution is completed.  
4.We should give request explicitly, in order to create ServletConfig object for the first time  
5.Object of ServletConfig will be created during initialization process of the servlet  
6.This Config object is public to a particular servlet only  
  
ServletContext  
  
1.ServletContext available in javax.servlet.\*; package  
2.ServletContext object is global to entire web application  
3.Object of ServletContext will be created at the time of web application deployment  
4.Scope: As long as web application is executing, ServletContext object will be available, and 5.it will be destroyed once the application is removed from the server.  
6.ServletContext object will be available even before giving the first request

request.forward() vs request.include()

|  |  |  |
| --- | --- | --- |
| PROPERTY | INCLUDE() METHOD | FORWARD() METHOD |
| WHAT CAN BE DONE? | Includes another file in our current file. | Will forward the client request to the forwarding page. |
| MERGE OF RESPONSE | Response of S1 and S2 are merged and sent to client (as if a single response). This way, the Programmer can achieve "server side includes". | No merge of response. Only S2 response will go to the client. |
| RETAINING EXECUTION CONTROL | Shifted temporarily from S1 to S2. It works like a general simple method call. | Shifted permanently from S1 to S2. |
| CONTROL COMING BACK | Execution control comes back to S1 after executing S2 for further processing of S1 after include() statement from where the execution control shifted. | Once shifted, the control never returns to S1. It is permanent shifting. |
| RESPONSE PLACEMENT | Response of S2 is placed in S1. | Response of S2 is not placed in S2. |
| CLIENT RECEIVES | Response of S1 and S2 is received by client. | Only response of S2 is received by client. |
| CONTROL RETURNED | After executing S2, control returned to S1. | After executing S2, control returned to client. |
| EXTRA ACTIVITY | Once control is returned to S1 from S2, any activity can be done on the server like calling another servlet with another RequestDispatcher object. | Once control returned to client, no activity can be done on S1 or S2. |
| USAGE | Used by Programmer when the output of both servlets S1 and S2 is required. | Used only S2 response is required. |
| SPEED OF DELIVERY TO CLIENT | Comparatively slower. | Faster. |
| ACCESS | S2 has access to the request and response objects of S1, but limitations exist. S2 cannot set headers and also cannot call any other method like setCookie etc. affecting the response headers. That is, S2 cannot attempt to change the HTTP headers or response status code etc. and performing any activities like this is simply ignored. | Here also S2 cannot alter as response is delivered on S1 URL. |
| USAGE | Often useful just to include another web resource response like banner content or copyright information etc. | Used when further processing responsibility and replying to user is given to another Web resource. |
| OUT.PRINTLN | Output of S1’s out.println() statements go to client. | Output of S1’s out.println() statements never go to client. |
| CLIENT RESPONSE | Client receives the response from the same servlet which he requested. | Client actually receives the response from a different servlet (not known to client). |
| TREATMENT OF PROCESSING | The processing S2 can be treated as part of S2 processing. | The processing of S2 can be treated completely as a different entity. It is used to show a different resource on the server by S1. |
| USED WHEN | Used when static information is to be included. | Used when dynamic information is to be included. It is can be used where a Servlet can play the role of a controller to process the client input and deciding what response page is to be returned. |

After knowing include vs forward, let us go their similarities.

**Similarities:**

1. Both are methods of RequestDispatcher interface.
2. Both are method calls.
3. Both are abstract methods in the interface but can be treated as concrete methods in Servlet programming. I mean, because we are not implementing RequestDispatcher to our Servlet program, there is no compulsion to the Programmer to override both.
4. Processing done on the server when both methods are called is completely transparent (not visible or known) to client.  
   Both are not treated as redirection.

JSP  
====  
explain in detail implicit objects

In this chapter, we will discuss the Implicit Objects in JSP. These Objects are the Java objects that the JSP Container makes available to the developers in each page and the developer can call them directly without being explicitly declared. JSP Implicit Objects are also called **pre-defined variables**.

Following table lists out the nine Implicit Objects that JSP supports −

|  |  |
| --- | --- |
| **S.No.** | **Object & Description** |
| 1 | **request**  This is the **HttpServletRequest** object associated with the request. |
| 2 | **response**  This is the **HttpServletResponse** object associated with the response to the client. |
| 3 | **out**  This is the **PrintWriter** object used to send output to the client. |
| 4 | **session**  This is the **HttpSession** object associated with the request. |
| 5 | **application**  This is the **ServletContext** object associated with the application context. |
| 6 | **config**  This is the **ServletConfig** object associated with the page. |
| 7 | **pageContext**  This encapsulates use of server-specific features like higher performance **JspWriters**. |
| 8 | **page**  This is simply a synonym for **this**, and is used to call the methods defined by the translated servlet class. |
| 9 | **Exception**  The **Exception** object allows the exception data to be accessed by designated JSP. |

## **The request Object**

The request object is an instance of a **javax.servlet.http.HttpServletRequest** object. Each time a client requests a page the JSP engine creates a new object to represent that request.

The request object provides methods to get the HTTP header information including form data, cookies, HTTP methods etc.

We can cover a complete set of methods associated with the request object in a subsequent chapter − [JSP - Client Request](https://www.tutorialspoint.com/jsp/jsp_client_request.htm).

## **The response Object**

The response object is an instance of a **javax.servlet.http.HttpServletResponse** object. Just as the server creates the request object, it also creates an object to represent the response to the client.

The response object also defines the interfaces that deal with creating new HTTP headers. Through this object the JSP programmer can add new cookies or date stamps, HTTP status codes, etc.

We will cover a complete set of methods associated with the response object in a subsequent chapter − [JSP - Server Response](https://www.tutorialspoint.com/jsp/jsp_server_response.htm).

## **The out Object**

The out implicit object is an instance of a **javax.servlet.jsp.JspWriter** object and is used to send content in a response.

The initial JspWriter object is instantiated differently depending on whether the page is buffered or not. Buffering can be easily turned off by using the **buffered = 'false'** attribute of the page directive.

The JspWriter object contains most of the same methods as the **java.io.PrintWriter** class. However, JspWriter has some additional methods designed to deal with buffering. Unlike the PrintWriter object, JspWriter throws **IOExceptions**.

Following table lists out the important methods that we will use to write **boolean char, int, double, object, String**, etc.

|  |  |
| --- | --- |
| **S.No.** | **Method & Description** |
| 1 | **out.print(dataType dt)**  Print a data type value |
| 2 | **out.println(dataType dt)**  Print a data type value then terminate the line with new line character. |
| 3 | **out.flush()**  Flush the stream. |

## **The session Object**

The session object is an instance of **javax.servlet.http.HttpSession** and behaves exactly the same way that session objects behave under Java Servlets.

The session object is used to track client session between client requests. We will cover the complete usage of session object in a subsequent chapter − [JSP - Session Tracking](https://www.tutorialspoint.com/jsp/jsp_session_tracking.htm).

## **The application Object**

The application object is direct wrapper around the **ServletContext** object for the generated Servlet and in reality an instance of a **javax.servlet.ServletContext** object.

This object is a representation of the JSP page through its entire lifecycle. This object is created when the JSP page is initialized and will be removed when the JSP page is removed by the **jspDestroy()** method.

By adding an attribute to application, you can ensure that all JSP files that make up your web application have access to it.

We will check the use of Application Object in [JSP - Hits Counter](https://www.tutorialspoint.com/jsp/jsp_hits_counter.htm) chapter.

## **The config Object**

The config object is an instantiation of **javax.servlet.ServletConfig** and is a direct wrapper around the **ServletConfig** object for the generated servlet.

This object allows the JSP programmer access to the Servlet or JSP engine initialization parameters such as the paths or file locations etc.

The following **config** method is the only one you might ever use, and its usage is trivial −

config.getServletName();

This returns the servlet name, which is the string contained in the **<servlet-name>** element defined in the **WEB-INF\web.xml** file.

## **The pageContext Object**

The pageContext object is an instance of a **javax.servlet.jsp.PageContext** object. The pageContext object is used to represent the entire JSP page.

This object is intended as a means to access information about the page while avoiding most of the implementation details.

This object stores references to the request and response objects for each request. The **application, config, session**, and out objects are derived by accessing attributes of this object.

The pageContext object also contains information about the directives issued to the JSP page, including the buffering information, the errorPageURL, and page scope.

The PageContext class defines several fields, including **PAGE\_SCOPE, REQUEST\_SCOPE, SESSION\_SCOPE,** and **APPLICATION\_SCOPE**, which identify the four scopes. It also supports more than 40 methods, about half of which are inherited from the **javax.servlet.jsp.JspContext class**.

One of the important methods is **removeAttribute**. This method accepts either one or two arguments. For example, **pageContext.removeAttribute ("attrName")** removes the attribute from all scopes, while the following code only removes it from the page scope −

pageContext.removeAttribute("attrName", PAGE\_SCOPE);

The use of pageContext can be checked in [JSP - File Uploading](https://www.tutorialspoint.com/jsp/jsp_file_uploading.htm) chapter.

## **The page Object**

This object is an actual reference to the instance of the page. It can be thought of as an object that represents the entire JSP page.

The page object is really a direct synonym for the **this** object.

## **The exception Object**

The exception object is a wrapper containing the exception thrown from the previous page. It is typically used to generate an appropriate response to the error condition.

why Jsp if we can use Servlet

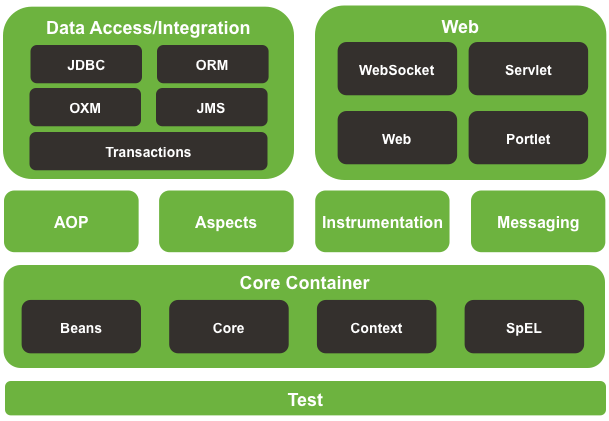
## **Difference Between Servlet and JSP**

In this article we will list some of the differences between Servlets and JSP.

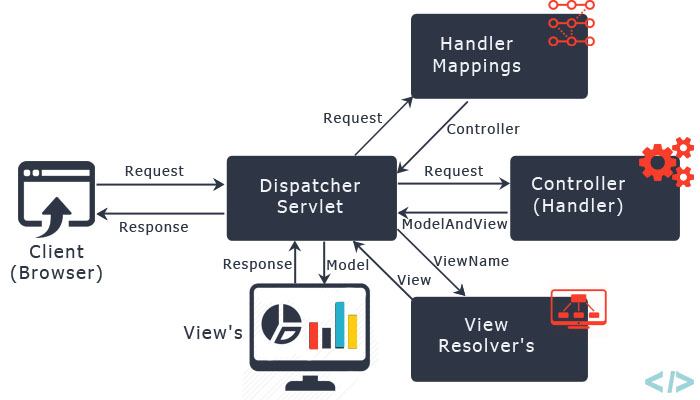
| **SERVLET** | **JSP** |
| --- | --- |
| A servlet is a server-side program and written purely on Java. | JSP is an interface on top of Servlets. In another way,  we can say that JSPs are extension of servlets to minimize  the effort of developers to write User Interfaces using  Java programming. |
| Servlets run faster than JSP | JSP runs slower because it has the transition phase for  converting from JSP page to a Servlet file. Once it is  converted to a Servlet then it will start the compilation |
| Executes inside a Web server, such as Tomcat | A JSP program is compiled into a Java servlet before  execution. Once it is compiled into a servlet, it's life cycle  will be same as of servlet. But, JSP has it's own API for the lifecycle. |
| Receives HTTP requests from users and provides HTTP responses | Easier to write than servlets as it is similar to HTML. |
| We can not build any custom tags | One of the key advantage is we can build custom tags  using JSP API (there is a separate package available for  writing the custom tags) which can be available as the  re-usable components with lot of flexibility |
| Servlet has the life cycle methods init(), service() and destroy() | JSP has the life cycle methods of jspInit(), \_jspService()  and jspDestroy() |
| Written in Java, with a few additional APIs specific to this kind of processing. Since it is written in Java, it follows all the Object Oriented programming techniques. | JSPs can make use of the Javabeans inside the web pages |
| In MVC architecture Servlet acts as controller. | In MVC architecture JSP acts as view. |
| **Servlet advantages include:** **1. Performance :** get loaded upon first request and remains in memory idenfinately. **2. Simplicity :**Run inside controlled server environment. No specific client software is needed:web broser is enough **3. Session Management** : overcomes HTTP's stateless nature **4. Java Technology :**network access,Database connectivity, j2ee integration | **JSP Provides an extensive infrastructure for:** 1. Tracking sessions. 2. Managing cookies. 3. Reading and sending HTML headers. 4. Parsing and decoding HTML form data. 5. **JSP is Efficient:** Every request for a JSP is handled by  a simple Java thread 6. **JSP is Scalable:** Easy integration with other backend  services 7. **Seperation of roles:** Developers, Content  Authors/Graphic Designers/Web Masters |

EL expression  
tags  
JSTL

spring  
=====  
spring core module



and MVC flow



diff between setter and constructor injection

setter injection

setting properties of bean through setter methods

[Difference between constructor and setter injection](https://www.javatpoint.com/difference-between-constructor-and-setter-injection)

There are many key differences between constructor injection and setter injection.

1. **Partial dependency**: can be injected using setter injection but it is not possible by constructor. Suppose there are 3 properties in a class, having 3 arg constructor and setters methods. In such case, if you want to pass information for only one property, it is possible by setter method only.
2. **Overriding**: Setter injection overrides the constructor injection. If we use both constructor and setter injection, IOC container will use the setter injection.
3. **Changes**: We can easily change the value by setter injection. It doesn't create a new bean instance always like constructor. So setter injection is flexible than constructor injection.
4. **Constructor injection is faster**

diff between BeanFactory and ApplicationContext

**1.** **BeanFactory**is fundamental IOC Container , it able to provide fundamental functionalities to the spring applications like creating and maintaning bean objects.

**ApplicationContext**IOC Container is an extension of BeanFactory IOC Container , it able to provide some advanced features like Internationalization, Event Handling.. etc along with fundamental functionalities what BeanFactory is providing.

**2. BeanFactory** is not supporting to integrate AOP services like Security, JTA,... to the spring applications.

**ApplicationContext**is supporting to integrate AOP services like Security, JTA,... to the spring applications.

**3. BeanFactory**is able to prepare Singleton objects when we send first request for bean, that is, Lazy Instantiation/Initialization.

**ApplicationContext**is able to prepare Singleton objects when we activate Container , that is , early Instantiation/Initialization.

**4. BeanFactory** is supporting only the scopes like Singleton and Prototype.

**ApplicationContext**is supporting almost all the Spring scopes like Singleton, Prototype, request, session, globalSession, webSocket.. etc.

**5. BeanFactory** is mainly for Standalone Applications.

**ApplicationContext**is for all the types of Spring framework applications.

**6. BeanFactory** is an outdated Container in Spring applications.

**ApplicationContext**is not outdated Container.

What is autowiring?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Autowiring feature of spring framework enables you to inject the object dependency implicitly. It internally uses setter or constructor injection.  Autowiring can't be used to inject primitive and string values. It works with reference only. **Advantage of Autowiring** It requires the **less code** because we don't need to write the code to inject the dependency explicitly. **Disadvantage of Autowiring** No control of programmer.  It can't be used for primitive and string values. **Autowiring Modes** There are many autowiring modes:   |  |  |  | | --- | --- | --- | | **No.** | **Mode** | **Description** | | 1) | no | It is the default autowiring mode. It means no autowiring  bydefault. | | 2) | byName | The byName mode injects the object dependency according  to name of the bean. In such case, property name and bean  name must be same. It internally calls setter method. | | 3) | byType | The byType mode injects the object dependency according to  type. So property name and bean name can be different.  It internally callssetter method. | | 4) | constructor | The constructor mode injects the dependency by calling the  constructor of the class. It calls the constructor having large  number of parameters. | | 5) | autodetect | It is deprecated since Spring 3. | |

<bean id="a" **class**="org.sssit.A" autowire="byName"></bean>

What are the annotation you used in spring?

1. @Configuration: Used to indicate that a class declares one or more @Bean methods. These classes are processed by the Spring container to generate bean definitions and service requests for those beans at runtime.
2. @Bean: Indicates that a method produces a bean to be managed by the Spring container. This is one of the most used and important spring annotation. @Bean annotation also can be used with parameters like name, initMethod and destroyMethod.
   * name – allows you give name for bean
   * initMethod – allows you to choose method which will be invoked on context register
   * destroyMethod – allows you to choose method which will be invoked on context shutdown

For example:

@Configuration

public class AppConfig {

@Bean(name = "comp", initMethod = "turnOn", destroyMethod = "turnOff")

Computer computer(){

return new Computer();

}

}

public class Computer {

public void turnOn(){

System.out.println("Load operating system");

}

public void turnOff(){

System.out.println("Close all programs");

}

}

1. @PreDestroy and @PostConstruct are alternative way for bean initMethod and destroyMethod. It can be used when the bean class is defined by us. For example;
2. public class Computer {
3. @PostConstruct
4. public void turnOn(){
5. System.out.println("Load operating system");
6. }
7. @PreDestroy
8. public void turnOff(){
9. System.out.println("Close all programs");
10. }
11. }
12. @ComponentScan: Configures component scanning directives for use with @Configuration classes. Here we can specify the base packages to scan for spring components.
13. @Component: Indicates that an annotated class is a “component”. Such classes are considered as candidates for auto-detection when using annotation-based configuration and classpath scanning.
14. @PropertySource: provides a simple declarative mechanism for adding a property source to Spring’s Environment. There is a similar annotation for adding an array of property source files i.e @PropertySources.
15. @Service: Indicates that an annotated class is a “Service”. This annotation serves as a specialization of @Component, allowing for implementation classes to be autodetected through classpath scanning.
16. @Repository: Indicates that an annotated class is a “Repository”. This annotation serves as a specialization of @Component and advisable to use with [DAO](https://www.journaldev.com/16813/dao-design-pattern) classes.
17. @Autowired: [Spring @Autowired annotation](https://www.journaldev.com/2623/spring-autowired-annotation) is used for automatic injection of beans. Spring @Qualifier annotation is used in conjunction with Autowired to avoid confusion when we have two of more bean configured for same type.

### Spring MVC Annotations

Some of the important Spring MVC annotations are:

1. @Controller
2. @RequestMapping
3. @PathVariable
4. @RequestParam
5. @ModelAttribute
6. @RequestBody and @ResponseBody
7. @RequestHeader and @ResponseHeader

### Spring Transaction Management Annotations

@Transactional is the spring declarative transaction management annotation, read more at [Spring MVC Hibernate](https://www.journaldev.com/3531/spring-mvc-hibernate-mysql-integration-crud-example-tutorial).

### Spring Security Annotations

@EnableWebSecurity is used with @Configuration class to have the Spring Security configuration defined, read more at [Spring Security Example](https://www.journaldev.com/2715/spring-security-example-tutorial).

### Spring Boot Annotations

1. @SpringBootApplication
2. @EnableAutoConfiguration

What are the scopes in Spring?

|  |  |
| --- | --- |
| **Sr.No.** | **Scope & Description** |
| 1 | **singleton**  This scopes the bean definition to a single instance per Spring IoC container (default). |
| 2 | **prototype**  This scopes a single bean definition to have any number of object instances. |
| 3 | **request**  This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext. |
| 4 | **session**  This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext. |
| 5 | **global-session**  This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext. |

What is bean inheritence ,Inner beans?

**Spring** - **Bean** Definition **Inheritance**. ... A child **bean** definition **inherits** configuration data from a parent definition. The child definition can override some values, or add others, as needed.

<bean id="CustomerBean" parent="BaseCustomerMalaysia">

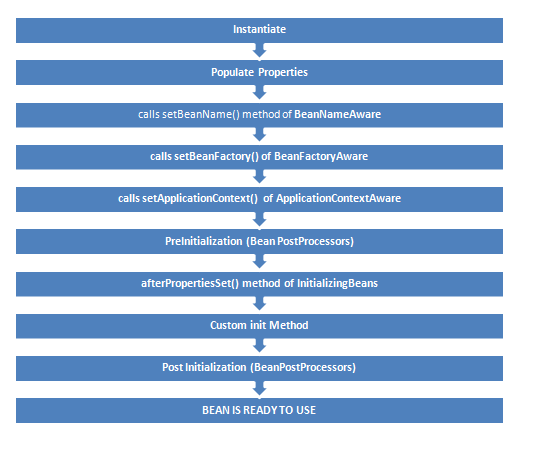
<property name="action" value="buy" />

<property name="type" value="1" />

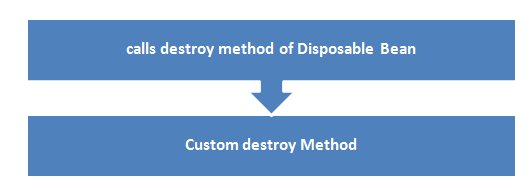
</bean>

**Inner beans** are the **beans** that are defined within the scope of another **bean**. Thus, a <**bean**/> element inside the <property/> or <constructor-arg/> elements is called **inner bean**.

Bean life cycle

)

Following diagram shows the method calling  at the time of destruction.



BeanPostProcessor

The **BeanPostProcessor** interface defines callback methods that you can implement to provide your own instantiation logic, dependency-resolution logic, etc. You can also implement some custom logic after the Spring container finishes instantiating, configuring, and initializing a bean by plugging in one or more BeanPostProcessor implementations.

public Object postProcessBeforeInitialization(Object bean, String beanName)

throws BeansException {

System.out.println("BeforeInitialization : " + beanName);

return bean; // you can return any other object as well

}

Explain Spring MVC architecture.

[**next →**](https://www.javatpoint.com/spring-mvc-multiple-view-page-example)[**← prev**](https://www.javatpoint.com/using-variable-in-spel)

# **Spring MVC Tutorial**

A Spring MVC is a Java framework which is used to build web applications. It follows the Model-View-Controller design pattern. It implements all the basic features of a core spring framework like Inversion of Control, Dependency Injection.

A Spring MVC provides an elegant solution to use MVC in spring framework by the help of **DispatcherServlet**. Here, **DispatcherServlet** is a class that receives the incoming request and maps it to the right resource such as controllers, models, and views.

## **Spring Web Model-View-Controller**



* **Model** - A model contains the data of the application. A data can be a single object or a collection of objects.
* **Controller** - A controller contains the business logic of an application. Here, the @Controller annotation is used to mark the class as the controller.
* **View** - A view represents the provided information in a particular format. Generally, JSP+JSTL is used to create a view page. Although spring also supports other view technologies such as Apache Velocity, Thymeleaf and FreeMarker.
* **Front Controller** - In Spring Web MVC, the DispatcherServlet class works as the front controller. It is responsible to manage the flow of the Spring MVC application.

Hi friends, this is arun..after seeing my previous post lot of people called me and asked me how to crack interview as experienced java developer.

few tips note down it,

tell about yourself with start your name and end with your roles and responsibility in previous project which you have worked in previous company.

ex- my self a arun kumar basically i am from odisha, i have completed bca in 2013,after completion of my studies i started my carrier by working in infogrid technology that is there in banglore. in this organization i am working as a java developer. i have been working in this organisation since 3.4 years in one project named as AGD. AGD stands for all goods details .its a erp domain project and our client for this project from Europe.  
the main objective of this project is a user can interact with suppliers through online. user can get the status through online by mail about transpored product and availability.user check and view about historical data and user can generate his own reports.this application is acts as a mediator between user and suplliers.  
to implement this project we have been following agile mathodology.we are developing code and transfer to tester.afer completion of testing we deploy into live environment then another sprint wil be released by our scrum master. again we follow same. each sprint we takes around 15 to 20 days.and we conduct daily one meeting roughly around 15 minutes about. in this duration we talk about what we have done yesterday what problem we faced and what we are going to do next.

in this project my roles and responsibilities are  
implementing dao by using hibernate  
developing modules by following spring mvc.  
testing web serviece by using saop ui.  
generate wsdl, creat wsdl.  
debug the code and fixing bugs.  
adding jars in pom.xml.  
to make application to get better performation developing code by following several design pattern like singleton, factory pattern, templet, abstarct factory pattern etc.

implementing bussiness logics,  
analys bussiness requirement.

these are my roles and responsibilities in this projects

then the interviewer will start asking queations from core java to frameworks.

important topics in core java  
1 strings  
2oops  
3collections include concurrent collections  
4multi threading  
5 several logical programs like, reversing, sorting, etc

hibernate  
1 difference between get and load

## **1. session.load()**

* It will always return a “**proxy**” (Hibernate term) without hitting the database. In Hibernate, proxy is an object with the given identifier value, its properties are not initialized yet, it just look like a temporary fake object.
* If no row found , it will throws an **ObjectNotFoundException**.

## **2. session.get()**

* It always **hit the database** and return the real object, an object that represent the database row, not proxy.
* If no row found , it return **null**.

3 difference between merge and saveOrUpate

session.update() - It is used for a scenario when you have load an object Person1 from hibernate session. Now it is being used in application - may be on client side also, has been updated. We want to save it again. We know that no change has been done in Person object in data base. So we can simply use update.

session.merge() - In above scenario, if changes have been done in person data before saving the changed Person1 object, then we should use merge. It will merge the changes.

session.save() - It can be used to save a new object. It returns the serialiable identity.

session.persist() - It is same as save(), but is void method and does not return anything.

session.saveOrUpdate() - This method will work for both new and old objects. If object is new, it will work like simple save or if object is already persistent, it will work like update.

session.lock() - It is used only to take the lock on object or you can say to check the version of object. It is not meant for updating the object. It should be used to reattach the object, only if you are sure that object state is not changed in database already. Otherwise, it may override the changes. < Inviting more points on this.

4 caching

**Hibernate** - **Caching**. **Caching** is a mechanism to enhance the performance of a system. It is a buffer memorythat lies between the application and the database. **Cache** memory stores recently used data items in order to reduce the number of database hits as much as possible.

## **First-level Cache**

The first-level cache is the Session cache and is a mandatory cache through which all requests must pass. The Session object keeps an object under its own power before committing it to the database.

If you issue multiple updates to an object, Hibernate tries to delay doing the update as long as possible to reduce the number of update SQL statements issued. If you close the session, all the objects being cached are lost and either persisted or updated in the database.

## **Second-level Cache**

Second level cache is an optional cache and first-level cache will always be consulted before any attempt is made to locate an object in the second-level cache. The second level cache can be configured on a per-class and per-collection basis and mainly responsible for caching objects across sessions.

Any third-party cache can be used with Hibernate. An **org.hibernate.cache.CacheProvider** interface is provided, which must be implemented to provide Hibernate with a handle to the cache implementation.

## **Query-level Cache**

Hibernate also implements a cache for query resultsets that integrates closely with the second-level cache.

This is an optional feature and requires two additional physical cache regions that hold the cached query results and the timestamps when a table was last updated. This is only useful for queries that are run frequently with the same parameters.

## **Cache Provider**

Your next step after considering the concurrency strategies, you will use your cache candidate classes to pick a cache provider. Hibernate forces you to choose a single cache provider for the whole application.

|  |  |
| --- | --- |
| **Sr.No.** | **Cache Name & Description** |
| 1 | **EHCache**  It can cache in memory or on disk and clustered caching and it supports the optional Hibernate query result cache. |
| 2 | **OSCache**  Supports caching to memory and disk in a single JVM with a rich set of expiration policies and query cache support. |
| 3 | **warmCache**  A cluster cache based on JGroups. It uses clustered invalidation, but doesn't support the Hibernate query cache. |
| 4 | **JBoss Cache**  A fully transactional replicated clustered cache also based on the JGroups multicast library. It supports replication or invalidation, synchronous or asynchronous communication, and optimistic and pessimistic locking. The Hibernate query cache is supported. |

Every cache provider is not compatible with every concurrency strategy. The following compatibility matrix will help you choose an appropriate combination.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Strategy/Provider** | **Read-only** | **Nonstrictread-write** | **Read-write** | **Transactional** |
| EHCache | X | X | X |  |
| OSCache | X | X | X |  |
| SwarmCache | X | X |  |  |
| JBoss Cache | X |  |  | X |

5 mappings one to many,many to one many to many

## When to use one to many mapping

Use one to mapping to create **1..N relationship** between entities or objects.

For example, we have to write two entities i.e. EmployeeEntity and AccountEntity such that multiple accounts can be associated with a single employee, but one single account can not be shared between two or more employees.

## Hibernate one to many mapping solutions

This problem can be solved in two different ways.

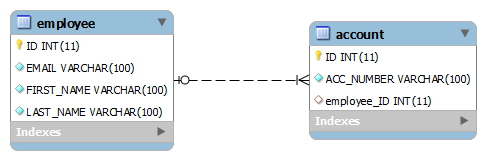
1. One is to have a **foreign key column** in account table i.e. EMPLOYEE\_ID. This column will refer to primary key of Employee table. This way no two accounts can be associated with multiple employees. Obviously, account number needs to be unique for enforcing this restriction.
2. Second approach is to have a common **join table** lets say EMPLOYEE\_ACCOUNT. This table will have two column i.e. EMP\_ID which will be foreign key referring to primary key in EMPLOYEE table and similarly ACCOUNT\_ID which will be foreign key referring to primary key of ACCOUNT table.

## 1. Hibernate one to many mapping with foreign key association

In this approach, **both entity will be responsible for making the relationship** and maintaining it. EmployeeEntity should declare that relationship is one to many, and AccountEntity should declare that relationship from its end is many to one.

#### 1.1. Design one to many mapping relationship

Lets first see the schema design.



@Entity(name = "ForeignKeyAssoEntity")

@Table(name = "Employee", uniqueConstraints = {

@UniqueConstraint(columnNames = "ID"),

@UniqueConstraint(columnNames = "EMAIL") })

public class EmployeeEntity implements Serializable {

    private static final long serialVersionUID = -1798070786993154676L;

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    @Column(name = "ID", unique = true, nullable = false)

    private Integer employeeId;

    @Column(name = "EMAIL", unique = true, nullable = false, length = 100)

    private String email;

    @Column(name = "FIRST\_NAME", unique = false, nullable = false, length = 100)

    private String firstName;

    @Column(name = "LAST\_NAME", unique = false, nullable = false, length = 100)

    private String lastName;

    @OneToMany(cascade=CascadeType.ALL)

    @JoinColumn(name="EMPLOYEE\_ID")

    private Set<AccountEntity> accounts;

**Hibernate many to many mapping** is made between two entities where one can have relation with multiple other entity instances. For example, for a subscription service SubscriptionEntity and ReaderEntity can be two type of entities. Any subscription can have multiple readers, where a reader can subscribe to multiple subscriptions.

## Hibernate many to many mapping design

To demonstrate many to many mapping using hibernate annotations, we will associate two entities i.e. ReaderEntity and SubscriptionEntity.

Their database schema should look like this. Using these tables, any application can save multiple associations between readers and subscriptions.



## Owner entity

@Entity(name = "ReaderEntity")

@Table(name = "READER", uniqueConstraints = {

        @UniqueConstraint(columnNames = "ID"),

        @UniqueConstraint(columnNames = "EMAIL") })

public class ReaderEntity implements Serializable

{

    private static final long serialVersionUID = -1798070786993154676L;

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    @Column(name = "ID", unique = true, nullable = false)

    private Integer readerId;

    @Column(name = "EMAIL", unique = true, nullable = false, length = 100)

    private String email;

    @Column(name = "FIRST\_NAME", unique = false, nullable = false, length = 100)

    private String firstName;

    @Column(name = "LAST\_NAME", unique = false, nullable = false, length = 100)

    private String lastName;

    @ManyToMany(cascade=CascadeType.ALL)

    @JoinTable(name="READER\_SUBSCRIPTIONS", joinColumns={@JoinColumn(referencedColumnName="ID")}

                                        , inverseJoinColumns={@JoinColumn(referencedColumnName="ID")})

    private Set<SubscriptionEntity> subscriptions;

## Mapped entity

@Entity(name = "SubscriptionEntity")

@Table(name = "SUBSCRIPTION", uniqueConstraints = {

        @UniqueConstraint(columnNames = "ID")})

public class SubscriptionEntity implements Serializable

{

    private static final long serialVersionUID = -6790693372846798580L;

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    @Column(name = "ID", unique = true, nullable = false)

    private Integer subscriptionId;

    @Column(name = "SUBS\_NAME", unique = true, nullable = false, length = 100)

    private String subscriptionName;

    @ManyToMany(mappedBy="subscriptions")

    private Set<ReaderEntity> readers;

    //Getters and setters

}

5 why sessionfactory is threadsafe

* SessionFactory  is a factory to hibernate Session objects.
* SessionFactory is often built during start-up and used by application code to get session object.
* Java JEE application has just one SessionFactory, and individual threads, which are servicing client’s request obtain hibernate Session instances from this factory.
* The internal state of a SessionFactory is **immutable**.
* Most problems with concurrency occur due to sharing of objects with mutable state.
* Once the object is immutable, its internal state is settled on creation and cannot be changed. So many threads can access it concurrently and request for sessions.hence the SessionFactory   is thread-safe.

spring

1 why dependency injection came into the picture

**Dependency injection** is a powerful technique that can be applied in many situations across all layers of an application. But this does not mean that **dependency injection** should be used every time a class depends on another class. ... You need to **inject** the same **dependency** into multiple components.

To achieve loose coupling without touching code

To share single object to multiple components

2 what is ioc

**Inversion of Control(IoC) Container:**  
This is common characteristic of frameworks, IoC **manages java objects**  
- from instantiation to destruction through its BeanFactory.  
- Java components that are instantiated by the IoC container are called beans, and the **IoC container manages a bean's scope, lifecycle events, and any AOP features** for which it has been configured and coded.

**Inversion of control as a design guideline serves the following purposes:**  
- There is a decoupling of the execution of a certain task from implementation.  
- Every module can focus on what it is designed for.  
- Modules make no assumptions about what other systems do but rely on their contracts.  
- Replacing modules has no side effect on other modules

3 what are the scopes  
4 what is defference between beanfactory container and application container

5 flow of spring mvc  
6 what are the view resolver

Spring MVC Framework provides the ViewResolver interface, that maps view names to actual views.

It also provides the View interface, which addresses the request of a view to the view technology. So when a ModelAndView instance is returned by a Controller, the view resolver will resolve the view according to the view name.

1. **AbstractCachingViewResolver :** Abstract view resolver that caches views. Often views need preparation before they can be used; extending this view resolver provides caching.
2. **XmlViewResolver :**Implementation of ViewResolver that accepts a configuration file written in XML with the same DTD as Spring’s XML bean factories. The default configuration file is /WEB-INF/views.xml.
3. **ResourceBundleViewResolver :**Implementation of ViewResolver that uses bean definitions in a ResourceBundle, specified by the bundle base name. Typically you define the bundle in a properties file, located in the classpath. The default file name is views.properties.
4. **UrlBasedViewResolver :**Simple implementation of the ViewResolver interface that effects the direct resolution of logical view names to URLs, without an explicit mapping definition. This is appropriate if your logical names match the names of your view resources in a straightforward manner, without the need for arbitrary mappings.
5. **InternalResourceViewResolver :**Convenient subclass of UrlBasedViewResolver that supports InternalResourceView (in effect, Servlets and JSPs) and subclasses such as JstlView and TilesView. You can specify the view class for all views generated by this resolver by using setViewClass(..).
6. **VelocityViewResolver/FreeMarkerViewResolver :**Convenient subclass of UrlBasedViewResolver that supports VelocityView (in effect, Velocity templates) or FreeMarkerView ,respectively, and custom subclasses of them.
7. **ContentNegotiatingViewResolver :**Implementation of the ViewResolver interface that resolves a view based on the request file name or Accept header.

7 what is aop

AOP addresses the problem of *cross-cutting concerns*, which would be any kind of code that is repeated in different methods and can't normally be completely refactored into its own module, like with logging or verification.

(Or) it is a programming approach using this approach we can able to generate new class by mixing primary logic and secondary logic dynamically.

## **AOP Concepts and Terminology**

## **AOP Concepts and Terminology**

AOP concepts and terminologies are as follows:

* Join point
* Advice
* Pointcut
* Introduction
* Target Object
* Aspect
* Interceptor
* AOP Proxy
* Weaving

#### Join point

Join point is any point in your program such as method execution, exception handling, field access etc. Spring supports only method execution join point.

#### Advice

Advice represents an action taken by an aspect at a particular join point. There are different types of advices:

* **Before Advice**: it executes before a join point.
* **After Returning Advice**: it executes after a joint point completes normally.
* **After Throwing Advice**: it executes if method exits by throwing an exception.
* **After (finally) Advice**: it executes after a join point regardless of join point exit whether normally or exceptional return.
* **Around Advice**: It executes before and after a join point.

#### Pointcut

It is an expression language of AOP that matches join points.

#### Introduction

It means introduction of additional method and fields for a type. It allows you to introduce new interface to any advised object.

#### Target Object

It is the object i.e. being advised by one or more aspects. It is also known as proxied object in spring because Spring AOP is implemented using runtime proxies.

#### Aspect

It is a class that contains advices, joinpoints etc.

#### Interceptor

It is an aspect that contains only one advice.

#### AOP Proxy

It is used to implement aspect contracts, created by AOP framework. It will be a JDK dynamic proxy or CGLIB proxy in spring framework.

#### Weaving

It is the process of linking aspect with other application types or objects to create an advised object. Weaving can be done at compile time, load time or runtime. Spring AOP performs weaving at runtime.

### **AOP Implementations**

AOP implementations are provided by:

1. AspectJ
2. Spring AOP
3. JBoss AOP

# **Spring AOP Example**

1. [Before Advice Example](https://www.javatpoint.com/spring-aop-example)
2. [After Returning Advice Example](https://www.javatpoint.com/spring-aop-example)
3. [Around Advice Example](https://www.javatpoint.com/spring-aop-example)
4. [After Throwing Advice Example](https://www.javatpoint.com/spring-aop-example)

There are given examples of **Spring1.2 old style AOP** (dtd based) implementation.

Though it is supported in spring 3, but it is recommended to use spring aop with aspectJ that we are going to learn in next page.

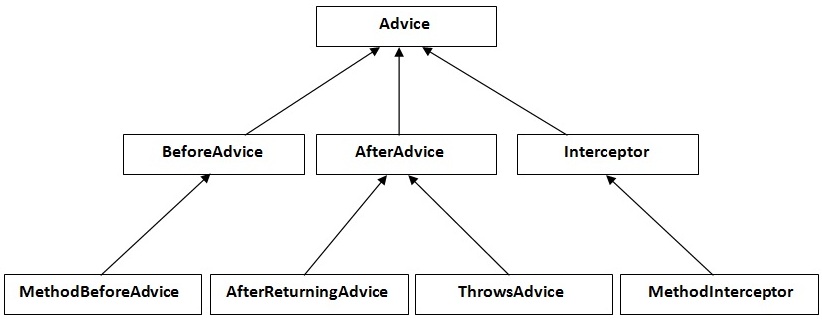
There are 4 types of advices supported in spring1.2 old style aop implementation.

1. **Before Advice** it is executed before the actual method call.
2. **After Advice** it is executed after the actual method call. If method returns a value, it is executed after returning value.
3. **Around Advice** it is executed before and after the actual method call.
4. **Throws Advice** it is executed if actual method throws exception.

#### To understand the basic concepts of Spring AOP, visit the previous page.

#### Understanding the hierarchy of advice interfaces

Let's understand the advice hierarchy by the diagram given below:



All are interfaces in aop.

**MethodBeforeAdvice** interface extends the **BeforeAdvice** interface.

**public** **void** before(Method method, Object[] args, Object target)**throws** Throwable {

<bean id="obj" **class**="com.javatpoint.A"></bean>

<bean id="ba" **class**="com.javatpoint.BeforeAdvisor"></bean>

<bean id="proxy" **class**="org.springframework.aop.framework.ProxyFactoryBean">

<property name="target" ref="obj"></property>

<property name="interceptorNames">

<list>

<value>ba</value>

</list>

</property>

</bean>

**AfterReturningAdvice** interface extends the **AfterAdvice** interface.

**public** **void** afterReturning(Object returnValue, Method method,

         Object[] args, Object target) **throws** Throwable {

**ThrowsAdvice** interface extends the **AfterAdvice** interface.

**public** **void** afterThrowing(Exception ex){

**MethodInterceptor** interface extends the **Interceptor** interface. It is used in around advice.

**public** Object invoke(MethodInvocation mi) **throws** Throwable {

# **Spring AOP AspectJ Annotation Example**

The **Spring Framework** recommends you to use **Spring AspectJ AOP implementation** over the Spring 1.2 old style dtd based AOP implementation because it provides you more control and it is easy to use.

[**next →**](https://www.javatpoint.com/spring-aop-aspectj-xml-configuration-example)[**← prev**](https://www.javatpoint.com/spring-aop-example)

# **Spring AOP AspectJ Annotation Example**

1. [@Before Example](https://www.javatpoint.com/spring-aop-aspectj-annotation-example)
2. [@After Example](https://www.javatpoint.com/spring-aop-aspectj-annotation-example)
3. [@AfterReturning Example](https://www.javatpoint.com/spring-aop-aspectj-annotation-example)
4. [@Around Example](https://www.javatpoint.com/spring-aop-aspectj-annotation-example)
5. [@AfterThrowing Example](https://www.javatpoint.com/spring-aop-aspectj-annotation-example)

The **Spring Framework** recommends you to use **Spring AspectJ AOP implementation** over the Spring 1.2 old style dtd based AOP implementation because it provides you more control and it is easy to use.

There are two ways to use Spring AOP AspectJ implementation:

1. By annotation: We are going to learn it here.
2. By xml configuration (schema based): We will learn it in next page.

#### To understand the aop concepts, its advantage etc. visit here [AOP Concepts Tutorial](http://www.javatpoint.com/spring-aop-tutorial)

[download all examples (developed using MyEclipse IDE)](https://static.javatpoint.com/src/sp/aopaspectjannotation.zip)

Spring AspectJ AOP implementation provides many annotations:

1. **@Aspect** declares the class as aspect.
2. **@Pointcut** declares the pointcut expression.

The annotations used to create advices are given below:

1. **@Before** declares the before advice. It is applied before calling the actual method.
2. **@After** declares the after advice. It is applied after calling the actual method and before returning result.
3. **@AfterReturning** declares the after returning advice. It is applied after calling the actual method and before returning result. But you can get the result value in the advice.
4. **@Around** declares the around advice. It is applied before and after calling the actual method.
5. **@AfterThrowing** declares the throws advice. It is applied if actual method throws exception.

## **Understanding Pointcut**

Pointcut is an expression language of Spring AOP.

The **@Pointcut** annotation is used to define the pointcut. We can refer the pointcut expression by name also. Let's see the simple example of pointcut expression.

It will be applied on all the methods of Operation class.

1. @Pointcut("execution(public Employee.set\*(..))")

@Aspect

**public** **class** TrackOperation{

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

**public** **void** k(){}//pointcut name

    @Before("k()")//applying pointcut on before advice

**public** **void** myadvice(JoinPoint jp)//it is advice (before advice)

    {

        System.out.println("additional concern");

        //System.out.println("Method Signature: "  + jp.getSignature());

in xml

<bean **class**="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCreator"></bean>

8 what are the autowire modes  
9 how to integrate with hibernate with spring, some times they ask you to write down xml file

<bean id="myDataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">

<property name="driverClassName" value="org.hsqldb.jdbcDriver"/>

<property name="url" value="jdbc:hsqldb:hsql://localhost:9001"/>

<property name="username" value="sa"/>

<property name="password" value=""/>

</bean>

<bean id="mySessionFactory" class="org.springframework.orm.hibernate3.

annotation.AnnotationSessionFactoryBean">

<property name="dataSource" ref="myDataSource" />

<property name="annotatedClasses">

<list>

<value>com.vaannila.domain.User</value>

</list>

</property>

<property name="hibernateProperties">

<props>

<prop key="hibernate.dialect"> org.hibernate.dialect.HSQLDialect</prop>

<prop key="hibernate.show\_sql">true</prop>

<prop key="hibernate.hbm2ddl.auto">create</prop>

</props>

</property>

</bean>

private HibernateTemplate hibernateTemplate;

public void setSessionFactory(SessionFactory sessionFactory)

{

this.hibernateTemplate = new HibernateTemplate(sessionFactory);

}

@Override

public void saveUser(User user) {

hibernateTemplate.saveOrUpdate(user);

}

web services

A **web service** is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a **web service**. For example, a client invokes a **web service** by sending an XML message, then waits for a corresponding XML response.

* Is available over the Internet or private (intranet) networks
* Uses a standardized XML messaging system
* Is not tied to any one operating system or programming language
* Is self-describing via a common XML grammar
* Is discoverable via a simple find mechanism

## **Components of Web Services**

The basic web services platform is XML + HTTP. All the standard web services work using the following components −

* SOAP (Simple Object Access Protocol)
* UDDI (Universal Description, Discovery and Integration)
* WSDL (Web Services Description Language)

All these components have been discussed in the [Web Services Architecture](https://www.tutorialspoint.com/webservices/web_services_architecture.htm) chapter.

## **How Does a Web Service Work?**

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP. A web service takes the help of −

* XML to tag the data
* SOAP to transfer a message
* WSDL to describe the availability of service.

what is rest

Representational state transfer is a software architectural style that defines a set of constraints to be used for creating Web services. Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the Internet.

**REST** is used to build Web **services** that are lightweight, maintainable, and scalable in nature. A **service** which is built on the **REST** architecture is called a **RESTful service**. The underlying protocol for **REST** is HTTP, which is the basic web protocol

what is deffernce between soap and rest,

**SOAP** - SOAP is a protocol which was designed before REST and came into the picture. The main idea behind designing SOAP was to ensure that programs built on different platforms and programming languages could exchange data in an easy manner.

**REST** - This was designed specifically for working with components such as media components, files, or even objects on a particular hardware device. Any web service that is defined on the principles of REST can be called a RestFul web service. A Restful service would use the normal HTTP verbs of GET, POST, PUT and DELETE for working with the required components.

Below are the main differences between SOAP and REST

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| * SOAP stands for Simple Object Access Protocol | * REST stands for Representational State Transfer |
| * SOAP is a protocol. SOAP was designed with a specification. It includes a WSDL file which has the required information on what the web service does in addition to the location of the web service. | * REST is an Architectural style in which a web service can only be treated as a RESTful service if it follows the constraints of being   1. Client Server   2. Stateless   3. Cacheable   4. Layered System   5. Uniform Interface |
| * SOAP cannot make use of REST since SOAP is a protocol and REST is an architectural pattern. | * REST can make use of SOAP as the underlying protocol for web services, because in the end it is just an architectural pattern. |
| * SOAP uses service interfaces to expose its functionality to client applications. In SOAP, the WSDL file provides the client with the necessary information which can be used to understand what services the web service can offer. | * REST use Uniform Service locators to access to the components on the hardware device. For example, if there is an object which represents the data of an employee hosted on a URL as http://demo.guru99 , the below are some of URI that can exist to access them   http://demo.guru99.com/Employee  http://demo.guru99.com/Employee/1 |
| * SOAP requires more bandwidth for its usage. Since SOAP Messages contain a lot of information inside of it, the amount of data transfer using SOAP is generally a lot.   <?xml version="1.0"?>  <SOAP-ENV:Envelope  xmlns:SOAP-ENV  ="http://www.w3.org/2001/12/soap-envelope"  SOAP-ENV:encodingStyle  =" http://www.w3.org/2001/12/soap-encoding">  <soap:Body>  <Demo.guru99WebService  xmlns="http://tempuri.org/">  <EmployeeID>int</EmployeeID>  </Demo.guru99WebService>  </soap:Body>  </SOAP-ENV:Envelope> | * REST does not need much bandwidth when requests are sent to the server. REST messages mostly just consist of JSON messages. Below is an example of a JSON message passed to a web server. You can see that the size of the message is comparatively smaller to SOAP.   {"city":"Mumbai","state":"Maharastra"} |
| * SOAP can only work with XML format. As seen from SOAP messages, all data passed is in XML format. | * REST permits different data format such as Plain text, HTML, XML, JSON, etc. But the most preferred format for transferring data is JSON. |

## When to use REST and when to use SOAP

One of the most highly debatable topics is when REST should be used or when to use SOAP while designing web services.

Below are some of the key factors that determine when each technology should be used for web services **REST services should be used in the following instances**

* **Limited resources and bandwidth** – Since SOAP messages are heavier in content and consume a far greater bandwidth, REST should be used in instances where network bandwidth is a constraint.
* **Statelessness** – If there is no need to maintain a state of information from one request to another then REST should be used. If you need a proper information flow wherein some information from one request needs to flow into another then SOAP is more suited for that purpose. We can take the example of any online purchasing site. These sites normally need the user first to add items which need to be purchased to a cart. All of the cart items are then transferred to the payment page in order to complete the purchase. This is an example of an application which needs the state feature. The state of the cart items needs to be transferred to the payment page for further processing.
* **Caching**– If there is a need to cache a lot of requests then REST is the perfect solution. At times, clients could request for the same resource multiple times. This can increase the number of requests which are sent to the server. By implementing a cache, the most frequent queries results can be stored in an intermediate location. So whenever the client requests for a resource, it will first check the cache. If the resources exist then, it will not proceed to the server. So caching can help in minimizing the amount of trips which are made to the web server.
* **Ease of coding**– Coding REST Services and subsequent implementation is far easier than SOAP. So if a quick win solution is required for web services, then REST is the way to go.

SOAP should be used in the following instances

1. **Asynchronous processing and subsequent invocation** – if there is a requirement that the client needs a guaranteed level of reliability and security then the new SOAP standard of SOAP 1.2 provides a lot of additional features, especially when it comes to security.
2. **A Formal means of communication** – if both the client and server have an agreement on the exchange format then SOAP 1.2 gives the rigid specifications for this type of interaction. An example is an online purchasing site in which users add items to a cart before the payment is made. Let's assume we have a web service that does the final payment. There can be a firm agreement that the web service will only accept the cart item name, unit price, and quantity. If such a scenario exists then, it's always better to use the SOAP protocol.
3. **Stateful operations –**ifthe application has a requirement that state needs to be maintained from one request to another, then the SOAP 1.2 standard provides the WS\* structure to support such requirements.

## SOAP vs. REST API challenges

API is known as the **Application Programming Interface** and is offered by both the client and the server. In the client world, this is offered by the browser whereas in the server world it's what is provided by the web service which can either be SOAP or REST.

**Challenges with the SOAP API**

1. WSDL file - One of the key challenges of the SOAP API is the WSDL document itself. The WSDL document is what tells the client of all the operations that can be performed by the web service. The WSDL document will contain all information such as the data types being used in the SOAP messages and what all operations are available via the web service. The below code snippet is just part of a sample WSDL file.

<?xml version="1.0"?>

<definitions name="Tutorial"

targetNamespace=http://demo.guru99.com/Tutorial.wsdl

xmlns:tns=http://demo.guru99.com/Tutorial.wsdl

xmlns:xsd1=http://demo.guru99.com/Tutorial.xsd

xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/

xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>

<schema targetNamespace=http://Demo.guru99.com/Tutorial.xsd

xmlns="http://www.w3.org/2000/10/XMLSchema">

<element name="TutorialNameRequest">

<complexType>

<all>

<element name="TutorialName" type="string"/>

</all>

</complexType>

</element>

<element name="TutorialIDRequest">

<complexType>

<all>

<element name="TutorialID" type="number"/>

</all>

</complexType>

</element>

</schema>

</types>

As per the above WSDL file, we have an element called "TutorialName" which is of the type String which is part of the element TutorialNameRequest.

Now, suppose if the WSDL file were to change as per the business requirements and the TutorialName has to become TutorialDescription. This would mean that all the clients who are currently connecting to this web service would then need to make this corresponding change in their code to accommodate the change in the WSDL file.

This shows the biggest challenge of the WSDL file which is the tight contract between the client and the server and that one change could cause a large impact, on the whole, client applications.

1. Document size – The other key challenge is the size of the SOAP messages which get transferred from the client to the server. Because of the large messages, using SOAP in places where bandwidth is a constraint can be a big issue.

**Challenges with the REST API**

1. **Lack of Security**– REST does not impose any sort of security like SOAP. This is why REST is very appropriate for public available URL's, but when it comes down to confidential data being passed between the client and the server, REST is the worst mechanism to be used for web services.
2. **Lack of state**– Most web applications require a stateful mechanism. For example, if you had a purchasing site which had the mechanism of having a shopping cart, it is required to know the number of items in the shopping cart before the actual purchase is made. Unfortunately, the burden of maintaining this state lies with the client, which just makes the client application heavier and difficult to maintain.

what are annotations in rest and their works

### @GET

Annotate your Get request methods with @GET.

|  |  |
| --- | --- |
| 1  2  3  4 | @GET  public String getHTML() {    ...  } |

### @Produces

@Produces annotation specifies the type of output this method (or web service) will produce.

|  |  |
| --- | --- |
| 1  2  3  4  5 | @GET  @Produces("application/xml")  public Contact getXML() {    ...  } |
| 1  2  3  4  5 | @GET  @Produces("application/json")  public Contact getJSON() {    ...  } |

### @Path

@Path annotation specify the URL path on which this method will be invoked.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | @GET  @Produces("application/xml")  @Path("xml/{firstName}")  public Contact getXML() {    ...  } |

### @PathParam

We can bind REST-style URL parameters to method arguments using @PathParam annotation as shown below.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | @GET  @Produces("application/xml")  @Path("xml/{firstName}")  public Contact getXML(@PathParam("firstName") String firstName) {    Contact contact = contactService.findByFirstName(firstName);    return contact;  } |
| 1  2  3  4  5  6  7 | @GET  @Produces("application/json")  @Path("json/{firstName}")  public Contact getJSON(@PathParam("firstName") String firstName) {    Contact contact = contactService.findByFirstName(firstName);    return contact;  } |

### @QueryParam

Request parameters in query string can be accessed using @QueryParam annotation as shown below.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | @GET  @Produces("application/json")  @Path("json/companyList")  public CompanyList getJSON(@QueryParam("start") int start, @QueryParam("limit") int limit) {    CompanyList list = new CompanyList(companyService.listCompanies(start, limit));    return list;  } |

The example above returns a list of companies (with server side pagination) which can be displayed with rich clients implemented using Ext-js or jQuery. You can read more more about setting up [ExtJS grid panel with remote sorting and pagination using Hibernate](http://blog.techferry.com/2012/01/25/extjs-grid-panel-with-remote-sorting-and-pagination-using-hibernate/).

### @POST

Annotate POST request methods with @POST.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | @POST  @Consumes("application/json")  @Produces("application/json")  public RestResponse<Contact> create(Contact contact) {  ...  } |

### @Consumes

The @Consumes annotation is used to specify the MIME media types a REST resource can consume.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | @PUT  @Consumes("application/json")  @Produces("application/json")  @Path("{contactId}")  public RestResponse<Contact> update(Contact contact) {  ...  } |

### @FormParam

The REST resources will usually consume XML/JSON for the complete Entity Bean. Sometimes, you may want to read parameters sent in POST requests directly and you can do that using @FormParam annotation. GET Request query parameters can be accessed using [@QueryParam](http://www.techferry.com/articles/RESTful-web-services-JAX-RS-annotations.html#QueryParam) annotation.

|  |  |
| --- | --- |
| 1  2  3  4  5 | @POST  public String save(@FormParam("firstName") String firstName,      @FormParam("lastName") String lastName) {        ...    } |

### @PUT

Annotate PUT request methods with @PUT.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | @PUT  @Consumes("application/json")  @Produces("application/json")  @Path("{contactId}")  public RestResponse<Contact> update(Contact contact) {  ...  } |

### @DELETE

Annotate DELETE request methods with @DELETE.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | @DELETE  @Produces("application/json")  @Path("{contactId}")  public RestResponse<Contact> delete(@PathParam("contactId") int contactId) {  ...  } |

what is advantage from soap and disadvantage of soap

### advantages of SOAP Web Services?

SOAP web services have all the advantages that web services has, some of the additional advantages are:

* + WSDL document provides contract and technical details of the web services for client applications without exposing the underlying implementation technologies.
  + SOAP uses XML data for payload as well as contract, so it can be easily read by any technology.
  + SOAP protocol is universally accepted, so it’s an industry standard approach with many easily available open source implementations.

### disadvantages of SOAP Web Services?

Some of the disadvantages of SOAP protocol are:

* + Only XML can be used, JSON and other lightweight formats are not supported.
  + SOAP is based on the contract, so there is a tight coupling between client and server applications.
  + SOAP is slow because payload is large for a simple string message, since it uses XML format.
  + Anytime there is change in the server side contract, client stub classes need to be generated again.
  + Can’t be tested easily in browser

. friends these are commonly asked questions in regular interview. instead of by chosing these question it is far better to learn in depth. but i am sure no quetions can come out of nataraz sir's meterial.

and one.more thing when your going to interview dont go with heavy make up. just go how your going daily to class

just go in cool mind. if you dont know any answer at all. say in positively i dont know but if you give me a chance i will learn.

but sure when your going to attend intervie make sure that remember nataraz sir classes. and go in cool mind.

Last 2 Months Java Interview Questions  
---------------------------------------------------------

1.what is OOPs feature?How to implement OOPs features?  
2.How to achieve Encapsulation ,Abstraction,Polymorphism and Inheritance.How to implement these programmatically with Realtime usecases?  
3.what is method overloading and method overriding?(some combinations)  
4.what is difference between Abstract class and Interface?

1. Main difference is methods of a Java interface are implicitly abstract and cannot have implementations. A Java abstract class can have instance methods that implements a default behavior.
2. Variables declared in a Java interface is by default final. An  abstract class may contain non-final variables.
3. Members of a Java interface are public by default. A Java abstract class can have the usual flavors of class members like private, protected, etc..
4. Java interface should be implemented using keyword “implements”; A Java abstract class should be extended using keyword “extends”.
5. An interface can extend another Java interface only, an abstract class can extend another Java class and implement multiple Java interfaces.
6. A Java class can implement multiple interfaces but it can extend only one abstract class.
7. Interface is absolutely abstract and cannot be instantiated; A Java abstract class also cannot be instantiated, but can be invoked if a main() exists.
8. In comparison with java abstract classes, java interfaces are slow as it requires extra indirection.

5.After After java 8,what is difference between Abstract class and Interface?

With the introduction of concrete methods (default and static methods) to interfaces from Java 8, the gap between interface and abstract class has been reduced significantly. Now both can have concrete methods as well as abstract methods. But, still there exist some minute differences between them. In this article, we will try to list down the differences between interface Vs abstract class after Java 8.

### Differences Between Interface And Abstract Class After Java 8 :

#### 1) Fields

Interface fields are public, static and final by default. Interfaces still don’t support non-static and non-final variables. Interfaces can only have public, static and final variables. On the other hand, abstract class can have static as well as non-static and final as well as non-final variables. They also support private and protected variables along with public variables.

#### 2) Methods

After Java 8, an interface can have default and static methods along with abstract methods. Interfaces don’t support final methods. But, abstract classes support final as well as non-final methods and static as well as non-static methods along with abstract methods.

Also note that, only interfaces can have default methods. Abstract classes can’t have default methods.

#### 3) Constructors

Interfaces can’t have constructors. Abstract classes can have any number of constructors.

#### 4) Member’s Accessibility

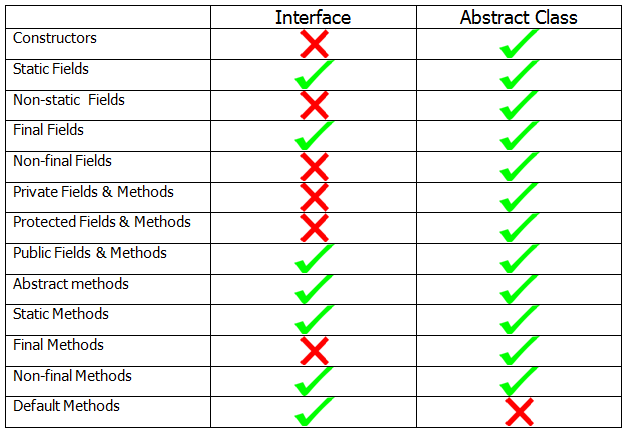
All members of interfaces are public by default. Interfaces don’t support private and protected members. But, abstract classes support all type of members – private, protected and public members.

#### 5) Multiple Inheritance

A class can extend only one abstract class, but can implement multiple interfaces. Thus, a class can inherit multiple properties from multiple sources only through interfaces, not through abstract classes.

### Interface Vs Abstract Class After Java 8 :

The below table and program summarizes the similarities and differences between interface and abstract class after Java 8.

[](https://i0.wp.com/javaconceptoftheday.com/wp-content/uploads/2019/04/InterfaceVsAbstractClassJava8.png?ssl=1)

6.why Interface is providing default and static methods?

Java **interface static method** helps us in providing security by not allowing implementation classes to override them. ... This **is** because it's not allowed in java, since Object **is** the base class for all the classes and **we can**'t **have** one class level **static method** and another instance **method** with same signature.

**Default methods** enable us to add new functionality to existing **interfaces** without breaking older implementation of these **interfaces**. When we extend an **interface** that contains a **default method**, we can perform following, Not override the **default method** and will inherit the **default method.**

7.what is Association and Aggregation and difference also?  
8.what is immutable?why String is immutable?  
9.How to make class is immutable?when we are taking attribute as Object type like-Date,Employee,then how to make class as immutable?  
10.What is difference between String,StringBuffer and StringBuilder?  
11.what is the Exception?What is checked and Unchecked Exceptions with Example?and difference also?

12.What is Custom Exception?How to make the class as Custom Exception class?  
13.How to Handle the Exceptions?What is the keywords are there in Exception class?and

Java exception handling is managed via five keywords: **try**, **catch**, [**throw**](https://www.geeksforgeeks.org/throw-throws-java/), [**throws**](https://www.geeksforgeeks.org/throw-throws-java/), and **finally**. Briefly, here is how they work. Program statements that you think can raise exceptions are contained within a try block. If an exception occurs within the try block, it is thrown. Your code can catch this exception (using catch block) and handle it in some rational manner. System-generated exceptions are automatically thrown by the Java run-time system. To manually throw an exception, use the keyword [throw](https://www.geeksforgeeks.org/throw-throws-java/). Any exception that is thrown out of a method must be specified as such by a [throws](https://www.geeksforgeeks.org/throw-throws-java/) clause. Any code that absolutely must be executed after a try block completes is put in a finally block.  
All the combinations of try-catch-finally?

**Various possible combinations of try- catch- finally?**

1. Whenever we are writing try block compulsory we should write catch or finally that is ‘try’ without catch or finally is invalid syntax.

2. Whenever we are writing catch block compulsory we should write try block that is catch without try is invalid.

3. Whenever we are writing finally block compulsory we should write try block. that is finally without try is invalid.

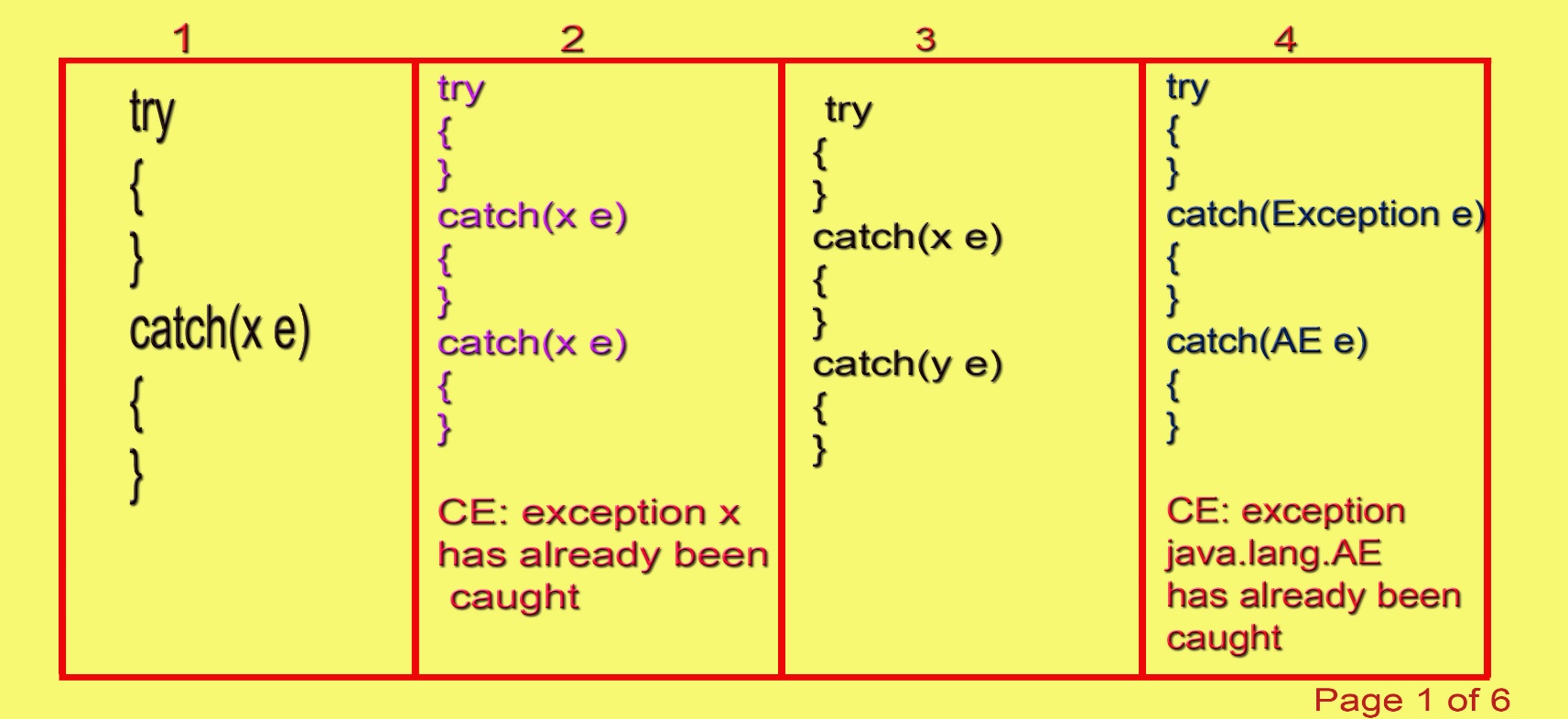
4. In try catch finally, order is important.

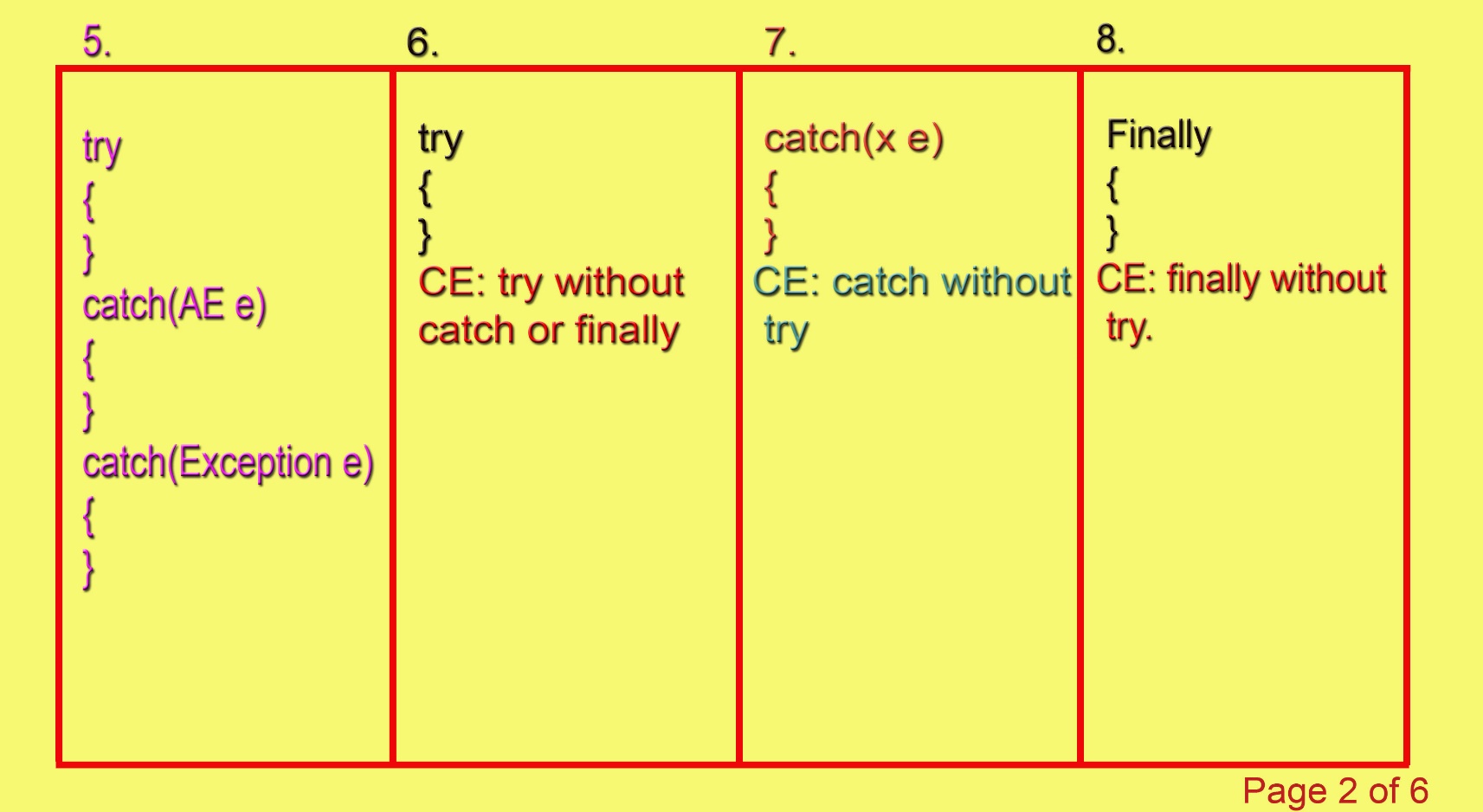
5. ‘try’ with multiple catch blocks Is valid but the order is important compulsory we should take from child to parent. by mistake if we are trying to take from parent to child then we will get compile time error.

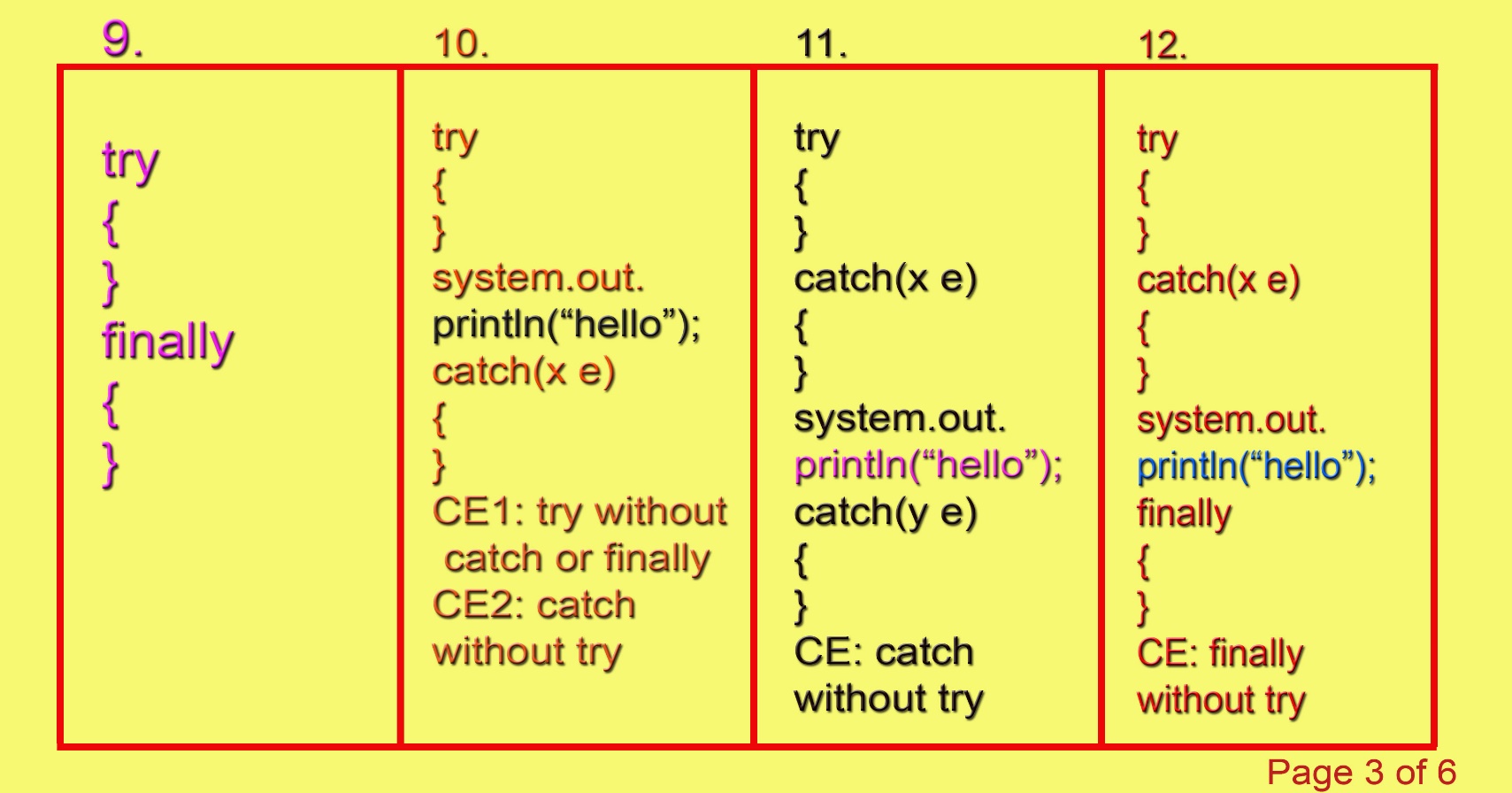
6. if we are defining to catch blocks for the same exception we will get compile time error.

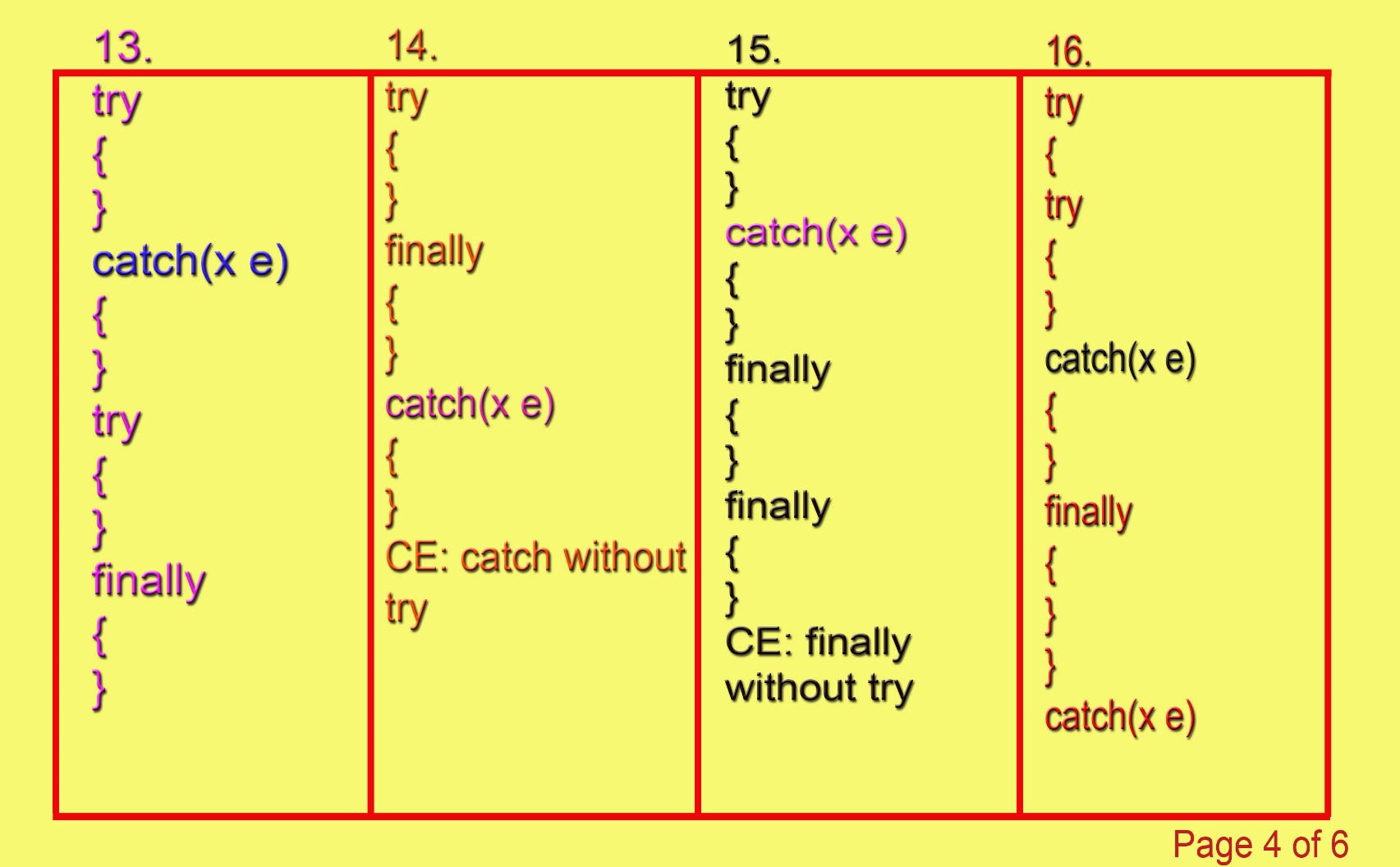
7. we can define try-catch-finally with in the try, with in the catch and with in finally blocks. Hence nesting of try-catch-finally is valid.

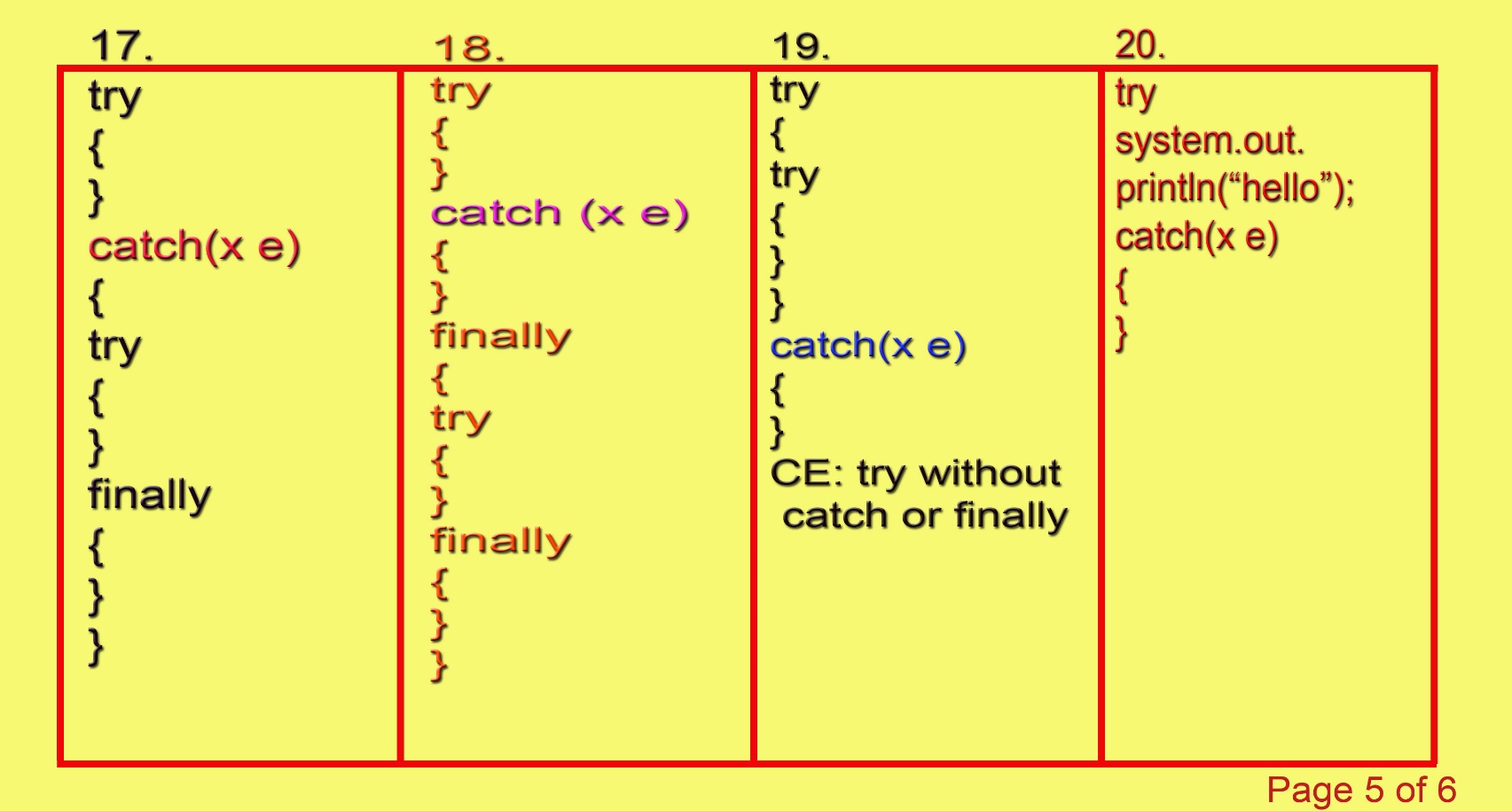
8. For try-catch-finally curly braces are mandatory.

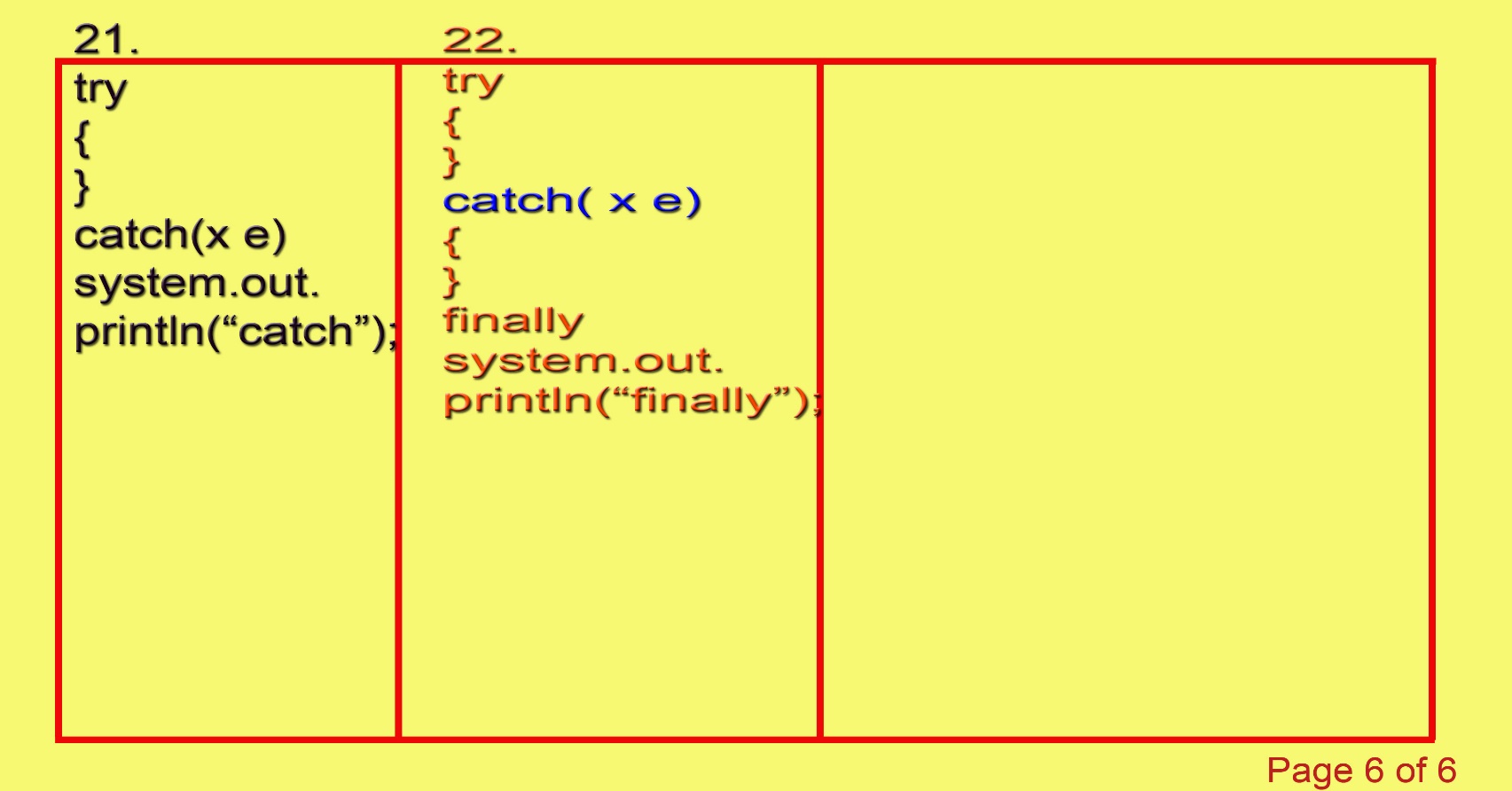












14.what is the output?  
public int fun(){  
try{  
return 1;  
}  
catch(Exception e){  
return 2;  
}  
finally{  
return 3;  
}  
}  
15.What is Serializable.How to make the class as Serialiazable?  
16.What is marker interface?

 A **marker interface** is an **interface** that has no methods or constants inside it. It provides run-time type information about objects, so the compiler and JVM have additional information about the object. A **marker interface** is also called a tagging **interface**.

17.what is enum?Example of enum?

Enums are lists of constants like unchangeable variables. Have you heard of Final keyword? It’s like that.

When you need a predefined list of values which do represent some kind of numeric or textual data, you should use an enum. For [instance](https://crunchify.com/lazy-creation-of-singleton-threadsafe-instance-without-using-synchronized-keyword/), in a chess game you could represent the different types of pieces as an enum:

enum Foo { ONE, TWO, THREE}

18.What is the Collection?It’s data Structures ? Where to use List and Set and Map and Properties in Project

The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects. ... **Java Collection** framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

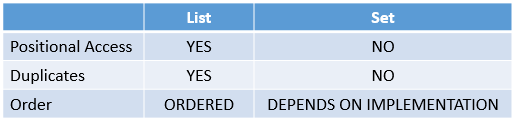
A data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently. Data structures provide a means to manage large amounts of data efficiently. efficient data structures are a key to designing efficient algorithms.

#### List of Data Structures

1. [Stacks](https://www.java2novice.com/data-structures-in-java/stacks/)
2. [Queue](https://www.java2novice.com/data-structures-in-java/queue/)
3. [Linked List](https://www.java2novice.com/data-structures-in-java/linked-list/)

19.Difference between List and Set?

List is an **ordered sequence** of elements whereas Set is a **distinct list** of elements which is unordered



List is a type of ordered collection that maintains the elements in **insertion order** while Set is a type of unordered collection so elements are not maintained any order.

List allows duplicates while Set doesn't allow **duplicate elements** . All the elements of a Set should be unique if you try to insert the duplicate element in Set it would replace the existing value.

List permits any number of **null values** in its collection while Set permits only one null value in its collection.

New methods are defined inside **List interface** . But, no new methods are defined inside Set interface, so we have to use Collection interface methods only with **Set subclasses** .

List can be inserted in in both **forward** direction and **backward** direction using Listiterator while Set can be traversed only in forward direction with the help of iterator

## When to use

List is an ordered sequence of elements, however set is distinct list of element which is unordered. So, use list for storing **non-unique objects** as per insertion order and use set for storing **unique objects** in random order

## Java collections framework

Java **Collections** are predefined set of classes or data structures which can be used to store multiple items in a single unit. **Dynamically allocated**data structures in Java (such as Hashtable, HashSet, HashMap,LinkedList, Vector, Stack, ArrayList) are supported in a unified architecture called the Collection Framework, which mandates the common behaviours of all the classes. More about.... [Java Collection Framework](http://net-informations.com/java/cjava/list.htm)

20.Difference between ArrayList and LinkedList?

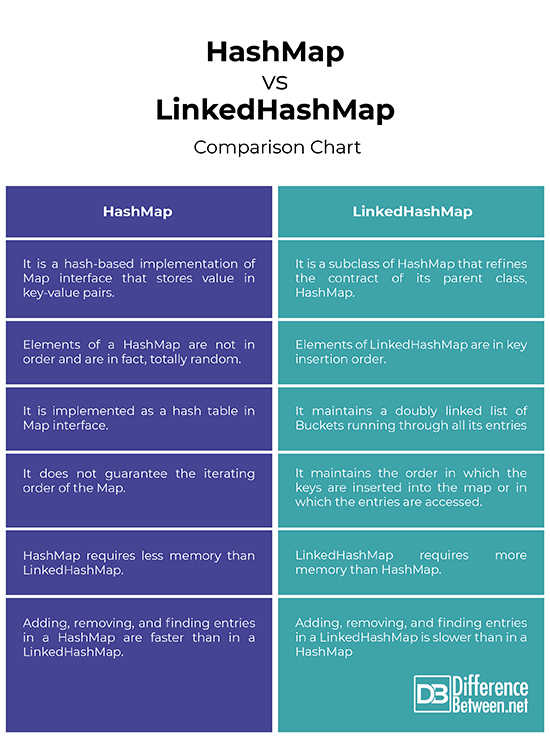
# **Difference between ArrayList and LinkedList**

ArrayList and LinkedList both implements List interface and maintains insertion order. Both are non synchronized classes.

However, there are many differences between ArrayList and LinkedList classes that are given below.

|  |  |
| --- | --- |
| **ArrayList** | **LinkedList** |
| 1) ArrayList internally uses a **dynamic array** to store the elements. | LinkedList internally uses a **doubly linked list**   to store the elements. |
| 2) Manipulation with ArrayList is **slow** because it internally uses an array. If any element is removed from the array, all the bits are shifted in memory. | Manipulation with LinkedList is **faster** than  ArrayList because it uses a doubly linked list,  so no bit shifting is required in memory. |
| 3) An ArrayList class can **act as a list** only because it implements List only. | LinkedList class can **act as a list and queue**   both because it implements List and Deque  interfaces. |
| 4) ArrayList is **better for storing and accessing** data. | LinkedList is **better for manipulating** data. |

21.what is Map?Difference between HashMap and LinkedHashmap?

  
22.How to make map as Synchronized map?

[HashMap is a non-synchronized collection class](https://beginnersbook.com/2013/12/hashmap-in-java-with-example/). If we need to perform thread-safe operations on it then we must need to synchronize it explicitly. In this tutorial we will see how to synchronize HashMap.

#### Example:

In this example we have a HashMap<Integer, String> it is having integer keys and String type values. In order to synchronize it we are using [Collections.synchronizedMap(hashmap)](https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#synchronizedMap(java.util.Map))  it returns a thread-safe map backed up by the specified HashMap.

Map map = Collections.synchronizedMap(new HashMap());

23.Internal of Map and Set?How to put and get is working?  
24.What is equals() and hashcode()?Example with code?  
23.what is the difference between == and ===?

# **Difference between == and === operator in JavaScript**

The ‘==’ operator tests for abstract equality i.e. it **does** the necessary type conversions before doing the equality comparison.  
But the ‘===’ operator tests for strict equality i.e it will **not** do the type conversion hence if the two values are not of the same type, when compared, it will return false.

Example 1:

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| <script>      // In R.H.S. string "3" is converted into      // number 3, hence returns true.      document.write(9 == "9");        // used for next line      document.write('<br>')        // Here no type conversion takes place,      // hence returns false      document.write(9 === "9"); |

24.When we get ConcurrentModificationException in Map?and How to resolve it?

java.util.ConcurrentModificationException is a very common exception when working with [Java collection](https://www.journaldev.com/1260/collections-in-java-tutorial) classes. Java Collection classes are fail-fast, which means if the Collection will be changed while some thread is traversing over it using iterator, the iterator.next() will throw **ConcurrentModificationException**. Concurrent modification exception can come in case of multithreaded as well as a single threaded java programming environment.

### To Avoid ConcurrentModificationException in multi-threaded environment

1. You can convert the list to an array and then iterate on the array. This approach works well for small or medium size list but if the list is large then it will affect the performance a lot.
2. You can lock the list while iterating by putting it in a synchronized block. This approach is not recommended because it will cease the benefits of [multithreading](https://www.journaldev.com/1079/multithreading-in-java).
3. If you are using JDK1.5 or higher then you can use **ConcurrentHashMap** and **CopyOnWriteArrayList** classes. This is the recommended approach to avoid concurrent modification exception.

25.What is MultiThreading?How to many ways to create the Thread with Example?

Threads can be created mainly in 3 different ways

1. **Extend** the java.lang.**Thread** class'

**2. Implement** the java.lang.**Runnable** interface

**3.Implement** the java.util.concurrent.**Callable** interface

class Counter implements Callable {

private static final int THREAD\_POOL\_SIZE = 2;

// method where the thread execution takes place

public String call() {

return Thread.currentThread().getName() + " executing ...";

}

26.Which case we have to go with Implements from Runnable interface and which case we have to go extending from Thread class?

If **you** want **to implements or extends** any other class then **Runnable** interface **is** most preferable, otherwise, if **you do** not want any other class **to extend or implement** then **Thread** class **is** preferable. When **you extends Thread** class, after **that you can**'t **extend** any other class **which you** required.

27.what is Synchronized?how to many types of locks are there?

28.What is the difference between synchronized method and block?  
29.Producer and consumer problem with Example in Thread?

The producer-consumer problem (also known as the bounded-buffer problem) is a classic [Java Example](https://crunchify.com/in-java-8-how-to-iterate-through-java-util-map-and-java-util-list-example-attached-with-total-5-different-ways/) of a multi-process [synchronization](https://crunchify.com/java-synchronized-collections-examples-java-util-collections/) problem.

The problem describes two [processes](https://crunchify.com/how-to-get-a-list-of-current-open-processes-with-java/), the producer and the consumer, who share a common, fixed-size buffer used as a [queue](https://crunchify.com/arrayblockingqueue-vs-google-guava-non-blocking-evictingqueue-example/). The producer’s job is to generate a piece of data, put it into the buffer and start again.

#### ProducerConsumer.java

public class ProducerConsumer  
{  
      public static void main(String[] args)  
      {  
            Shop c = new Shop();  
            Producer p1 = new Producer(c, 1);  
            Consumer c1 = new Consumer(c, 1);  
            p1.start();  
            c1.start();  
      }  
}  
class Shop  
{  
      private int materials;  
      private boolean available = false;  
      public synchronized int get()  
      {  
            while (available == false)  
            {  
                  try  
                  {  
                        wait();  
                  }  
                  catch (InterruptedException ie)  
                  {  
                  }  
            }  
            available = false;  
            notifyAll();  
            return materials;  
      }  
      public synchronized void put(int value)  
      {  
            while (available == true)  
            {  
                  try  
                  {  
                        wait();  
                  }  
                  catch (InterruptedException ie)  
                  {  
                        ie.printStackTrace();  
                  }  
            }  
            materials = value;  
            available = true;  
            notifyAll();  
      }  
}  
class Consumer extends Thread  
{  
      private Shop Shop;  
      private int number;  
      public Consumer(Shop c, int number)  
      {  
            Shop = c;  
            this.number = number;  
      }  
      public void run()  
      {  
            int value = 0;  
            for (int i = 0; i < 10; i++)  
            {  
                  value = Shop.get();  
                  System.out.println("Consumed value " + this.number+ " got: " + value);  
            }  
      }  
}  
class Producer extends Thread  
{  
      private Shop Shop;  
      private int number;  
  
      public Producer(Shop c, int number)  
      {  
            Shop = c;  
            this.number = number;  
      }  
      public void run()  
      {  
            for (int i = 0; i < 10; i++)  
            {  
                  Shop.put(i);  
                  System.out.println("Produced value " + this.number+ " put: " + i);  
                  try  
                  {  
                        sleep((int)(Math.random() \* 100));  
                  }  
                  catch (InterruptedException ie)  
                  {  
                        ie.printStackTrace();  
                  }  
            }  
      }  
}

30.Difference between sleep() and wait()?  
31.Difference between yield() and join()?  
32.Why wait() method is available in Object class?  
33.What is ThreadLocal with Example?

34.What is Comparable and Comparator?Difference and Example alse  
Usecase:  
Class Employee{  
Private int id;  
Private String name;  
Private float salary;  
}  
We want to sort the List on the basic of Name by using comparator?  
JEE  
35.what is JDBC API?

Java Database Connectivity (**JDBC**) is an application programming interface (**API**) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation.

36.What is difference between Statement and PreparedStatement?

## **1) Statement**

Statement interface is used to execute normal SQL queries. You can’t pass the parameters to SQL query at run time using this interface. This interface is preferred over other two interfaces if you are executing a particular SQL query only once. The performance of this interface is also very less compared to other two interfaces. In most of time, Statement interface is used for DDL statements like **CREATE**, **ALTER**, **DROP** etc. For example,

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | //Creating The Statement Object    Statement stmt = con.createStatement();    //Executing The Statement    stmt.executeUpdate("CREATE TABLE STUDENT(ID NUMBER NOT NULL, NAME VARCHAR)"); |

## **2) PreparedStatement**

PreparedStatement is used to execute dynamic or parameterized SQL queries. PreparedStatement extends Statement interface. You can pass the parameters to SQL query at run time using this interface. It is recommended to use PreparedStatement if you are executing a particular SQL query multiple times. It gives better performance than Statement interface. Because, PreparedStatement are precompiled and the query plan is created only once irrespective of how many times you are executing that query. This will save lots of time.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | //Creating PreparedStatement object    PreparedStatement pstmt = con.prepareStatement("update STUDENT set NAME = ? where ID = ?");    //Setting values to place holders using setter methods of PreparedStatement object    pstmt.setString(1, "MyName");   //Assigns "MyName" to first place holder    pstmt.setInt(2, 111);     //Assigns "111" to second place holder    //Executing PreparedStatement    pstmt.executeUpdate(); |

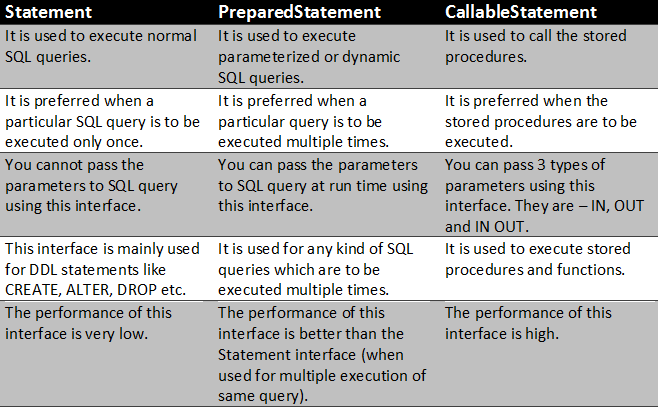
## **3) CallableStatement**

CallableStatement is used to execute the stored procedures. CallableStatement extends PreparedStatement. Usng CallableStatement, you can pass 3 types of parameters to stored procedures. They are : ***IN*** – used to pass the values to stored procedure, **OUT** – used to hold the result returned by the stored procedure and **IN OUT** – acts as both IN and OUT parameter. Before calling the stored procedure, you must register OUT parameters using **registerOutParameter()** method of CallableStatement. The performance of this interface is higher than the other two interfaces. Because, it calls the stored procedures which are already compiled and stored in the database server.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | //Creating CallableStatement object    CallableStatement cstmt = con.prepareCall("{call anyProcedure(?, ?, ?)}");    //Use cstmt.setter() methods to pass IN parameters    //Use cstmt.registerOutParameter() method to register OUT parameters    //Executing the CallableStatement    cstmt.execute();    //Use cstmt.getter() methods to retrieve the result returned by the stored procedure |

## Statement Vs PreparedStatement Vs CallableStatement In Java :

|  |  |  |
| --- | --- | --- |
| **Statement** | **PreparedStatement** | **CallableStatement** |
| It is used to execute normal SQL queries. | It is used to execute parameterized or dynamic SQL queries. | It is used to call the stored procedures. |
| It is preferred when a particular SQL query is to be executed only once. | It is preferred when a particular query is to be executed multiple times. | It is preferred when the stored procedures are to be executed. |
| You cannot pass the parameters to SQL query using this interface. | You can pass the parameters to SQL query at run time using this interface. | You can pass 3 types of parameters using this interface. They are – IN, OUT and IN OUT. |
| This interface is mainly used for DDL statements like CREATE, ALTER, DROP etc. | It is used for any kind of SQL queries which are to be executed multiple times. | It is used to execute stored procedures and functions. |
| The performance of this interface is very low. | The performance of this interface is better than the Statement interface (when used for multiple execution of same query). | The performance of this interface is high. |



37.What is SQL Injection Problem and How to Resolve it?

38.How to configure datasource?

DataSource ds = (DataSource) org.apache.derby.jdbc.ClientDataSource()

ds.setPort(1527);

ds.setHost("localhost");

ds.setUser("APP")

ds.setPassword("APP");

Connection con = ds.getConnection();

or

BasicDataSource dataSource = new BasicDataSource();

dataSource.setDriverClassName("com.mysql.jdbc.Driver");

dataSource.setUsername("username");

dataSource.setPassword("password");

dataSource.setUrl("jdbc:mysql://<host>:<port>/<database>");

dataSource.setMaxActive(10);

dataSource.setMaxIdle(5);

dataSource.setInitialSize(5);

dataSource.setValidationQuery("SELECT 1");

39.Servlets and JSP life cyle?

The following are the paths followed by a JSP −

* Compilation
* Initialization
* Execution
* Cleanup

The four major phases of a JSP life cycle are very similar to the Servlet Life Cycle. The four phases have been described below −



## **JSP Compilation**

When a browser asks for a JSP, the JSP engine first checks to see whether it needs to compile the page. If the page has never been compiled, or if the JSP has been modified since it was last compiled, the JSP engine compiles the page.

The compilation process involves three steps −

* Parsing the JSP.
* Turning the JSP into a servlet.
* Compiling the servlet.

## **JSP Initialization**

When a container loads a JSP it invokes the **jspInit()** method before servicing any requests. If you need to perform JSP-specific initialization, override the **jspInit()** method −

public void jspInit(){

// Initialization code...

}

Typically, initialization is performed only once and as with the servlet init method, you generally initialize database connections, open files, and create lookup tables in the jspInit method.

## **JSP Execution**

This phase of the JSP life cycle represents all interactions with requests until the JSP is destroyed.

Whenever a browser requests a JSP and the page has been loaded and initialized, the JSP engine invokes the **\_jspService()** method in the JSP.

The \_jspService() method takes an **HttpServletRequest** and an **HttpServletResponse** as its parameters as follows −

void \_jspService(HttpServletRequest request, HttpServletResponse response) {

// Service handling code...

}

The **\_jspService()** method of a JSP is invoked on request basis. This is responsible for generating the response for that request and this method is also responsible for generating responses to all seven of the HTTP methods, i.e, **GET, POST, DELETE**, etc.

## **JSP Cleanup**

The destruction phase of the JSP life cycle represents when a JSP is being removed from use by a container.

The **jspDestroy()** method is the JSP equivalent of the destroy method for servlets. Override jspDestroy when you need to perform any cleanup, such as releasing database connections or closing open files.

The jspDestroy() method has the following form −

public void jspDestroy() {

// Your cleanup code goes here.

}

40.Difference between ServletConfig and ServletContext.  
41.Implicit object of JSP?whta is JSTL tag library with Example?

|  |  |
| --- | --- |
| **S.No.** | **Object & Description** |
| 1 | **request**  This is the **HttpServletRequest** object associated with the request. |
| 2 | **response**  This is the **HttpServletResponse** object associated with the response to the client. |
| 3 | **out**  This is the **PrintWriter** object used to send output to the client. |
| 4 | **session**  This is the **HttpSession** object associated with the request. |
| 5 | **application**  This is the **ServletContext** object associated with the application context. |
| 6 | **config**  This is the **ServletConfig** object associated with the page. |
| 7 | **pageContext**  This encapsulates use of server-specific features like higher performance **JspWriters**. |
| 8 | **page**  This is simply a synonym for **this**, and is used to call the methods defined by the translated servlet class. |
| 9 | **Exception**  The **Exception** object allows the exception data to be accessed by designated JSP. |

The JavaServer Pages Standard Tag Library (JSTL) is a collection of useful JSP tags which encapsulates the core functionality common to many JSP applications.

JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating the existing custom tags with the JSTL tags.

## **Install JSTL Library**

To begin working with JSP tages you need to first install the JSTL library. If you are using the Apache Tomcat container, then follow these two steps −

**Step 1** − Download the binary distribution from [Apache Standard Taglib](https://tomcat.apache.org/taglibs/index.html) and unpack the compressed file.

**Step 2** − To use the Standard Taglib from its **Jakarta Taglibs distribution**, simply copy the JAR files in the distribution's 'lib' directory to your application's **webapps\ROOT\WEB-INF\lib** directory.

To use any of the libraries, you must include a <taglib> directive at the top of each JSP that uses the library.

## **Classification of The JSTL Tags**

The JSTL tags can be classified, according to their functions, into the following JSTL tag library groups that can be used when creating a JSP page −

* **Core Tags**
* **Formatting tags**
* **SQL tags**
* **XML tags**
* **JSTL Functions**

## **Core Tags**

The core group of tags are the most commonly used JSTL tags. Following is the syntax to include the JSTL Core library in your JSP −

<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core" %>

Following table lists out the core JSTL Tags −

|  |  |
| --- | --- |
| **S.No.** | **Tag & Description** |
| 1 | [<c:out>](https://www.tutorialspoint.com/jsp/jstl_core_out_tag.htm)  Like <%= ... >, but for expressions. |
| 2 | [<c:set >](https://www.tutorialspoint.com/jsp/jstl_core_set_tag.htm)  Sets the result of an expression evaluation in a **'scope'** |
| 3 | [<c:remove >](https://www.tutorialspoint.com/jsp/jstl_core_remove_tag.htm)  Removes a **scoped variable** (from a particular scope, if specified). |
| 4 | [<c:catch>](https://www.tutorialspoint.com/jsp/jstl_core_catch_tag.htm)  Catches any **Throwable** that occurs in its body and optionally exposes it. |
| 5 | [<c:if>](https://www.tutorialspoint.com/jsp/jstl_core_if_tag.htm)  Simple conditional tag which evalutes its body if the supplied condition is true. |
| 6 | [<c:choose>](https://www.tutorialspoint.com/jsp/jstl_core_choose_tag.htm)  Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by **<when>** and **<otherwise>**. |
| 7 | [<c:when>](https://www.tutorialspoint.com/jsp/jstl_core_choose_tag.htm)  Subtag of **<choose>** that includes its body if its condition evalutes to **'true'**. |
| 8 | [<c:otherwise >](https://www.tutorialspoint.com/jsp/jstl_core_choose_tag.htm)  Subtag of **<choose>** that follows the **<when>** tags and runs only if all of the prior conditions evaluated to **'false'**. |
| 9 | [<c:import>](https://www.tutorialspoint.com/jsp/jstl_core_import_tag.htm)  Retrieves an absolute or relative URL and exposes its contents to either the page, a String in **'var'**, or a Reader in **'varReader'**. |
| 10 | [<c:forEach >](https://www.tutorialspoint.com/jsp/jstl_core_foreach_tag.htm)  The basic iteration tag, accepting many different collection types and supporting subsetting and other functionality . |
| 11 | [<c:forTokens>](https://www.tutorialspoint.com/jsp/jstl_core_foreach_tag.htm)  Iterates over tokens, separated by the supplied delimeters. |
| 12 | [<c:param>](https://www.tutorialspoint.com/jsp/jstl_core_param_tag.htm)  Adds a parameter to a containing **'import'** tag's URL. |
| 13 | [<c:redirect >](https://www.tutorialspoint.com/jsp/jstl_core_redirect_tag.htm)  Redirects to a new URL. |
| 14 | [<c:url>](https://www.tutorialspoint.com/jsp/jstl_core_url_tag.htm)  Creates a URL with optional query parameters |

## **Formatting Tags**

The JSTL formatting tags are used to format and display text, the date, the time, and numbers for internationalized Websites. Following is the syntax to include Formatting library in your JSP −

<%@ taglib prefix = "fmt" uri = "http://java.sun.com/jsp/jstl/fmt" %>

Following table lists out the Formatting JSTL Tags −

|  |  |
| --- | --- |
| **S.No.** | **Tag & Description** |
| 1 | [<fmt:formatNumber>](https://www.tutorialspoint.com/jsp/jstl_format_formatnumber_tag.htm)  To render numerical value with specific precision or format. |
| 2 | [<fmt:parseNumber>](https://www.tutorialspoint.com/jsp/jstl_format_parsenumber_tag.htm)  Parses the string representation of a number, currency, or percentage. |
| 3 | [<fmt:formatDate>](https://www.tutorialspoint.com/jsp/jstl_format_formatdate_tag.htm)  Formats a date and/or time using the supplied styles and pattern. |
| 4 | [<fmt:parseDate>](https://www.tutorialspoint.com/jsp/jstl_format_parsedate_tag.htm)  Parses the string representation of a date and/or time |
| 5 | [<fmt:bundle>](https://www.tutorialspoint.com/jsp/jstl_format_bundle_tag.htm)  Loads a resource bundle to be used by its tag body. |
| 6 | [<fmt:setLocale>](https://www.tutorialspoint.com/jsp/jstl_format_setlocale_tag.htm)  Stores the given locale in the locale configuration variable. |
| 7 | [<fmt:setBundle>](https://www.tutorialspoint.com/jsp/jstl_format_setbundle_tag.htm)  Loads a resource bundle and stores it in the named scoped variable or the bundle configuration variable. |
| 8 | [<fmt:timeZone>](https://www.tutorialspoint.com/jsp/jstl_format_timezone_tag.htm)  Specifies the time zone for any time formatting or parsing actions nested in its body. |
| 9 | [<fmt:setTimeZone>](https://www.tutorialspoint.com/jsp/jstl_format_settimezone_tag.htm)  Stores the given time zone in the time zone configuration variable |
| 10 | [<fmt:message>](https://www.tutorialspoint.com/jsp/jstl_format_message_tag.htm)  Displays an internationalized message. |
| 11 | [<fmt:requestEncoding>](https://www.tutorialspoint.com/jsp/jstl_format_requestencoding_tag.htm)  Sets the request character encoding |

## **SQL Tags**

The JSTL SQL tag library provides tags for interacting with relational databases (RDBMSs) such as **Oracle, mySQL**, or **Microsoft SQL Server**.

Following is the syntax to include JSTL SQL library in your JSP −

<%@ taglib prefix = "sql" uri = "http://java.sun.com/jsp/jstl/sql" %>

Following table lists out the SQL JSTL Tags −

|  |  |
| --- | --- |
| **S.No.** | **Tag & Description** |
| 1 | [<sql:setDataSource>](https://www.tutorialspoint.com/jsp/jstl_sql_setdatasource_tag.htm)  Creates a simple DataSource suitable only for prototyping |
| 2 | [<sql:query>](https://www.tutorialspoint.com/jsp/jstl_sql_query_tag.htm)  Executes the SQL query defined in its body or through the sql attribute. |
| 3 | [<sql:update>](https://www.tutorialspoint.com/jsp/jstl_sql_update_tag.htm)  Executes the SQL update defined in its body or through the sql attribute. |
| 4 | [<sql:param>](https://www.tutorialspoint.com/jsp/jstl_sql_param_tag.htm)  Sets a parameter in an SQL statement to the specified value. |
| 5 | [<sql:dateParam>](https://www.tutorialspoint.com/jsp/jstl_sql_dateparam_tag.htm)  Sets a parameter in an SQL statement to the specified java.util.Date value. |
| 6 | [<sql:transaction >](https://www.tutorialspoint.com/jsp/jstl_sql_transaction_tag.htm)  Provides nested database action elements with a shared Connection, set up to execute all statements as one transaction. |

## **XML tags**

The JSTL XML tags provide a JSP-centric way of creating and manipulating the XML documents. Following is the syntax to include the JSTL XML library in your JSP.

The JSTL XML tag library has custom tags for interacting with the XML data. This includes parsing the XML, transforming the XML data, and the flow control based on the XPath expressions.

<%@ taglib prefix = "x"

uri = "http://java.sun.com/jsp/jstl/xml" %>

Before you proceed with the examples, you will need to copy the following two XML and XPath related libraries into your **<Tomcat Installation Directory>\lib** −

* **XercesImpl.jar** − Download it from <https://www.apache.org/dist/xerces/j/>
* **xalan.jar** − Download it from <https://xml.apache.org/xalan-j/index.html>

Following is the list of XML JSTL Tags −

|  |  |
| --- | --- |
| **S.No.** | **Tag & Description** |
| 1 | [<x:out>](https://www.tutorialspoint.com/jsp/jstl_xml_out_tag.htm)  Like <%= ... >, but for XPath expressions. |
| 2 | [<x:parse>](https://www.tutorialspoint.com/jsp/jstl_xml_parse_tag.htm)  Used to parse the XML data specified either via an attribute or in the tag body. |
| 3 | [<x:set >](https://www.tutorialspoint.com/jsp/jstl_xml_set_tag.htm)  Sets a variable to the value of an XPath expression. |
| 4 | [<x:if >](https://www.tutorialspoint.com/jsp/jstl_xml_if_tag.htm)  Evaluates a test XPath expression and if it is true, it processes its body. If the test condition is false, the body is ignored. |
| 5 | [<x:forEach>](https://www.tutorialspoint.com/jsp/jstl_xml_foreach_tag.htm)  To loop over nodes in an XML document. |
| 6 | [<x:choose>](https://www.tutorialspoint.com/jsp/jstl_xml_choose_tag.htm)  Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by **<when>** and **<otherwise>** tags. |
| 7 | [<x:when >](https://www.tutorialspoint.com/jsp/jstl_xml_choose_tag.htm)  Subtag of **<choose>** that includes its body if its expression evalutes to 'true'. |
| 8 | [<x:otherwise >](https://www.tutorialspoint.com/jsp/jstl_xml_choose_tag.htm)  Subtag of **<choose>** that follows the **<when>** tags and runs only if all of the prior conditions evaluates to 'false'. |
| 9 | [<x:transform >](https://www.tutorialspoint.com/jsp/jstl_xml_transform_tag.htm)  Applies an XSL transformation on a XML document |
| 10 | [<x:param >](https://www.tutorialspoint.com/jsp/jstl_xml_param_tag.htm)  Used along with the **transform** tag to set a parameter in the XSLT stylesheet |

## **JSTL Functions**

JSTL includes a number of standard functions, most of which are common string manipulation functions. Following is the syntax to include JSTL Functions library in your JSP −

<%@ taglib prefix = "fn"

uri = "http://java.sun.com/jsp/jstl/functions" %>

Following table lists out the various JSTL Functions −

|  |  |
| --- | --- |
| **S.No.** | **Function & Description** |
| 1 | [fn:contains()](https://www.tutorialspoint.com/jsp/jstl_function_contains.htm)  Tests if an input string contains the specified substring. |
| 2 | [fn:containsIgnoreCase()](https://www.tutorialspoint.com/jsp/jstl_function_containsignorecase.htm)  Tests if an input string contains the specified substring in a case insensitive way. |
| 3 | [fn:endsWith()](https://www.tutorialspoint.com/jsp/jstl_function_endswith.htm)  Tests if an input string ends with the specified suffix. |
| 4 | [fn:escapeXml()](https://www.tutorialspoint.com/jsp/jstl_function_escapexml.htm)  Escapes characters that can be interpreted as XML markup. |
| 5 | [fn:indexOf()](https://www.tutorialspoint.com/jsp/jstl_function_indexof.htm)  Returns the index withing a string of the first occurrence of a specified substring. |
| 6 | [fn:join()](https://www.tutorialspoint.com/jsp/jstl_function_join.htm)  Joins all elements of an array into a string. |
| 7 | [fn:length()](https://www.tutorialspoint.com/jsp/jstl_function_length.htm)  Returns the number of items in a collection, or the number of characters in a string. |
| 8 | [fn:replace()](https://www.tutorialspoint.com/jsp/jstl_function_replace.htm)  Returns a string resulting from replacing in an input string all occurrences with a given string. |
| 9 | [fn:split()](https://www.tutorialspoint.com/jsp/jstl_function_split.htm)  Splits a string into an array of substrings. |
| 10 | [fn:startsWith()](https://www.tutorialspoint.com/jsp/jstl_function_startswith.htm)  Tests if an input string starts with the specified prefix. |
| 11 | [fn:substring()](https://www.tutorialspoint.com/jsp/jstl_function_substring.htm)  Returns a subset of a string. |
| 12 | [fn:substringAfter()](https://www.tutorialspoint.com/jsp/jstl_function_substringafter.htm)  Returns a subset of a string following a specific substring. |
| 13 | [fn:substringBefore()](https://www.tutorialspoint.com/jsp/jstl_function_substringbefore.htm)  Returns a subset of a string before a specific substring. |
| 14 | [fn:toLowerCase()](https://www.tutorialspoint.com/jsp/jstl_function_tolowercase.htm)  Converts all of the characters of a string to lower case. |
| 15 | [fn:toUpperCase()](https://www.tutorialspoint.com/jsp/jstl_function_touppercase.htm)  Converts all of the characters of a string to upper case. |
| 16 | [fn:trim()](https://www.tutorialspoint.com/jsp/jstl_function_trim.htm)  Removes white spaces from both ends of a string. |

Framewok  
Spring  
42.What is Spring?What is features of Spring?Why Spring is popular?

## **Spring Framework**

Spring is a lightweight framework. It can be thought of as a framework of frameworks because it provides support to various frameworks such as [Struts](https://www.javatpoint.com/struts-2-tutorial), [Hibernate](https://www.javatpoint.com/hibernate-tutorial), Tapestry, [EJB](https://www.javatpoint.com/ejb-tutorial), [JSF](https://www.javatpoint.com/jsf-tutorial), etc. The framework, in broader sense, can be defined as a structure where we find solution of the various technical problems.

The Spring framework comprises several modules such as IOC, AOP, DAO, Context, ORM, WEB MVC etc. We will learn these modules in next page. Let's understand the IOC and Dependency Injection first.

## **Spring Framework Features**

These Spring Framework features are as follow

#### ****i. Lightweight****

The Spring Framework is very lightweight with respect to its size and functionality. It is due to its POJO implementation which doesn’t force to inherit any class or implement any interfaces.

#### ****ii. Aspect Oriented Programming(AOP)****

It is an important part of Spring Framework. [**Aspect Oriented Programming**](https://data-flair.training/blogs/spring-aop-tutorial/) is used for separating cross-cutting concerns (for example logging, security etc.) from the business logic of the application.

#### ****iii. Transaction Management****

[**Transaction Management**](https://data-flair.training/blogs/spring-transaction-management/) use for unify several transaction management APIs and is used to coordinate transactions for Java object. Also, not tie to the J2EE environment and use with containerless environments.

#### ****iv. Container****

The Spring Framework designs and manages the lifecycle and configurations of application objects.

#### ****v. Dependency Injection****

[**Dependency Injection**](https://data-flair.training/blogs/spring-dependency-injection/) is a feature of Spring Framework allows you to develop loosely coupled applications. Therefore, the unit testing of these loosely coupled applications becomes easier. This also allows the developer to swap out some of the modules according to its need.

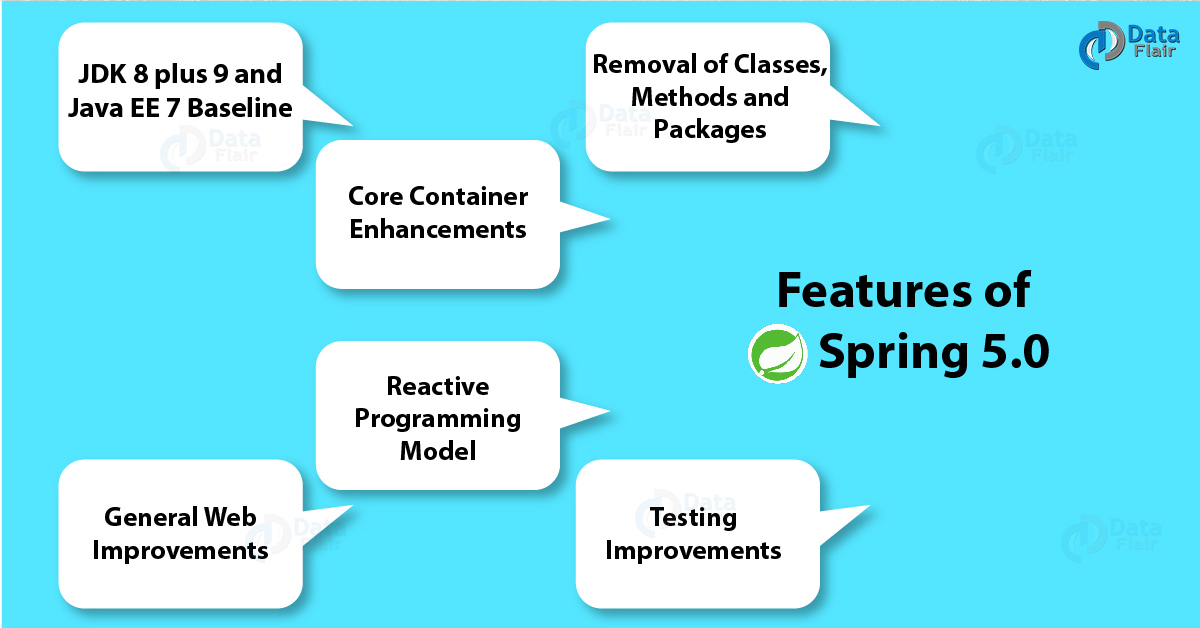
#### ****vi. Integration with other frameworks****

A great thing about this framework is that it doesn’t try to solve the problems have already solved. It just tries to integrate them with its framework which provides a solution to greater problems. Example IBATIS, Hibernate, Toplink etc.

[**Follow this link to know about Spring IoC Containers**](https://data-flair.training/blogs/spring-ioc-containers/)

## **4. Features of Spring 5.0**

Now major enhancements are done since the introduction of Spring Framework by Rod Johnson in 2003. Several versions have been released after its first. As of now the Spring Framework 5.0x versions are released in the market. So, let’s run down to the major upgraded features and enhancements of Spring 5.0 with Java 8 as a minimum requirement across all coding bases with the compatibility of JDK 9 at runtime as well as for build and test suit:

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/06/Features-of-Spring-Framework.jpg)

*Features of Spring 5.0*

### **a. JDK 8 plus 9 and Java EE 7 Baseline**

The Spring Framework codebase is based on code level of Java Programming 8. This code level provides readability using inferred generics and the conditional support for Java 8 is directly provided in the code. Now Java EE 7 API level requires in Spring modules along with Servlet 3.1, JMS2.0, JPA2.1, Bean validation1.1. Recent servers such as Jetty 9.3+, Tomcat 8.5+, WildFly10+ adds to Spring 5.0. The compatibility of JDK 9 has been added at runtime as well as for build and test suit.

### **b. Removal of Classes, Methods, and Packages**

The Package mock.staticmock has been removed from Spring module along with the support for AnnotationDrivenStaticEntityMockingControl. The minimum requirement is Tiles3 and Hibernate5 which means packages web.view.tiles2 and orm.hibernate3/orm.hibernate4 drop. Spring 5.0 has also withdrawn its support for Portlet, Velocity, XMLBeans, Guava, JDO and JasperReports. Therefore, those who need them to work recommends to Spring Framework 4.3.x. Many of the deprecated methods and classes remove with some compromises for commonly use methods in Spring ecosystem.

[**Do you know What is Spring Boot CLI?**](https://data-flair.training/blogs/spring-boot-cli/)

### **c. Core Container Enhancements**

With the introduction of Spring 5.0 there are major upgrades made in Core container. These are as follows:

#### i. JDK 8+Improvements

* Some declarations of Java 8 methods in core Spring Framework interfaces.
* Use of Standard Charsets of JDK 7.
* Efficient parameter method access using Java 8.

#### ii. JDK 9 Preparations

* Using revised exceptional handling consistency of initialization of constructor is done.

#### iii. XML Related Upgrades

* XML namespaces streamline for non-versioned schemas.
* No support for deprecated features along with a resolution for latest XSD files.
* Version specific declarations continue to support but validates.

### **d. Reactive Programming Model**

The Spring Framework’s latest version 5 has several changes with respect to the Reactive Programming model. Major ones being the use of spring core data buffer along with encoder/decoder abstractions and spring-web HTTP message codec implementations with XML and JSON support. Also, the new WebClient and the new spring web reactive module introduces with reactive support for @Controller model adapting Servlet3.1 as well as non-Servlets runtime containers for example Undertow.

### **e. General Web Improvements**

In web improvements unified support for media type resolution is provided using MediaTypeFactory. Also, full Servel 3.1 support is provided to [Spring](https://spring.io/) Framework along with Protobuf 3.0 support in version 5.

### **f. Testing Improvements**

With the introduction of this version, major changes made to the testing environment of Spring Framework.

i. Complete support to Junit 5’s Jupiter programming in Spring TestContext Framework.

* SpringExtension is an implementation of several extension of APIs from Jupiter that gives full support for existing feature of TestContext Framework which enables using @ExtendWith a SpringExtention.class.
* @SpringJUintConfig which combines @ExtendedWith from Jupiter with @ContextConfigurationfrom (a TestContext framework).
* @SpringJunitWebConfig an annotation which combines @ExtendedWith from Jupiter with @ContextConfiguration and @WebAppConfiguration of a TestContext framework.

[**Let’s learn – Spring BeanPostProcessors**](https://data-flair.training/blogs/beanpostprocessors/)

ii. New test execution callback methods introduced in Spring TestContext Framework using TestNG, JUnit 5 via SpringRunner.  
iii. XMLUnit support is upgraded to version 2.2.  
So, this was all about Spring Framework Features. Hope you like our explanation.

43.What is IOC container and How to create IOC container?What is Dependency Injection?What is difference between Setter and Constructor Injection?

|  |
| --- |
| How to Create XmlBeanFactory |
| InputStream is = new FileInputStream("beans.xml");  BeanFactory factory = new XmlBeanFactory(is);    //Get bean  HelloWorld obj = (HelloWorld) factory.getBean("helloWorld");   1. [**FileSystemXmlApplicationContext**](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/support/FileSystemXmlApplicationContext.html) – This container loads the definitions of the beans from an XML file. Here you need to provide the full path of the XML bean configuration file to the constructor. 2. [**ClassPathXmlApplicationContext**](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/support/ClassPathXmlApplicationContext.html) – This container loads the definitions of the beans from an XML file. Here you do not need to provide the full path of the XML file but you need to set CLASSPATH properly because this container will look bean configuration XML file in CLASSPATH. 3. [**WebXmlApplicationContext**](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/context/support/XmlWebApplicationContext.html) – This container loads the XML file with definitions of all beans from within a web application.  |  | | --- | | How to create ApplicationContext | | ApplicationContext context = new FileSystemXmlApplicationContext("beans.xml");  HelloWorld obj = (HelloWorld) context.getBean("helloWorld"); | |

**Dependency Injection** is a technique to separate the creation of **dependencies** from the main class under consideration. Using DI you **inject** the objects needed by a class typically through a **constructor**.

44.What is Autowiring?How many types of autowiring is there?  
Usecase:  
If two bean are Injecting the same Type then we getting ambiguity problem and how to resolve it?

45.Difference between BeanFactory and ApplicationContext IOC container?

46.What is the Spring Jdbc? and How to configure DataSource in Spring Jdbc?

**Spring JDBC** is an abstraction framework for **JDBC** that provides easier access to datasources without all the exception handling and parsing of SQL fetch results. **Spring JDBC** basically does lots of things for you

 private static DataSource dataSource;

    public static void main(String[] args) throws Exception {

        dataSource = getDataSource();

        // JdbcTemplate template = new JdbcTemplate(dataSource); // constructor

        JdbcTemplate template = new JdbcTemplate();

        template.setDataSource(dataSource);

        System.out.println(dataSource.getClass());

    }

    public static DriverManagerDataSource getDataSource() {

  DriverManagerDataSource dataSource = new DriverManagerDataSource();

  dataSource.setDriverClassName(driverClassName);

  dataSource.setUrl(url);

  dataSource.setUsername(dbUsername);

  dataSource.setPassword(dbPassword);

  return dataSource;

    }

}

47.What is the Jdbc Template?  
48.What is the AOP ?and Principles of AOP and advices of AOP and how to apply cross cutting logic?  
49.What is Transaction and How to manage Global Transaction?

A database **transaction** is a sequence of actions that are treated as a single unit of work. These actions should either complete entirely or take no effect at all. **Transaction management** is an important part of RDBMS-oriented enterprise application to ensure data integrity and consistency

**Global transaction management** is the monitoring of **transactions** that can include operations on two or more different data sources. This feature of **transaction** processing enables data resources to be returned to a pre-**transaction** state if some error occurs. Either all data resources are updated or none are.

50.What is the Isolation and Propagation?

[**Propagation**](http://static.springsource.org/spring/docs/3.0.x/javadoc-api/org/springframework/transaction/annotation/Propagation.html)

Defines how transactions relate to each other. Common options:

* Required: Code will always run in a transaction. Creates a new transaction or reuses one if available.
* Requires\_new: Code will always run in a new transaction. Suspends the current transaction if one exists.

[**Isolation**](http://static.springsource.org/spring/docs/3.0.x/javadoc-api/org/springframework/transaction/TransactionDefinition.html)

Defines the data contract between transactions.

* Read Uncommitted: Allows dirty reads.
* Read Committed: Does not allow dirty reads.
* Repeatable Read: If a row is read twice in the same transaction, the result will always be the same.
* Serializable: Performs all transactions in a sequence.

The different levels have different performance characteristics in a multi-threaded application. I think if you understand the dirty reads concept you will be able to select a good option.

@Transactional(propagation=Propagation.REQUIRES\_NEW)

public void provideService() {

repo1.retrieveFoo();

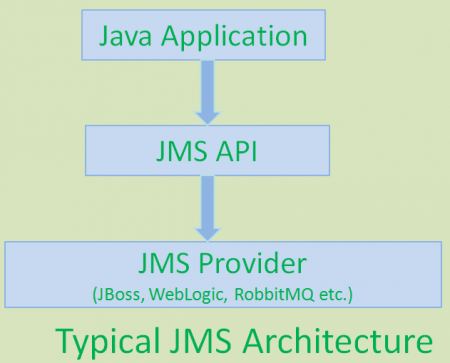
repo2.retrieveFoo();

}

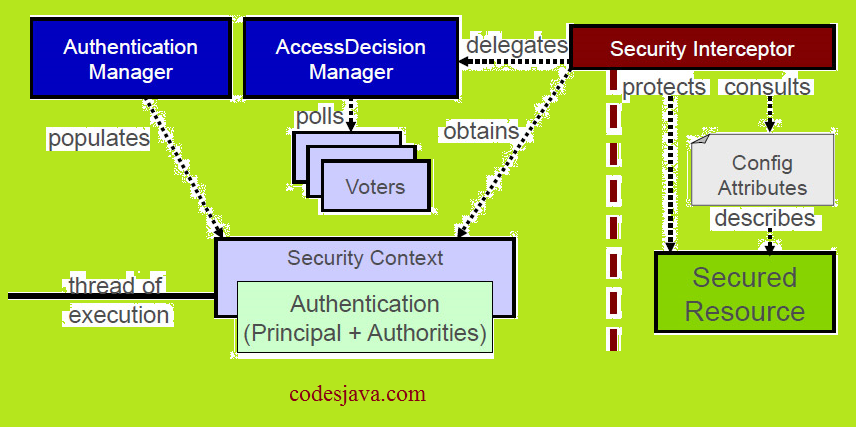
}

51.What is JMS?How does it works?

Java Message Service (**JMS**) is an application program interface (API) from Sun Microsystems that supports the formal communication known as messaging between computers in a network. Sun's **JMS** provides a common interface to standard messaging protocols and also to special messaging services in support of Java programs.

  
52.MVC flow and Architecture with Spring bean Configuration and Annotations also?How to implement  
53. Spring Security architecture and what are components are there and How to implement it?

## Spring security architecture diagram



## Spring Security Authentication

Spring security provides AuthenticationManager interface for authentication process. It has only one method.

|  |
| --- |
| **public** **interface** AuthenticationManager {  Authentication authenticate(Authentication authentication) **throws** AuthenticationException;  } |

The authenticate() method can return an Authentication if the input represents a valid principal. Normally it returns authenticated=truefor the above mention case. It will throw an AuthenticationException if the input represents an invalid principal. It will return null if it can’t decide whether the input value is valid or invalid.

The ProviderManager is the most common implementation of AuthenticationManager. It delegates to a chain of AuthenticationProvider objects. It has an optional parent. It can consult to it if all providers return null. AuthenticationException will be thrown if no parent is available.

AuthenticationProvider is like an AuthenticationManager only difference is that it has an extra method. This extra method allow the caller to query if it supports a given Authentication type.

|  |
| --- |
| **public** **interface** AuthenticationProvider {  Authentication authenticate(Authentication authentication)  **throws** AuthenticationException;  **boolean** supports(Class<?> authentication);  } |

**Note: In an application we may have logical groups of protected resources. For example all web resources that match a path pattern /app/\*\*. In such situations each group can have its own dedicated AuthenticationManager. These dedicated AuthenticationManager instances can share a common parent which will act like a global resource.**

Spring security framework provides the facility to customize the Authentication Managers with the help of AuthenticationManagerBuilder.

|  |
| --- |
| @Configuration  **public** **class** ApplicationSecurity **extends** WebSecurityConfigurerAdapter {  *// Our code statements*  @Autowired  **public** initialize(AuthenticationManagerBuilder builder, DataSource dataSource) {  builder.jdbcAuthentication().dataSource(dataSource).withUser("jai")  .password("unknown").roles("USER");  }  } |

**Note: Spring boot comes with a default global AuthenticationManager which is secure enough on its own. We can replace it by providing your own bean of type AuthenticationManager.**

54.What is Spring Data JPA with example?

Spring Data JPA API provides JpaTemplate class to integrate spring application with JPA.

JPA (Java Persistent API) is the sun specification for persisting objects in the enterprise application. It is currently used as the replacement for complex entity beans.

The implementation of JPA specification are provided by many vendors such as:

* Hibernate
* Toplink
* iBatis
* OpenJPA etc.

## **Advantage of Spring JpaTemplate**

You don't need to write the before and after code for persisting, updating, deleting or searching object such as creating Persistence instance, creating EntityManagerFactory instance, creating EntityTransaction instance, creating EntityManager instance, commiting EntityTransaction instance and closing EntityManager.

So, it **save a lot of code**.

Spring Data Commons project provides repository abstraction which is extended by the datastore-specific subprojects.

We have to be familiar with the Spring Data repository interfaces as it will help us with the implementation of the interfaces. Let’s have a look at the interfaces.

**Spring Data Commons: –** Following interfaces are provided as part of this project:

* [*Repository<T, ID extends Serializable>*](https://docs.spring.io/spring-data/commons/docs/current/api/index.html?org/springframework/data/repository/Repository.html)  : This interface is a marker interface.
  1. It captures the type of the managed entity and the type of the entity’s id.
  2. It helps the Spring container to discover the “concrete” repository interfaces when classpath is scanned.
* [*CrudRepository<T, ID extends Serializable>*](https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/repository/CrudRepository.html) : It provides CRUD operations for the managed entity.
* [*PagingAndSortingRepository<T, ID extends Serializable>*](https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/repository/PagingAndSortingRepository.html) : This interface declares the methods that are used to sort and paginate entities that are retrieved from the database.
* [*QueryDslPredicateExecutor<T>*](https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/querydsl/QueryDslPredicateExecutor.html) : It is not a “repository interface”. It declares the methods that are used to retrieve entities from the database by using [QueryDsl](http://www.querydsl.com/) *Predicate* objects.

**Spring Data JPA: –**  This project provides the following interfaces:

* [*JpaRepository<T, ID extends Serializable>*](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html)  : This interface is a JPA specific repository interface that combines the methods declared by the common repository interfaces behind a single interface.
* [*JpaSpecificationExecutor<T>*](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaSpecificationExecutor.html) : This is again not a “repository interface”. It declares the methods that are used to retrieve entities from the database by using [*Specification<T>*](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/domain/Specification.html) objects that use the JPA criteria API.

55.Spring with Hibernate Integration?  
56.Spring with Restful Integration?

Hibernate  
57.What is ORM frame?What is the advantages over JDBC API?

HIBERNATE.

1. Hibernate is flexible and powerful ORM solution to map Java classes to database tables. Hibernate itself takes care of this mapping using XML files so developer does not need to write code for this.
2. Hibernate provides transparent persistence and developer does not need to write code explicitly to map database tables tuples to application objects during interaction with RDBMS.
3. Hibernate provides a powerful query language Hibernate Query Language (independent from type of database) that is expressed in a familiar SQL like syntax and includes full support for polymorphic queries. Hibernate also supports native SQL statements. It also selects an effective way to perform a database manipulation task for an application.
4. Hibernate provides this mapping itself. The actual mapping between tables and application objects is done in XML files. If there is change in Database or in any table then the only need to change XML file properties.
5. Hibernate reduces lines of code by maintaining object-table mapping itself and returns result to application in form of Java objects. It relieves programmer from manual handling of persistent data, hence reducing the development time and maintenance cost.
6. Hibernate, with Transparent Persistence, cache is set to application work space. Relational tuples are moved to this cache as a result of query. It improves performance if client application reads same data many times for same write. Automatic Transparent Persistence allows the developer to concentrate more on business logic rather than this application code.
7. Hibernate enables developer to define version type field to application, due to this defined field Hibernate updates version field of database table every time relational tuple is updated in form of Java class object to that table. So if two users retrieve same tuple and then modify it and one user save this modified tuple to database, version is automatically updated for this tuple by Hibernate. When other user tries to save updated tuple to database then it does not allow saving it because this user does not have updated data.

58.What is Hibernate BootStraping?Which hibernate bootstrapping you worked on Project?(Hibernate Bootstrapping 4).

The term **bootstrapping** refers to initializing and starting a software component. In **Hibernate**, we are specifically talking about the process of building a fully functional SessionFactory instance or EntityManagerFactory instance, for JPA. The process is very different for each

**Hibernate** 5 **Bootstrapping**  
59.What first and Second Level cache in Hibernate.  
60.What is Session and SessionFactory.

SessionFactory is an interface. SessionFactory can be created by providing Configuration object, which will contain all DB related property details pulled from either hibernate.cfg.xml file or hibernate.properties file. SessionFactory is a factory for Session objects.

We can create one SessionFactory implementation per database in any application. If your application is referring to multiple databases, then you need to create one SessionFactory per database.

The SessionFactory is a heavyweight object; it is usually created during application start up and kept for later use. The SessionFactory is a thread safe object and used by all the threads of an application.

 A **Session** is used to get a physical connection with a database. The **Session** object is lightweight and designed to be instantiated each time an interaction is needed with the database. Persistent objects are saved and retrieved through a **Session** object.

Configuration configuration = new Configuration();

        configuration.configure("/j2n-hibernate.cfg.xml");

        configuration.addAnnotatedClass(Employee.class);

        ServiceRegistry srvcReg = new StandardServiceRegistryBuilder().applySettings(configuration.getProperties()).build();

        sessionFactory = configuration.buildSessionFactory(srvcReg);

    }

    public static Session getSession() throws HibernateException {

        Session retSession=null;

            try {

                retSession = sessionFactory.openSession();

61.What is HQL?give example of HQL?

**Hibernate** Query Language (**HQL**) is an object-oriented query language, similar to SQL, but instead of operating on tables and columns, **HQL** works with persistent objects and their properties. **HQL** queries are translated by **Hibernate** into conventional SQL queries, which in turns perform action on database.

String hql = "FROM com.hibernatebook.criteria.Employee";

Query query = session.createQuery(hql);

List results = query.list();

String hql = "FROM Employee E WHERE E.id = :employee\_id";

Query query = session.createQuery(hql);

query.setParameter("employee\_id",10);

List results = query.list();

62.What is Criteria API with example?

The **Hibernate Criteria** Query Language (HCQL) is used to fetch the records based on the specific **criteria**. The **Criteria** interface provides methods to apply **criteria** such as retreiving all the records of table whose salary is greater than 50000 etc.

Crietria c=session.createCriteria(Emp.**class**);

c.setFirstResult(10);

c.setMaxResult(20);

List list=c.list();

Crietria c=session.createCriteria(Emp.**class**);

c.add(Restrictions.gt("salary",10000));//salary is the propertyname

List list=c.list();

63.Difference between HQL and Criteria?

* HQL is to perform both select and non-select operations on the data,  but Criteria is only for selecting the data, we cannot perform non-select operations using criteria
* HQL is suitable for executing Static Queries, where as Criteria is suitable for executing Dynamic Queries
* HQL doesn’t support pagination concept, but we can achieve pagination with Criteria
* Criteria used to take more time to execute then HQL
* With Criteria we are safe with SQL Injection because of its dynamic query generation but in HQL as your queries are either fixed or parametrized, there is no safe from SQL Injection.

64.Difference between Criteria and Detached Criteria?

Using a DetachedCriteria is exactly the same as a Criteria except you can do the initial creation and setup of your query without having access to the session. When it comes time to run your query, you must convert it to an executable query with getExecutableCriteria(session).

This is useful if you are building complicated queries, possibly through a multi-step process, because you don't need access to the Session everywhere. You only need the Session at the final step when you run the query.

Under the hood, DetachedCriteria uses a CriteriaImpl which is the same class you get if you call session.createCriteria().

-Detached criteria is very good alternate when the hibernate session is not present.

-The criteria are online, which means that it uses Session class object. But the detached criteria is offline because it *doesn't need a session*.

-Then the detach criteria allow code reusability.

DetachedCriteria dcr = DetachedCriteria.forClass(Student.class).add(Property.forName("gender").eq("male"));

dcr.addOrder(Order.desc("stud\_id"));

List<Student> results = dcr.getExecutableCriteria(session).list();

65.How to call Procedures and functions in HQL?

DELIMITER $$

CREATE PROCEDURE `GetStocks`(int\_stockcode varchar(20))

BEGIN

SELECT \* FROM stock where stock\_code = int\_stockcode;

END $$

DELIMITER ;

Copy

In MySQL, you can simple call it with a **call** keyword :

CALL GetStocks('7277');

Copy

## **Hibernate call store procedure**

In Hibernate, there are three approaches to call a database store procedure.

## **1. Native SQL – createSQLQuery**

You can use **createSQLQuery()** to call a store procedure directly.

Query query = session.createSQLQuery(

"CALL GetStocks(:stockCode)")

.addEntity(Stock.class)

.setParameter("stockCode", "7277");

List result = query.list();

for(int i=0; i<result.size(); i++){

Stock stock = (Stock)result.get(i);

System.out.println(stock.getStockCode());

}

Copy

## **2. NamedNativeQuery in annotation**

Declare your store procedure inside the **@NamedNativeQueries** annotation.

//Stock.java

...

@NamedNativeQueries({

@NamedNativeQuery(

name = "callStockStoreProcedure",

query = "CALL GetStocks(:stockCode)",

resultClass = Stock.class

)

})

@Entity

@Table(name = "stock")

public class Stock implements java.io.Serializable {

...

Copy

Call it with **getNamedQuery()**.

Query query = session.getNamedQuery("callStockStoreProcedure")

.setParameter("stockCode", "7277");

List result = query.list();

for(int i=0; i<result.size(); i++){

Stock stock = (Stock)result.get(i);

System.out.println(stock.getStockCode());

}

Copy

## **3. sql-query in XML mapping file**

Declare your store procedure inside the “**sql-query**” tag.

<!-- Stock.hbm.xml -->

...

<hibernate-mapping>

<class name="com.mkyong.common.Stock" table="stock" ...>

<id name="stockId" type="java.lang.Integer">

<column name="STOCK\_ID" />

<generator class="identity" />

</id>

<property name="stockCode" type="string">

<column name="STOCK\_CODE" length="10" not-null="true" unique="true" />

</property>

...

</class>

<sql-query name="callStockStoreProcedure">

<return alias="stock" class="com.mkyong.common.Stock"/>

<![CDATA[CALL GetStocks(:stockCode)]]>

</sql-query>

</hibernate-mapping>

Copy

Call it with **getNamedQuery()**.

Query query = session.getNamedQuery("callStockStoreProcedure")

.setParameter("stockCode", "7277");

List result = query.list();

for(int i=0; i<result.size(); i++){

Stock stock = (Stock)result.get(i);

System.out.println(stock.getStockCode());

}

66.How to manage Transactions in Hibernate?

A transaction is associated with Session and instantiated by calling **session.beginTransaction()**.

The methods of Transaction interface are as follows:

1. **void begin()** starts a new transaction.
2. **void commit()** ends the unit of work unless we are in FlushMode.NEVER.
3. **void rollback()** forces this transaction to rollback.
4. **void setTimeout(int seconds)** it sets a transaction timeout for any transaction started by a subsequent call to begin on this instance.
5. **boolean isAlive()** checks if the transaction is still alive.
6. **void registerSynchronization(Synchronization s)** registers a user synchronization callback for this transaction.
7. **boolean wasCommited()** checks if the transaction is commited successfully.
8. **boolean wasRolledBack()** checks if the transaction is rolledback successfully.

67.What is n+1 problem and How to resolve it?

N+1 problem is a performance issue in Object Relational Mapping that fires multiple select queries (N+1 to be exact, where N= number of records in table) in database for a single select query at application layer. Hibernate provides multiple ways to catch and prevent this problem.

@ManyToMany(fetch = FetchType.LAZY)

### **Spring Data JPA Approach**

If we are using Spring Data JPA, then we have two options to achieve this - using EntityGraph or using select query with fetch join.

public interface UserRepository extends CrudRepository<User, Long> {

List<User> findAllByRolesIn(List<Role> roles);

@Query("SELECT p FROM User p LEFT JOIN FETCH p.roles")

List<User> findWithoutNPlusOne();

@EntityGraph(attributePaths = {"roles"})

List<User> findAll();

}

|  |  |
| --- | --- |
|  | N+1 queries are issued at database level. |
|  | using left join fetch, we resolve the N+1 problem |
|  | using attributePaths, Spring Data JPA avoids N+1 problem |

### **Hibernate Aproach**

if its pure Hibernate, then the following solution will work.

*HQL Query*

"from User u join fetch u.roles roles roles"

*Hibernate Criteria API*

Criteria criteria = session.createCriteria(User.class);

criteria.setFetchMode("roles", FetchMode.EAGER);

under the hood, all these approaches work similar and they issue a similar database query with left join fetch

68.Hibernate Entity Object Life cycle?

## Transient:

One newly created object,with out having any relation with the database, means never persistent, not associated with any Session object

## Persistent:

Having the relation with the database, associated with a unique Session object

## Detached:

previously having relation with the database [persistent ], now not associated with any Session

see the next sessions for the better understanding of the life cycle states of pojo class object(s) the hibernate

69.Difference between merge and update?and difference between save() persit()?  
70.Difference between get() and load()?  
SOAP  
71.What is XML and XSD?

**XSD:**  
XSD (XML Schema Definition) specifies how to formally describe the elements in an Extensible Markup Language (XML) document.  
**Xml:**  
XML was designed to **describe data**.It is independent from software as well as hardware.  
It enhances the following things.  
-Data sharing.  
-Platform independent.  
-Increasing the availability of Data.

Differences:

1. XSD is based and written on XML.
2. XSD defines elements and structures that can appear in the document, while XML does not.
3. XSD ensures that the data is properly interpreted, while XML does not.
4. An XSD document is validated as XML, but the opposite may not always be true.
5. XSD is better at catching errors than XML.

An XSD **defines elements** that can be used in the documents, relating to the actual data with which it is to be encoded.

72.Difference between XML and JSON?

|  |  |
| --- | --- |
| **JSON** | **XML** |
| JSON object has a type | XML data is typeless |
| JSON types: string, number, array, Boolean | All XML data should be string |
| Data is readily accessible as JSON objects | XML data needs to be parsed. |
| JSON is supported by most browsers. | Cross-browser XML parsing can be tricky |
| JSON has no display capabilities. | XML offers the capability to display data because it is a markup language. |
| JSON supports only text and number data type. | XML support various data types such as number, text, images, charts, graphs, etc. It also provides options for transferring the structure or format of the data with actual data. |
| Retrieving value is easy | Retrieving value is difficult |
| Supported by many Ajax toolkit | Not fully supported by Ajax toolkit |
| A fully automated way of deserializing/serializing JavaScript. | Developers have to write JavaScript code to serialize/de-serialize from XML |
| Native support for object. | The object has to be express by conventions - mostly missed use of attributes and elements. |
| It supports only UTF-8 encoding. | It supports various encoding. |
| It doesn't support comments. | It supports comments. |
| JSON files are easy to read as compared to XML. | XML documents are relatively more difficult to read and interpret. |
| It does not provide any support for namespaces. | It supports namespaces. |
| It is less secured. | It is more secure than JSON. |

## JSON Code vs XML Code

Let's see a sample JSON Code

{

"student": [

{

"id":"01",

"name": "Tom",

"lastname": "Price"

},

{

"id":"02",

"name": "Nick",

"lastname": "Thameson"

}

]

}

Let's study the same code in XML

<?xml version="1.0" encoding="UTF-8" ?>

<root>

<student>

<id>01</id>

<name>Tom</name>

<lastname>Price</lastname>

</student>

<student>

<id>02</id>

<name>Nick</name>

<lastname>Thameson</lastname>

</student>

</root>

## Advantages of using JSON

Here are the important benefits/ pros of using JSON:

* Provide support for all browsers
* Easy to read and write
* Straightforward syntax
* You can natively parse in JavaScript using eval() function
* Easy to create and manipulate
* Supported by all major JavaScript frameworks
* Supported by most backend technologies
* JSON is recognized natively by JavaScript
* It allows you to transmit and serialize structured data using a network connection.
* You can use it with modern programming languages.
* JSON is text which can be converted to any object of JavaScript into JSON and send this JSON to the server.

## Advantages of using XML

Here are significant benefits/cons of using XML:

* Makes documents transportable across systems and applications. With the help of XML, you can exchange data quickly between different platforms.
* XML separates the data from HTML
* XML simplifies platform change process

## Disadvantages of using JSON

Here are cons/ drawback of using JSON:

* No namespace support, hence poor extensibility
* Limited development tools support
* It offers support for formal grammar definition

## Disadvantages of using XML

Here, are cons/ drawbacks of using XML:

* XML requires a processing application
* The XML syntax is very similar to other alternatives 'text-based' data transmission formats which is sometimes confusing
* No intrinsic data type support
* The XML syntax is redundant
* Does n't allow the user to create his tags.

73.Difference between SAX and DOM?

## 1. Key Difference of DOM and SAX

* **DOM**stands for Document Object Model while **SAX**stands for Simple API for XML parsing.
* DOM parser load full XML file in memory and creates a tree representation of XML document, while SAX is an event based XML parser and doesn’t load whole XML document into memory.
* DOM parser load entire XML file in memory and creates a tree structure of XML document, while SAX is an event based XML parser and doesn’t load whole XML document into memory
* If you know you have sufficient amount of memory in your server you can choose DOM as this faster because load entire xml in memory and works as tree structure which is faster to access.
* As a thumb rule, for small and medium sized XML documents, DOM is much faster than SAX because of in memory agnostic.

## 2. DOM Vs SAX

|  | **DOM (Document Object Model) Parser** | **SAX (Simple API for XML) Parser** |
| --- | --- | --- |
| Abbreviation | **DOM**stands for Document Object Model | **SAX**stands for Simple API for XML Parsing |
| type | Load entire memory and keep in tree structure | event based parser |
| size of Document | good for smaller size | good to choose for larger size of file. |
| Load | Load entire document in memory | does not load entire document. |
| suitable | better suitable for smaller and efficient memory | SAX is suitable for larger XML doc |

74.Difference between SOAP and Restful web Services?  
75.Which SOAP based Web Services API is used in Project?

**What Is a SOAP API**? **SOAP** is a standard communication protocol system that permits processes using different operating systems like Linux and Windows to communicate via HTTP and its XML. **SOAP** based **APIs** are designed to create, recover, update and delete records like accounts, passwords, leads, and custom objects

76.How to apply Security in SOAP Web Services?

8

Different ways and different types of security we can implement: Message-level security

* **Transport-level security**: Such as HTTP Basic/Digest and SSL
* **Message level security**: Such as WS-Security, XML digital signature, XML Encryption,XKMS (**X**ML **K**ey **M**anagement **S**pecification), XACML (e**X**tensible **A**ccess **C**ontrol **M**arkup **L**anguage), SAML (**S**ecure **A**ssertion **M**arkup **L**anguage), ebXML Message Service, The Liberty Alliance Project. [for more detals](https://www.javaworld.com/article/2073287/soa/secure-web-services.html)
* **Access control security**:A security role is a privilege granted to users or groups based on specific conditions.

Most commonly we use WS-Security for SOAP Web Services. A [**WS-security profile**](https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/integrate/inbound-soap/concept/c_WS-SecurityProfiles.html) determines how WS-security is enabled.

1. **WSS X.509 Token Profile:** Use the X.509 framework for a WSS X.509 security profile.
2. **WSS UsernameToken Profile**: When specifying the X.509 Token Profile, you can also supply a UsernameToken in the SOAP request.

example:

<wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="SecurityToken-6138db82-5a4c-4bf7-915f-af7a10d9ae96">

<wsse:Username>user</wsse:Username>

<wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordDigest">CBb7a2itQDgxVkqYnFtggUxtuqk=</wsse:Password>

<wsse:Nonce>5ABcqPZWb6ImI2E6tob8MQ==</wsse:Nonce>

<wsu:Created>2010-06-08T07:26:50Z</wsu:Created>

</wsse:UsernameToken>

The above element includes into SOAP header as follows:

SOAPEnvelope envelope = smc.getMessage().getSOAPPart().getEnvelope();

SOAPHeader header = envelope.addHeader();

SOAPElement security = header.addChildElement("Security", "wsse", "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd");

SOAPElement usernameToken = security.addChildElement("UsernameToken", "wsse");

SOAPElement username = usernameToken.addChildElement("Username", "wsse");

username.addTextNode(user);

SOAPElement password = usernameToken.addChildElement("Password", "wsse");

password.setAttribute("Type", "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordDigest");

password.addTextNode(encodedPass); //encodedPass = Base64 ( SHA-1 ( nonce + created + password ) )

SOAPElement nonce =

usernameToken.addChildElement("Nonce", "wsse");

nonce.addTextNode(Base64.encodeBytes(nonceString.getBytes()));

SOAPElement created = usernameToken.addChildElement("Created", "wsu","http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd");

created.addTextNode(creatTime);

The following example is simple adding user and password to HTTP header only.

* [Application Authentication with JAX-WS](http://www.mkyong.com/webservices/jax-ws/application-authentication-with-jax-ws/) Here’s a detail example to show you how to handle application level authentication with JAX-WS.
* [Container Authentication with JAX-WS + (Tomcat version)](http://www.mkyong.com/webservices/jax-ws/container-authentication-with-jax-ws-tomcat/) Here’s a detail example to show you how to implement container authentication with JAX-WS, under Tomcat.

Application Authentication with JAX-WS using [WebServiceContext](https://docs.oracle.com/javaee/6/api/javax/xml/ws/WebServiceContext.html) interface

WebServiceImpl.java

package com.javacodegeeks.enterprise.ws;

import java.util.List;

import java.util.Map;

import javax.annotation.Resource;

import javax.jws.WebService;

import javax.xml.ws.WebServiceContext;

import javax.xml.ws.handler.MessageContext;

@WebService(endpointInterface = "com.javacodegeeks.enterprise.ws.WebServiceInterface")

public class WebServiceImpl implements WebServiceInterface {

@Resource

WebServiceContext webServiceContext;

@Override

public String getHelloWorldAsString(String str) {

MessageContext messageContext = webServiceContext.getMessageContext();

// get request headers

Map<?,?> requestHeaders = (Map<?,?>) messageContext.get(MessageContext.HTTP\_REQUEST\_HEADERS);

List<?> usernameList = (List<?>) requestHeaders.get("username");

List<?> passwordList = (List<?>) requestHeaders.get("password");

String username = "";

String password = "";

if (usernameList != null) {

username = usernameList.get(0).toString();

}

if (passwordList != null) {

password = passwordList.get(0).toString();

}

// of course this is not real validation

// you should validate your users from stored databases credentials

if (username.equals("nikos") && password.equals("superpassword")) {

return "Valid User :"+str;

} else {

return "Unknown User!";

}

}

}

WebServiceClient.java

package com.javacodegeeks.enterprise.ws.client;

import java.net.URL;

import java.util.Collections;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.xml.namespace.QName;

import javax.xml.ws.BindingProvider;

import javax.xml.ws.Service;

import javax.xml.ws.handler.MessageContext;

import com.javacodegeeks.enterprise.ws.WebServiceInterface;

public class WebServiceClient{

public static void main(String[] args) throws Exception {

URL wsdlUrl = new URL("http://localhost:8888/webservice/helloworld?wsdl");

//qualifier name ...

QName qname = new QName("http://ws.enterprise.javacodegeeks.com/", "WebServiceImplService");

Service service = Service.create(wsdlUrl, qname);

WebServiceInterface sayHello = service.getPort(WebServiceInterface.class);

Map<String, Object> requestContext = ((BindingProvider)sayHello).getRequestContext();

requestContext.put(BindingProvider.ENDPOINT\_ADDRESS\_PROPERTY, "http://localhost:8888/webservice/helloworld?wsdl");

Map<String, List<String>> requestHeaders = new HashMap<String, List<String>>();

requestHeaders.put("username", Collections.singletonList("nikos"));

requestHeaders.put("Password", Collections.singletonList("superpassword"));

requestContext.put(MessageContext.HTTP\_REQUEST\_HEADERS, requestHeaders);

System.out.println(sayHello.getHelloWorldAsString("- This is Java Code Geeks"));

}

77.What is WSDL? And what are sections are there in WSDL document?

**WSDL**, or Web Service Description Language, is an XML based definition language. It's used for describing the functionality of a SOAP based web service. **WSDL** files are central to testing SOAP-based services. SoapUI uses **WSDL** files to generate test requests, assertions and mock services.

WSDL: WSDL tells about the functions that you can implement or exposed to the client. For example: add, delete, subtract and so on.

**WSDL:** When we go to a restaurant we see the Menu Items, those are the WSDL's.

WSDL -> SOAP message consist of SoapEnevelope->SoapHeader and SoapBody. It doesn't define what would be message format? what are all the transports(HTTP,JMS) it supports? without this info, It is hard for any client who wants to consume the particular web service to construct the SOAP message. Even if they do, they won't be sure, it'll work all the time. WSDL is the rescue. WSDL (Web Service description Language) defines the operations, message formats and transport details for the SOAP message.

Restful  
78.What is Restful Services ?what the Rest principles are there?

## Principles of Rest Services:

### Uniform Interface:

In the rest architecture everything is resource. Everything means a document, picture, Video or even module, class, method in the service program that is available to service consumer. The consumer can access any resource using just URI. REST make explicit use of HTTP methods to perform different operations in resources. We can delete a resource using DELETE method of HTTP protocol. Similarly we can use GET, POST and PUT to access, add and modify resources. Most Important as well best thing about rest is you can perform different operation on resources using different HTTP methods but with same URI. So you can relate the HTTP methods as verbs and resources as nouns.

Second thing is the resources have some representations like XML, JSON, text or TEXT and we can modify the representation as per our requirement. Client can select the resource representation. REST request and responses are self descriptive that means these not only contain data but also contain information like whether it should be cached or not, authentication required or not etc.. This principle of rest architecture is also known as address ability.

So In conclusion there must be a unique URI for a resource in web service. Naming conventions for uri is same as the normal web uri. I want to let you understand through small example. Lets suppose we have a resource which have information of the students. So in normal way we have /getstudentInfo URI to access the student information and /addstudentInfo uri to add the information of new student. But According to REST you can use a single URI for student information resource like /studentInfo. Here You can get information with the same URI with get HTTP method and add information using POST HTTP method. So there is a single URI for single resource.

### Statelessness:

It is very simple concept. Stateless in REST means server do not store any information about client session information. This makes server more scalable and reduce work load. You may think being statelessness is odd for this time but if you look back one and half decades back servers not used to store session data. Fat clients send the necessary state related data with each request. Same thing we can implement in REST API. If server need client session data, then client will send that data with subsequent request.

### Use of cache:

Server can use the cache to save the frequent response to reduce load. Caching is possible using cache, proxy or reverse cache therefore any coming request will go through these cache. GET requests are most supposed to be cached. But we can not cache put and delete methods. We rarely cache the responses for POST requests. In the cached responses header may contain information about date of response generation, expire date, a flag to represent change in resource and etc. It is a good practice to cache the static contents like images and graphics content, CSS or JavaScript files. We should not set expire date very high. If you are caching dynamic content, Expire time should be few hour. Use of cache reduce bandwidth, latency as well as load on server.

### Layered System:

Rest allows the series of servers and hide this from service consumers. For example You can keep API s in a server, Database in another server and authentication system in separate server. Client do not knows the numbers of servers that responses come through.

### Code on Demand:

This term has relation with representation. We can ask for JSON, XML or YAML from server. If client is Ajax or JS it may need json, Or if client is JAVA based it may require XML and if client program is written in ruby then they may require YAML. You may surprised that client can ask for different representation of the resources without changing the URI. Client send the media type with request header to specify for the required representation of resource. You can add header Accept and pass representation values like application/XML or application/json with it.

### HATEOAS:

HATEOAS stands for Hypermedia As The Engine Of Application State. Just like you get hyperlinks as response to change the state of browser, Its possible to get hyperlinks in rest response to change the state of client application. Generally XML based REST services follow the syntax from [**Atom**](https://en.wikipedia.org/wiki/Atom_(standard)). Atom is XML based document format. Now we can publish a main URI to users and That service can return other service resource URIs on response. So this makes transparency in the resources. Another advantage of using HATEOAS is that we can reduce application state transition errors. Error codes are not always sufficient to know that what happened in the resource. So in such case we can send links with error information. It even decouple the information about client server interaction.

79. How to make the class as Resource class?

## Creating a RESTful Root Resource Class

**Root resource classes** are POJOs that are either annotated with @Path or have at least one method annotated with @Path or a **request method designator**, such as @GET, @PUT, @POST, or @DELETE. **Resource methods** are methods of a resource class annotated with a request method designator. This section explains how to use JAX-RS to annotate Java classes to create RESTful web services.

|  |  |
| --- | --- |
| **Annotation** | **Description** |
| @Path | The @Path annotation’s value is a relative URI path indicating where the Java class will be hosted: for example, /helloworld. You can also embed variables in the URIs to make a URI path template. For example, you could ask for the name of a user and pass it to the application as a variable in the URI: /helloworld/{username}. |
| @GET | The @GET annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP GET requests. The behavior of a resource is determined by the HTTP method to which the resource is responding. |
| @POST | The @POST annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP POST requests. The behavior of a resource is determined by the HTTP method to which the resource is responding. |
| @PUT | The @PUT annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP PUT requests. The behavior of a resource is determined by the HTTP method to which the resource is responding. |
| @DELETE | The @DELETE annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP DELETE requests. The behavior of a resource is determined by the HTTP method to which the resource is responding. |
| @HEAD | The @HEAD annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP HEAD requests. The behavior of a resource is determined by the HTTP method to which the resource is responding. |
| @PathParam | The @PathParam annotation is a type of parameter that you can extract for use in your resource class. URI path parameters are extracted from the request URI, and the parameter names correspond to the URI path template variable names specified in the @Path class-level annotation. |
| @QueryParam | The @QueryParam annotation is a type of parameter that you can extract for use in your resource class. Query parameters are extracted from the request URI query parameters. |
| @Consumes | The @Consumes annotation is used to specify the MIME media types of representations a resource can consume that were sent by the client. |
| @Produces | The @Produces annotation is used to specify the MIME media types of representations a resource can produce and send back to the client: for example, "text/plain". |
| @Provider | The @Provider annotation is used for anything that is of interest to the JAX-RS runtime, such as MessageBodyReader and MessageBodyWriter. For HTTP requests, the MessageBodyReader is used to map an HTTP request entity body to method parameters. On the response side, a return value is mapped to an HTTP response entity body by using a MessageBodyWriter. If the application needs to supply additional metadata, such as HTTP headers or a different status code, a method can return a Response that wraps the entity and that can be built using Response.ResponseBuilder. |

80.Which API is used in project to implement restful Services?

Jersey

Jersey is the open source, production quality, JAX-RS (JSR 311) Reference **Implementation** for building **RESTful** Web **services**.

81.Difference between Path Param and Query Param?

Use @PathParam for retrieval based on id. User @QueryParam for filter or if you have any fixed list of options that user can pass.

This is what I do.

If there is a scenario to retrieve a record based on id, for example you need to get the details of the employee whose id is 15, then you can have resource with @PathParam.

GET /employee/{id}

If there is a scenario where you need to get the details of all employees but only 10 at a time, you may use query param

GET /employee?start=1&size=10

This says that starting employee id 1 get ten records.

To summarize, use @PathParam for retrieval based on id. User @QueryParam for filter or if you have any fixed list of options that user can pass.

82.What are Http methods are there?

## HTTP GET

Use GET requests **to retrieve resource representation/information only** – and not to modify it in any way. As GET requests do not change the state of the resource, these are said to be **safe methods**. Additionally, GET APIs should be **idempotent**, which means that making multiple identical requests must produce the same result every time until another API (POST or PUT) has changed the state of the resource on the server.

If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process.

For any given HTTP GET API, if the resource is found on the server, then it must return HTTP response code 200 (OK) – along with the response body, which is usually either XML or JSON content (due to their platform-independent nature).

In case resource is NOT found on server then it must return HTTP response code 404 (NOT FOUND). Similarly, if it is determined that GET request itself is not correctly formed then server will return HTTP response code 400 (BAD REQUEST).

#### Example request URIs

* HTTP GET http://www.appdomain.com/users
* HTTP GET http://www.appdomain.com/users?size=20&page=5
* HTTP GET http://www.appdomain.com/users/123
* HTTP GET http://www.appdomain.com/users/123/address

## HTTP POST

Use POST APIs **to create new subordinate resources**, e.g., a file is subordinate to a directory containing it or a row is subordinate to a database table. Talking strictly in terms of REST, POST methods are used to create a new resource into the collection of resources.

Ideally, if a resource has been created on the origin server, the response SHOULD be HTTP response code 201 (Created) and contain an entity which describes the status of the request and refers to the new resource, and a [Location](https://en.wikipedia.org/wiki/HTTP_location) header.

Many times, the action performed by the POST method might not result in a resource that can be identified by a URI. In this case, either HTTP response code 200 (OK) or 204 (No Content) is the appropriate response status.

Responses to this method are **not cacheable**, unless the response includes appropriate [Cache-Control](https://en.wikipedia.org/wiki/Web_cache#Cache_control) or [Expires](https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html) header fields.

Please note that POST is **neither safe nor idempotent**, and invoking two identical POST requests will result in two different resources containing the same information (except resource ids).

#### Example request URIs

* HTTP POST http://www.appdomain.com/users
* HTTP POST http://www.appdomain.com/users/123/accounts

## HTTP PUT

Use PUT APIs primarily **to update existing resource** (if the resource does not exist, then API may decide to create a new resource or not). If a new resource has been created by the PUT API, the origin server MUST inform the user agent via the HTTP response code 201 (Created) response and if an existing resource is modified, either the 200 (OK) or 204 (No Content) response codes SHOULD be sent to indicate successful completion of the request.

If the request passes through a cache and the Request-URI identifies one or more currently cached entities, those entries SHOULD be treated as stale. Responses to this method are **not cacheable**.

The difference between the POST and PUT APIs can be observed in request URIs. POST requests are made on resource collections, whereas PUT requests are made on an individual resource.

#### Example request URIs

* HTTP PUT http://www.appdomain.com/users/123
* HTTP PUT http://www.appdomain.com/users/123/accounts/456

## HTTP DELETE

As the name applies, DELETE APIs are used **to delete resources** (identified by the Request-URI).

A successful response of DELETE requests SHOULD be HTTP response code 200 (OK) if the response includes an entity describing the status, 202 (Accepted) if the action has been queued, or 204 (No Content) if the action has been performed but the response does not include an entity.

DELETE operations are **idempotent**. If you DELETE a resource, it’s removed from the collection of resources. Repeatedly calling DELETE API on that resource will not change the outcome – however calling DELETE on a resource a second time will return a 404 (NOT FOUND) since it was already removed. Some may argue that it makes the DELETE method non-idempotent. It’s a matter of discussion and personal opinion.

If the request passes through a cache and the Request-URI identifies one or more currently cached entities, those entries SHOULD be treated as stale. Responses to this method are **not cacheable**.

#### Example request URIs

* HTTP DELETE http://www.appdomain.com/users/123
* HTTP DELETE http://www.appdomain.com/users/123/accounts/456

## HTTP PATCH

HTTP PATCH requests are **to make partial update on a resource**. If you see PUT requests also modify a resource entity so to make more clear – PATCH method is the correct choice for partially updating an existing resource and PUT should only be used if you’re replacing a resource in its entirety.

Please note that there are some challenges if you decide to use PATCH APIs in your application:

* Support for PATCH in browsers, servers, and web application frameworks is not universal. IE8, PHP, Tomcat, Django, and lots of other software has missing or broken support for it.
* Request payload of PATCH request is not straightforward as it is for PUT request. e.g.

HTTP GET /users/1

produces below response:

{id: 1, username: 'admin', email: 'email@example.org'}

A sample patch request to update the email will be like this:

HTTP PATCH /users/1

[  
{ “op”: “replace”, “path”: “/email”, “value”: “new.email@example.org” }  
]

There may be following possible operations are per the HTTP specification.

[  
{ "op": "test", "path": "/a/b/c", "value": "foo" },  
{ "op": "remove", "path": "/a/b/c" },  
{ "op": "add", "path": "/a/b/c", "value": [ "foo", "bar" ] },  
{ "op": "replace", "path": "/a/b/c", "value": 42 },  
{ "op": "move", "from": "/a/b/c", "path": "/a/b/d" },  
{ "op": "copy", "from": "/a/b/d", "path": "/a/b/e" }  
]

PATCH method is not a replacement for the POST or PUT methods. It applies a delta (diff) rather than replacing the entire resource.

## Summary of HTTP Methods for RESTful APIs

The below table summarises the use of HTTP methods discussed above.

| **HTTP Method** | **CRUD** | **Entire Collection (e.g. /users)** | **Specific Item (e.g. /users/123)** |
| --- | --- | --- | --- |
| POST | Create | 201 (Created), ‘Location’ header with link to /users/{id} containing new ID. | Avoid using POST on single resource |
| GET | Read | 200 (OK), list of users. Use pagination, sorting and filtering to navigate big lists. | 200 (OK), single user. 404 (Not Found), if ID not found or invalid. |
| PUT | Update/Replace | 404 (Not Found), unless you want to update every resource in the entire collection of resource. | 200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid. |
| PATCH | Partial Update/Modify | 404 (Not Found), unless you want to modify the collection itself. | 200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid. |
| DELETE | Delete | 404 (Not Found), unless you want to delete the whole collection — use with caution. | 200 (OK). 404 (Not Found), if ID not found or invalid. |

82.Difference put and Post?why post is idempotent?

814

**HTTP PUT:**

PUT puts a file or resource at a specific URI, and exactly at that URI. If there's already a file or resource at that URI, PUT replaces that file or resource. If there is no file or resource there, PUT creates one. PUT is [idempotent](http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.1.2), but paradoxically PUT responses are not cacheable.

[HTTP 1.1 RFC location for PUT](http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.6)

**HTTP POST:**

POST sends data to a specific URI and expects the resource at that URI to handle the request. The web server at this point can determine what to do with the data in the context of the specified resource. The POST method is not [idempotent](http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.1.2), however POST responses *are* cacheable so long as the server sets the appropriate Cache-Control and Expires headers.

The official HTTP RFC specifies POST to be:

* Annotation of existing resources;
* Posting a message to a bulletin board, newsgroup, mailing list, or similar group of articles;
* Providing a block of data, such as the result of submitting a form, to a data-handling process;
* Extending a database through an append operation.

Simply put, an operation is idempotent if it produces the same result when called over and over. An identical request should return an identical result when done twice, two thousand, or two million times. The source of most confusion around this concept comes with the idea of identical results, however. What we expect to see is identical results in the return form rather than in the return value.

When a call is issued, the response can be broken generally into two different camps – the **return form**, and the **return value**. If we were to make a GET request for a resource status update, for instance, we might get vastly different results depending on the resource that is being inspected and the time in which we do the inspection.

**POST is not idempotent**, so making a **POST** request more than one time may have additional side effects, like creating a second, third and fourth programmer. But the key word here is may. Just because an endpoint uses **POST** doesn't mean that it must have side effects on every request. It just might have side effects.

83.How to read JSON and XML data in Restful resource?

21

The same resource may return either XML or JSON depending upon the request, but it shouldn't return both at the same time. You will know which one to return based upon the request, so there is no need to generate both -- just generate the one you'll be returning.

Here is how you might choose to decide which to return. Evaluate in order, stopping when you've determined the format to return:

1. If an extension has been added to the resource (GET /user/1234.json or GET /user/1234.xml), use that as the requested format.
2. If an Accept header is set, use that header's value as the requested format.
3. If there is a request body (as in the case of a POST), and the Content-Type header specifies JSON or XML, use that.
4. Use a default format if none of the above apply (generally use JSON as your default unless your customers are generally still using XML).

JUNIT  
84.What is junit?what the code for Junit test cases.

JUnit is a unit testing framework for Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks collectively known as xUnit, that originated with JUnit.

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Annotations** | **Description** |
| 1. | @Test | This annotation is a replacement of org.junit.TestCase which indicates that public void method to which it is attached can be executed as a test Case. |
| 2. | @Before | This annotation is used if you want to execute some statement such as preconditions before each test case. |
| 3. | @BeforeClass | This annotation is used if you want to execute some statements before all the test cases for e.g. test connection must be executed before all the test cases. |
| 4. | @After | This annotation can be used if you want to execute some statements after each[Test Case](https://www.guru99.com/test-case.html)for e.g resetting variables, deleting temporary files ,variables, etc. |
| 5. | @AfterClass | This annotation can be used if you want to execute some statements after all test cases for e.g. Releasing resources after executing all test cases. |
| 6. | @Ignores | This annotation can be used if you want to ignore some statements during test execution for e.g. disabling some test cases during test execution. |
| 7. | @Test(timeout=500) | This annotation can be used if you want to set some timeout during test execution for e.g. if you are working under some SLA (Service level agreement), and tests need to be completed within some specified time. |
| 8. | @Test(expected=IllegalArgumentException.class) | This annotation can be used if you want to handle some exception during test execution. For, e.g., if you want to check whether a particular method is throwing specified exception or not. |

Let's discuss on this with an example:

@BeforeClass

public static void m1() {

System.out.println("Using @BeforeClass , executed before all test cases ");

}

@Before

public void m2() {

list = new ArrayList<String>();

System.out.println("Using @Before annotations ,executed before each test cases ");

}

@AfterClass

public static void m3() {

System.out.println("Using @AfterClass ,executed after all test cases");

}

@After

public void m4() {

list.clear();

System.out.println("Using @After ,executed after each test cases");

}

@Test

public void m5() {

list.add("test");

assertFalse(list.isEmpty());

assertEquals(1, list.size());

}

@Ignore

public void m6() {

System.out.println("Using @Ignore , this execution is ignored");

}

@Test(timeout = 10)

public void m7() {

System.out.println("Using @Test(timeout),it can be used to enforce timeout in JUnit4 test case");

}

@Test(expected = NoSuchMethodException.class)

public void m8() {

System.out.println("Using @Test(expected) ,it will check for specified exception during its execution");

}

}

85.What is Mockito unit Testing?

**Mockito** is a mocking framework, JAVA-based library that is used for effective unit testing of JAVA applications. **Mockito** is used to mock interfaces so that a dummy functionality can be added to a mock interface that can be used in unit testing.

86.Write the test cases to test the Layered Applications?  
Maven  
87.What is Maven? and Maven Life Cycle?

**Maven** is a powerful project management tool that is based on POM (project object model). It is **used** for projects build, dependency and documentation. It simplifies the build process like ANT. ... In short terms we can tell **maven** is a tool that can be **used** for building and managing any Java-based project.

## **What is Build Lifecycle?**

A Build Lifecycle is a well-defined sequence of phases, which define the order in which the goals are to be executed. Here phase represents a stage in life cycle. As an example, a typical **Maven Build Lifecycle** consists of the following sequence of phases.

|  |  |  |
| --- | --- | --- |
| **Phase** | **Handles** | **Description** |
| prepare-resources | resource copying | Resource copying can be customized in this phase. |
| validate | Validating the information | Validates if the project is correct and if all necessary information is available. |
| compile | compilation | Source code compilation is done in this phase. |
| Test | Testing | Tests the compiled source code suitable for testing framework. |
| package | packaging | This phase creates the JAR/WAR package as mentioned in the packaging in POM.xml. |
| install | installation | This phase installs the package in local/remote maven repository. |
| Deploy | Deploying | Copies the final package to the remote repository. |

There are always **pre** and **post** phases to register **goals**, which must run prior to, or after a particular phase.

When Maven starts building a project, it steps through a defined sequence of phases and executes goals, which are registered with each phase.

Maven has the following three standard lifecycles −

* clean
* default(or build)
* site

A **goal** represents a specific task which contributes to the building and managing of a project. It may be bound to zero or more build phases. A goal not bound to any build phase could be executed outside of the build lifecycle by direct invocation.

The order of execution depends on the order in which the goal(s) and the build phase(s) are invoked. For example, consider the command below. The **clean** and **package** arguments are build phases while the **dependency:copy-dependencies** is a goal.

mvn clean dependency:copy-dependencies package

88.How to resolve conflict between dependency?

Maven can handle both situations without any conflict. Conflicts will exist when two versions of a transitive dependency are required. The ClassNotFoundException you describe results from the app (or a dependency) attempting to use a class not available in the version of the conflicted dependency that actually gets used. There are multiple ways to fix the problem.

1. Update the versions of the libraries you are using that depend on the conflicted dependency, so that they all depend on the same version version of that dependency
2. Declare the conflicted dependency as a direct dependency of your project with the version you want to be included (in the example, the one with the missing class included in it)
3. Specify which version of the conflicted dependency that transitive dependencies should use, via the <dependencyManagement> section of the POM
4. Explicitly exc

GIT  
89.What is GIT and what is the difference between GIT and SVN?

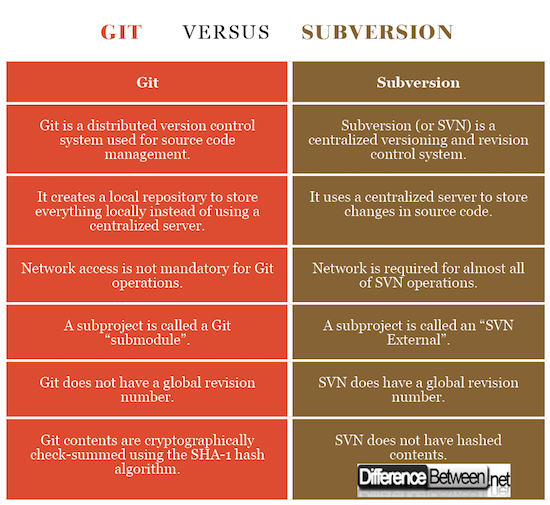
Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

SVN is a Centralized Version Control System (CVCS), and Git is a Distributed Version Control System (DVCS).

A centralized version control system operates on the basic idea that there is one single copy of the project that developers will commit changes to, and where all version of the project are stored.

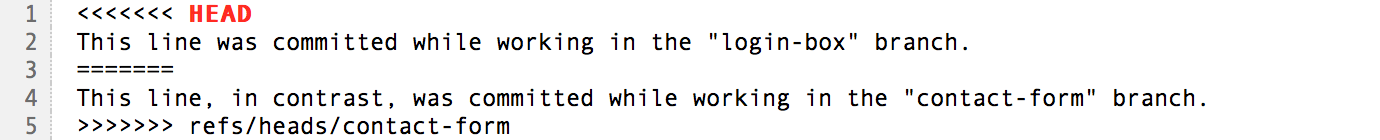
A distributed version control system, however, works on the principle that each developer “clones” the project repository to their hard drive. A copy of the project is stored on every developer’s local machine, and changes are either “pushed” up to the online repository, or “pulled” down from the repo to update the version that the developer has on their machine.

1. **Trunk:**The trunk is the hub of your current, stable code and product. It only includes tested, unbroken code.
2. **Branches:**Here is where you house new code and features. Using a copy of the trunk code, team members conduct research and development in the branch. Doing so allows each team member to work on the enhanced features without disrupting each other’s progress.
3. **Tags:**Consider tags a duplicate of a branch at a given point in time. Tags aren’t used during development, but rather during deployment after the branch’s code is finished. Marking your code with tags make it easy to review and, if necessary, revert your code.



90.How to resolve Confict between GIT?

**A Conflict is Just an Annotation**  
It helps to realize that a conflict is nothing magical. In the concerned file, Git simply marks the areas that were edited in contradictory ways:



91.What is Jenkins?

Jenkins is a free and open source automation server. Jenkins helps to automate the non-human part of the software development process, with continuous integration and facilitating technical aspects of continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat.

92.What is JIRA?

**JIRA is** a tool developed by Australian Company Atlassian. It **is** used for bug tracking, issue tracking, and project management. The name "**JIRA**" **is** actually inherited from the Japanese word "Gojira" which means "Godzilla". The basic use of this tool **is** to track issue and bugs related to your **software** and Mobile apps

93.How to deploy project and what Servers are used?  
94.What is elastic Search?

**ElasticSearch** is an Open-source Enterprise REST based Real-time **Search** and Analytics Engine. It's core **Search** Functionality is built using Apache Lucene, but supports many other features. It is written in **Java** Language. It supports Store, Index, **Search** and Analyze Data in Real-time

95.How to optimize the performace of your application

### **1. Don’t optimize before you know it’s necessary**

That might be one of the most important performance tuning tips. You should follow common best practices and try to implement your use cases efficiently.

But that doesn’t mean that you should replace any standard libraries or build complex optimizations before you proved that it’s necessary.

In most cases, [premature optimization](https://stackify.com/premature-optimization-evil/) takes up a lot of time and makes the code hard to read and maintain.



And to make it even worse, these optimizations most often don’t provide any benefits because you’re spending a lot of time optimizing non-critical parts of your application.

So, how do you prove that you need to optimize something?

First of all, you need to define how fast your application code has to be, e.g., by specifying a maximum response time for all API calls or the number of records that you want to import within a specified time frame.

After you’ve done that, you can measure which parts of your application are too slow and need to be improved. And when you’ve done that, you should take a look at the second tip.

### **2. Use a profiler to find the real bottleneck**

After you followed the first recommendation and identified the parts of your application you need to improve, ask yourself where to start?

You can approach this question in two ways:

* You can take a look at your code and start with the part that looks suspicious or where you feel that it might create problems.
* Or you[use a profiler](https://stackify.com/prefix/) and get detailed information about the behavior and performance of each part of your code.

I hope I don’t need to explain why you should always follow the second approach.

It should be obvious that the profiler-based method gives you a better understanding of the performance implications of your code and allows you to focus on the most critical parts.

And if you ever used a profiler, you will remember a few situations in which you were surprised by which parts of your code created the performance issues. More than once my first guess would have led me in the wrong direction.

### **3. Create a performance test suite for the whole application**

This is another general tip that helps you avoid a lot of unexpected problems that often occur after you have deployed your performance improvement to production.

You should always define a performance test suite that tests the whole application, and run it before and after you worked on a performance improvement.

These additional test runs will help you to identify the functional and performance side effects of your change and make sure that you don’t ship an update that caused more harm than good.

That is especially important if you work on components that are used by several different parts of your application, like databases or caches.

### **4. Work on the biggest bottleneck first**

And after you have created your test suite and analyzed your application with a[profiler](https://stackify.com/prefix/), you have a list of issues you want to address to improve the performance.

That’s good, but it still doesn’t answer the question where you should start. You could focus on the quick wins, or start with the most significant issue.

It might be tempting to start with the quick wins because you will be able to show first results soon. Sometimes, that might be necessary to convince other team members or your management that the performance analysis was worth the effort.



But in general, I recommend starting at the top and begin work on the most significant performance problem first.

That will provide you with the biggest performance improvement, and you might not need to fix more than a few of these issues to fulfill your performance requirements.

Enough about general performance tuning tips. Let’s take a closer look at some Java-specific ones.

### **5. Use StringBuilder to concatenate Strings programmatically**

There are lots of different options to concatenate Strings in Java. You can, for example, use a simple + or +=, the good old StringBuffer or a[StringBuilder](https://docs.oracle.com/javase/9/docs/api/java/lang/StringBuilder.html).

So, which approach should you prefer?

The answer depends on the code that concatenates the String. If you’re programmatically adding new content to your String, e.g., in a for-loop, you should use the StringBuilder.

It’s easy to use and provides better performance than StringBuffer. But please keep in mind, that the StringBuilder, in contrast to StringBuffer, is not thread-safe and might not be a good fit for all use cases.

You just need to instantiate a new StringBuilder and call the append method to add a new part to the String. And when you’ve added all parts, you can call the toString() method to retrieve the concatenated String.

The following code snippet shows a simple example. During each iteration, this loop converts i into a String and adds it together with a space to the StringBuilder sb. So, in the end, this code writes “This is a test0 1 2 3 4 5 6 7 8 9” to the log file.

StringBuilder sb = new StringBuilder(“This is a test”);  
for (int i=0; i<10; i++) {  
sb.append(i);  
sb.append(” “);  
}  
log.info(sb.toString());

As you can see in the code snippet, you can provide the first element of your String to the constructor method.

That will create a new StringBuilder containing the provided String and a capacity for 16 additional characters. When you add more characters to the StringBuilder, your JVM will dynamically increase the size of the StringBuilder.

If you already know how many characters your String will contain, you can provide that number to different constructor method to instantiate a StringBuilder with the defined capacity.

That improves its efficiency even further because it doesn’t need to dynamical extend its capacity.

### **6. Use + to concatenate Strings in in one statement**

When you implemented your first application in Java, someone probably told you that you shouldn’t concatenate Strings with +. And that’s correct if you’re concatenating Strings in your application logic.

Strings are immutable, and the result of each String concatenation is stored in a new String object. That requires additional memory and slows down your application, especially if you’re concatenating multiple Strings within a loop.

In these cases, you should follow tip number 5 and use a StringBuilder.

But that’s not the case if you’re just breaking a String into multiple lines to improve the readability of your code.

Query q = em.createQuery(“SELECT a.id, a.firstName, a.lastName “  
+ “FROM Author a “  
+ “WHERE a.id = :id”);

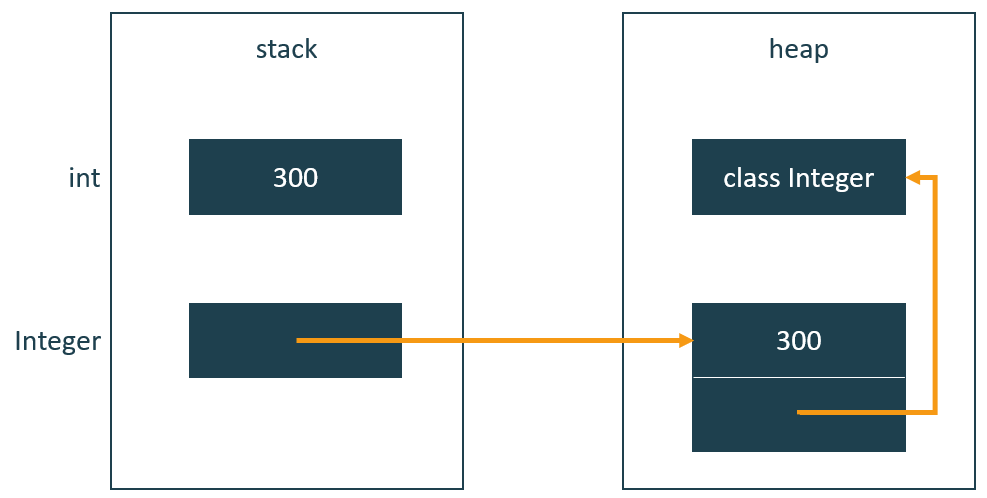
In these situations, you should concatenate your Strings with a simple +. Your Java compiler will optimize this and perform the concatenation at compile time.

So, at runtime, your code will just use 1 String, and no concatenation will be required.

### **7. Use primitives where possible**

Another quick and easy way to avoid any overhead and improve the performance of your application is to use primitive types instead of their wrapper classes.

So, it’s better to use an int instead of an Integer, or a double instead of a Double. That allows your [JVM](https://stackify.com/jvm-metrics/) to[store the value in the stack instead of the heap](https://www.javaworld.com/article/2150208/java-language/a-case-for-keeping-primitives-in-java.html) to reduce memory consumption and overall handle it more efficiently.



### **8. Try to avoid**BigInteger**and**BigDecimal

As we’re already talking about data types, we should also take a quick look at[BigInteger](https://docs.oracle.com/javase/9/docs/api/java/math/BigInteger.html) and[BigDecimal](https://docs.oracle.com/javase/9/docs/api/java/math/BigDecimal.html). Especially the latter one is popular because of its precision. But that comes at a price.

BigInteger and BigDecimal require much more memory than a simple long or double and slow down all calculations dramatically.

So, better think twice if you need the additional precision, or if your numbers will exceed the range of a long.

This might be the only thing you need to change to fix your performance problems, especially if you’re implementing a mathematical algorithm.

### **9. Check the current log level first**

This recommendation should be obvious, but unfortunately, you can find lots of code that ignores it. Before you create a debug message, you should always check the current log level first.

Otherwise, you might create a String with your[log message](https://stackify.com/log-management/) that will be ignored afterward.

Here are 2 examples of how you should NOT do it.

// don’t do this  
log.debug(“User [” + userName + “] called method X with [” + i + “]”);  
// or this  
log.debug(String.format(“User [%s] called method X with [%d]”, userName, i));

In both cases, you will perform all required steps to create the log message without knowing if your logging framework will use the log message.

It’s better to check the current log level first before you create the debug message.

// do this  
if (log.isDebugEnabled()) {  
log.debug(“User [” + userName + “] called method X with [” + i + “]”);  
}

### **10. Use Apache Commons**StringUtils.replace**instead of**String.replace

In general, the String.replace method works fine and is pretty efficient, especially if you’re using Java 9.

But if your application requires a lot of replace operations and you haven’t updated to the newest Java version, it still makes sense to check for faster and more efficient alternatives.

One candidate is[Apache Commons Lang’s](https://commons.apache.org/proper/commons-lang/)[StringUtils.replace](https://commons.apache.org/proper/commons-lang/javadocs/api-release/org/apache/commons/lang3/StringUtils.html#replace-java.lang.String-java.lang.String-java.lang.String-) method. As Lukas Eder described in[one of his recent blog posts](https://blog.jooq.org/2017/10/11/benchmarking-jdk-string-replace-vs-apache-commons-stringutils-replace/), it dramatically outperforms Java 8’s String.replace method.

And it just requires a minimal change. You need to add a [Maven](https://stackify.com/gradle-vs-maven/) dependency for the Apache’s Commons Lang project to your application pom.xml, and replace all calls of the String.replace method with the StringUtils.replace method.

// replace this  
test.replace(“test”, “simple test”);

// with this  
StringUtils.replace(test, “test”, “simple test”);

### **11. Cache expensive resources, like database connections**

Caching is a popular solution to avoid the repeated execution of expensive or frequently used code snippets.

The general idea is simple: Reusing such resources is cheaper than creating a new one over and over again.

A typical example is caching database connections in a pool. The creation of a new connection takes time, which you can avoid if you reuse an existing connection.

You can also find other examples in the Java language itself. The valueOf method of the Integer class, for example, caches the values between -128 and 127.

You might say that the creation of a new Integer isn’t too expensive, but it’s used so often that the caching of the most used values provides a performance benefit.



But when you think about caching, please keep in mind that your caching implementation also creates an overhead.

You need to spend additional memory to store the reusable resources, and you might need to manage your cache to make the resources accessible or to remove outdated ones.

So, before you start caching any resources, make sure that you use them often enough to outweigh the overhead of your cache implementation.

Concurrency API  
96.What is Concurrency API?

The **concurrent** collection **APIs**, apart from the Java Collection **API**, are a set of collections **APIs** that are designed and optimized specifically for synchronized multithreaded access. They are grouped under the java. util. **concurrent** package.  
97.What is the difference between Runnable and Callable?

81

* A Callable needs to implement call() method while a Runnable needs to implement run() method.
* A Callable can return a value but a Runnable cannot.
* A Callable can throw checked exception but a Runnable cannot.
* A Callable can be used with ExecutorService#invokeXXX(Collection<? extends Callable<T>> tasks) methods but a Runnable cannot be.
* public interface Runnable {
* void run();
* }
* public interface Callable<V> {
* V call() throws Exception;

}

98.What is Future Object?  
99.What is the Pooled Connection Pooling?

**Connection pooling** is a pattern used by software applications to **connect** to databases using a pre-created set of reusable **connection** objects. When a new **connection** is required, an existing **connection** is retrieved from the **pool**. ... Many **Java** Application Frameworks include their own **connection pooling** APIs.

## C3P0

## Proxool

JAVA 8  
100.What are features of there Java 8?

1. Lambda Expressions
2. Pipelines and Streams
3. Date and Time API
4. Default Methods
5. Type Annotations
6. Nashorn JavaScript Engine
7. Concurrent Accumulators
8. Parallel operations
9. PermGen Space Removed
10. TLS SNI

101.What is Lambda Expression?an d What is Functional Interface?  
102.What is Stream?and What Stream Pipeline operations(write the code).  
103.Changed Internal of Hashmap in Java?  
104.Why Static and default methods are Provided in Java 8?

Spring Boot  
105.What is difference between Spring MVC and Spring Boot

## What is Spring boot?

There were many difficulties faced by Spring programmers in the process of setting up Hibernate Datasource, Session Factory, Entity Manager, and Transaction Management. The Spring framework proved to be slow, cumbersome, and time-consuming for developers to set up any basic project that needed Spring MVC as it provided minimum functionality. Spring Boot is an extension of the popular Spring framework that was launched to eliminate the necessary boilerplate configurations for building Spring applications.

### Feature of Spring boot

Spring Boot provides an opinionated view of the overall Spring platform that has laid down the base for a more effective and faster development eco-system. **Some important features of Spring Boot are:**

* Spring boot offers opinionated ‘starter’ dependencies for the sake of simplifying the build and application configurations of the Spring framework.
* It offers an embedded server for avoiding the complexities of application deployment.
* Spring Boot presents production-ready features of the likes of health check-up, metrics gathering, auditing, HTTP tracing, etc.
* It presents an automatic configuration for Spring functionality as and when possible. There is absolutely no need for the manual configuration of static resource mappings, dispatcher servlet, message converters, property source loaders, etc.
* The Dependency Management feature of Spring Boot relates to various versions of commonly used libraries that are categorized and pre-selected in different starter POMs for being included in projects. The selection of any Spring Boot version is akin to the developer selecting dozens of various dependencies that would have to be chosen otherwise and harmonized manually.
* In Spring Boot, the feature of Advanced Externalized Configuration offers a big list of bean-specific properties that are capable of being configured via the application. Properties file, even without the inclusion of java or xml config.
* Spring Boot packages applications as a runnable jar that have embedded tomcat features for being presented as self-contained deployment units.

## Key difference between Spring MVC and Spring boot

Spring MVC serves as a fully-equipped HTTP oriented MVC framework that’s managed by the popular Spring Framework; it has its base in Servlets. Spring MVC can be equated to JSF in the JavaEE stack. In Spring MVC, the most popular elements are in the form of classes annotated with **@Controller**. These classes help implement methods that may be accessed using various HTTP requests. Spring MVC has an equivalent in the form of **@RestController**for the cause of implementing REST-based APIs.

Spring Boot is a vital utility that helps in quick and efficient setting up of applications. It provides out of the box configurations with a view of building different types of Spring-powered applications. Spring MVC is a Spring framework that is used by developers for creating web applications. On the other hand, Spring boot mainly serves as a useful Spring-based, production-ready initialize for projects. This is one of the main differences between spring and spring boot.

**Spring:**

**1.** The main difference lies in**"Test-ability."**  
**2.** Spring comes packaged with DI and IOC that automate all the hard work and cut out the need for manual interventions to a huge extent. So, even as Spring app developers define class objects manually, they may want to use Di for annotating with **@Service** or **@Component** for matching and managing classes.  
**3.** The **@Autowired** annotation allows for the mock() function at unit testing time.  
**4.** Unlike in JDBC, where the same code has to be written multiple times for the performance of different types of database operations, Spring goes about the act via Hibernate and ORM.  
**5.** The Spring framework provides efficient integration with structures such as Hibernate, ORM, Junit, and Mockito.

**Spring Boot:**

**1.**It allows the creation of Quick Application that includes different microservices complete with their scope and capabilities. This feature of Spring Boot eliminates the need for managing a single, big web application.  
**2.** In the standard Spring framework, there are several configurations of the likes of Component Scan, DispatcherServlet, View Resolver, XMLs, Web Jar, etc. On the other hand, Spring Boot allows for automatic configurations using pre-defined classpaths.  
**3.** Spring Boot presents Default Spring Starters featuring default Spring configuration dependencies of the kinds of Spring Core, Jackson, Web-MVC, Tomcat, Data Binding, Validation, Logging, etc. Developers need not worry about versioning issues in this case.

106.How to configure Multiple data Sources in Spring Boot?

#first db

spring.datasource.url = [url]

spring.datasource.username = [username]

spring.datasource.password = [password]

spring.datasource.driverClassName = oracle.jdbc.OracleDriver

#second db ...

spring.secondDatasource.url = [url]

spring.secondDatasource.username = [username]

spring.secondDatasource.password = [password]

spring.secondDatasource.driverClassName = oracle.jdbc.OracleDriver

@Bean

@Primary

@ConfigurationProperties(prefix="spring.datasource")

public DataSource primaryDataSource() {

return DataSourceBuilder.create().build();

}

@Bean

@ConfigurationProperties(prefix="spring.secondDatasource")

public DataSource secondaryDataSource() {

return DataSourceBuilder.create().build();

}

@Configuration

@EnableTransactionManagement

@EnableJpaRepositories(

entityManagerFactoryRef = "customerEntityManagerFactory",

transactionManagerRef = "customerTransactionManager",

basePackages = {

"com.javadevjournal.customer.repo"

}  
JavaScript & JQuery  
107.What is closure in Java Script?  
108.How to write validation logic in JavaScript?  
109.How to write validation logic in JQuery?  
AJAX  
110.What is AJAX?How to implement it?

**AJAX** stands for Asynchronous JavaScript and XML. **AJAX** is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script

**AJAX** uses JavaScript to send and receive data between a web browser and a web server. The **AJAX** technique makes web pages more responsive by exchanging data with the web server behind the scenes, instead of reloading an entire web page each time a user makes a change.

function searchViaAjax() {

var search = {}

search["username"] = $("#username").val();

search["email"] = $("#email").val();

$.ajax({

type : "POST",

contentType : "application/json",

url : "${home}search/api/getSearchResult",

data : JSON.stringify(search),

dataType : 'json',

timeout : 100000,

success : function(data) {

console.log("SUCCESS: ", data);

display(data);

},

error : function(e) {

console.log("ERROR: ", e);

display(e);

},

done : function(e) {

console.log("DONE");

enableSearchButton(true);

}

});

}

--

@RestController

public class AjaxController {

List<User> users;

// @ResponseBody, not necessary, since class is annotated with @RestController

// @RequestBody - Convert the json data into object (SearchCriteria) mapped by field name.

// @JsonView(Views.Public.class) - Optional, filters json data to display.

@JsonView(Views.Public.class)

@RequestMapping(value = "/search/api/getSearchResult")

public AjaxResponseBody getSearchResultViaAjax(@RequestBody SearchCriteria search) {

AjaxResponseBody result = new AjaxResponseBody();

if (isValidSearchCriteria(search)) {

List<User> users = findByUserNameOrEmail(search.getUsername(), search.getEmail());

if (users.size() > 0) {

result.setCode("200");

result.setMsg("");

result.setResult(users);

} else {

result.setCode("204");

result.setMsg("No user!");

}

} else {

result.setCode("400");

result.setMsg("Search criteria is empty!");

}

//AjaxResponseBody will be converted into json format and send back to the request.

return result;

}

111.What is XMLHttpRequest Object?

**XMLHttpRequest** (**XHR**) is an API that can be **used** by JavaScript, JScript, VBScript, and other web browser scripting languages to transfer and manipulate XML data to and from a webserver using HTTP, establishing an independent connection channel between a webpage's Client-Side and Server-Side.

SQL

**SQL** (pronounced "ess-que-el") stands for Structured Query Language. ... **SQL** statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use **SQL** are: Oracle, Sybase, Microsoft **SQL** Server, Access, Ingres, etc.

112.What is Index?

**SQL** - Indexes. Advertisements. Indexes are special lookup tables that the database search engine can use to speed up data retrieval. Simply put, an **index** is a pointer to data in a table. An **index** in a database is very similar to an **index** in the back of a book.

113.Write the Join Query.  
114.Write the query for calculating the Second Max Highest Salary?

Micro Services  
115.What is MicroServices?Architecture of Micro Service?

What are **microservices**? ... In short a **microservices** architecture is a term used to describe the practice of breaking up an application into a series of smaller, more specialised parts, each of which communicate with one another across common interfaces such as APIs and REST interfaces like HTTP

This post presents the basic architecture of an online flower shop in a microservices architecture. I have no intention to provide detailed information on each component on this post.

I do provide a working example of a microservices application based on [Spring Cloud](https://spring.io/projects/spring-cloud) project. I also provide a basic explanation of the components' responsibilities and how they achieve that, which I believe is a good way to understand the microservices architecture.

## **Components**

### **Store Front**

storefront folder on the repository.

* The Store itself, like a website (although here it's just an API) where a user buys flowers and stuff.
* Balances the requests to suppliers and carriers locally (client-side load-balancing) by fetching and caching Eureka server information (ribbon).
* Uses Hystrix to control timeout on methods that use other microservices.
* Uses Bulkhead (from Hystrix) to separate a group of threads for each operation managed by Hystrix.
* Controls the integration with other microservices (using states).
  + RECEIVED: order created on the storefront microservice.
  + SUPPLIER\_ORDER\_CREATED: order incremented with information from the supplier microservice.
  + CARRIER\_VOUCHER\_CREATED: order incremented with information from the carrier microservice.
* Integrates with the authentication server.
* Forwards authentication headers to other microservices with a RequestInterceptor bean.

### **Supplier**

supplier folder on the repository.

* Someone who provides products on different locations.
* Provides Orders to the storefront component.
* Integrates with the authentication server.

### **Carrier**

carrier folder on the repository.

* Someone who delivers stuff to clients from the suppliers.
* Provides Vouchers to the storefront component.
* Integrates with the authentication server.

### **Authentication Server**

auth-service folder on the repository.

* This is an OAuth server tying together Spring Security and Spring Cloud OAuth2.
* Spring Security configures a user (hardcoded =).
* Spring Cloud OAuth2 configures a client of the authorization server, the storefront (hardcoded =).

### **API Gateway**

zuul-api-gateway folder on the repository.

* Clients make requests for the API Gateway, which then redirects to the correct microservice for the request.
* Implemented with Netflix Zuul.
* Integrates (automatically) with Eureka to get microservices instances.
* Does client-side load balancing with Ribbon automatically.

### **Eureka server**

eureka-server folder.

* Handles service registry and discovery.
* Every component registers itself here.
* Balancing is on the client,  (this was replaced by the FeignClient. See older commits to understand how to configure a RestTemplate to work with Eureka Client).
* Balancing is on the client through FeignClient, which is auto-configured on each application to balance requests when eureka client is being used.

### **Config Server**

config-server folder.

* Provides configuration to the microservices. The config-repo folder is used to store the configuration files.

### **Other Stuff**

* I've configured to log to papertrail (a log aggregator as a service) and used Spring Cloud Sleuth to add a traceId to every user request, being able to trace the request across microservices.
* The logback.xml on each "domain" microservice does the magic. This will stop working after a while, so provide your own configuration for logback...

## **Hands-on!**

1. Clone/fork [this GitHub repository](https://github.com/brunodrugowick/microservices-flowershop).
2. Import everything on [STS](https://spring.io/tools).
3. Provide the following configurations:
   * configure the config-server application.yml file with at least the search-locations property. You may change to the location of your own config repo on your local machine or use the commented configuration to use a github repository.
   * provide a MariaDB instance running with the following schemas already created: supplier; storefront;
   * Note: you can configure the database, connection info and schema name on the configuration files on the config-server.
4. Run each of them separately using ./mvnw spring-boot:run on each folder OR the Boot Dashboard on STS.

## **Test Stuff**

There's a [cartRequests.jmx](https://github.com/brunodrugowick/microservices-flowershop/blob/master/README/cartRequests.jmx) to be imported on [JMeter](https://jmeter.apache.org/) and test Hystrix. Ask me if you want to know more about it.

There's also a [insomnia\_requests.json](https://github.com/brunodrugowick/microservices-flowershop/blob/master/README/insomnia_requests.json) file to be imported on [Insomnia](https://insomnia.rest/download/) and test all endpoints.

## **Notes**

* commits to master tell the story of this project. You can follow [here](https://github.com/brunodrugowick/microservices-flowershop/commits/master).
* this implementation is based on two courses from [Alura](https://www.alura.com.br/): [this (in portuguese)](https://www.alura.com.br/curso-online-microservices-spring-cloud-service-registry-config-server) and [this (in portuguese)](https://www.alura.com.br/curso-online-microservices-spring-cloud-circuit-breaker-api-gateway).
* This is a [work in progress](https://github.com/brunodrugowick/microservices-flowershop/issues).
* The cover image for this post is taken from the spring.io homepage ([here](https://spring.io/img/homepage/diagram-distributed-systems.svg)).

116.Difference between Monolithic and Microservices?

***So what's actually this monolithic, what's a microservice. ???***

*Monolithic :*

* *The entire front end is one application*
* *The entire back end is another humongous application.*

*The advantages with monolithic approach are*

* *You avoid a lot of network calls between components. It's more like including a bunch of SDK's .*
* *You avoid the complexity of setting up a microservice based cluster.*

*The disadvantages with monolithic*

* *As mentioned it becomes humongous as the application and the features grow.*
* *Tonnes of open pull requests on github.*
* *Time spent on Merge conflicts will be more than the actual development.*
* *Database structure will be so huge that****when I saw our old monolithic app database , it consists close to 1000 SQL tables.***
* *Let's say one of the developer didn't handled an exception gracefully and at certain condition, the app intermittently shut down.****It takes couple of minutes for such humongous apps to be live again and most of large webscale applications can't afford that down time.***
* *Let's say if there is a small bug in the code and the development team want to take it live, the entire application containers need to be restarted which is a ton of effort and cost .*
* *It becomes next to impossible to understand the codebase given a new joinee joined a team.*
* ***Build time for the project will exponentially increase and this will be really a hectic thing for a developer.****Lets say if I as a developer want to make a small change and test a feature, I need to wait for 5 minutes for the project to build and if the change doesn't work, I need to make a build again and wait for 5 more minutes.****This is wait more code less model.***
* *This model of monolithic completely disagrees the philosophy of docker.****Docker says the container had to be as light as possible so that the restart or spinning up new container will be in flash of an eye.***

**Monolithic -**All in one, does everything, for instance a single API or service that literally handles everything.

* simple to develop with, everything is self - contained
* easier to comprehend for developers.
* simple to deploy, test, possibly scale
* deployed as a single unit
* Traditional development

**Microservices** - Smaller (as Micro implies) focused services for specific functionality.. a combination of micro-services combine to form a whole.

* loosely coupled
* smaller independent units of functionality allows ability to change specific features easier.
* can be written in different languages and platforms
* can be deployed independently of each other
* deployed as different units
* must co-operate with each other with authentication, api interfaces etc.

Traditional legacy applications tend to be monolithic with more modern approaches leaning towards sets of microservices.

117.What is Docker and Kubernate?  
118.How to make image on Docker?  
Design Pattern  
119.What is Singleton?What the code of Sinletion with all the iterartions including Enum?  
120.Decorator,Fly wieight,Template,Factory,Abstract Factory,Builder,Interceptor Filter are asking?  
121.Bean scope in spring?  
122.Difference between final,finally,finalize.  
123.static and instance control flow.

1. dentification of static members from top to bottom. All the static variables, methods, and blocks are identified during this step.
2. Execution of static variable assignments and static blocks from top to bottom.
3. Finally, the static main method is executed in the last step of static control flow.

**Instance control flow**

Now, instance control flow is mostly similar to static control flow,

The first operation it does in a program is, it will perform the static control flow sequence

The third point was, execution of main method, and inmain method, there is a new object creation.

Now, when control goes to the object creation statement (Test t=new Test();), instance control flow will be initiated according to the following rules

* Identifying instance members from top to bottom
* Execution of instance variables assignment and instance blocks
* Execution of constructor

Programs

1.find the output of below code:  
package com.nhc.core.enumm;

public class BalanceAccount {  
private enum Payments {  
POSTPOND, OLD, NEW  
}  
String payments;  
public BalanceAccount() {  
String balance = "12345";  
payments = Payments.POSTPOND.toString();  
}

public String toString() {  
return "Payments =" + payments;  
}

public static void main(String[] args) {  
System.out.println(new BalanceAccount());  
System.out.println(Payments.NEW);

}  
}

Output- >Payments =POSTPOND NEW

2.Display the output like this in HashMap:  
Input-(“A”,”1”),(“B”,”3”),(“C”,”5”),(“D”,”0”),(“E”,”7”)  
Output-(“D”,”0”),(“A”,”1”),(“B”,”3”),(“C”,”5”),(“E”,”7”)

3.Write a code in MVC to find/store Patient address of the given Id?

4.write a SQL query to display eid,ename,salary from emp\_salary and find a minimum salary for nth employee.  
5.Where we should prefer abstract class over interface?

If **you** want **to** provide common, implemented functionality among all implementations of your component, **use an abstract class**. **Abstract classes** allow **you to** partially implement your **class**, whereas **interfaces** contain no implementation for any members

6.Develop a MVC program like BookMyShow?

7.Wap to find the occurrence of character in String  
Ex:- String str=abababaal  
8.Wap to find the occurrence of word in String  
Ex:- String str=”hello how are you,hello I am f9”  
9.Permutations and Combinations of String?  
10.Recursion Program of Fibonacci and Factorial?  
11.find the duplicate elements in Array?  
12.Remove Duplicate elements in Array?  
13.Based on the sum and size find the elements in array?  
14.Reverse String program without using reverse() method?  
15.Swap the number without using third variable.  
16.Swap the String without using third variable.  
17.Find the most occurrence character in String and write the test cases also?  
18.how to check String is Anagram?  
19.print the even and odd nos by creating two threads sequentially.

1. What is an exception?  
   2) How the exceptions are handled in java? OR Explain exception handling mechanism in java?  
   3) What is the difference between error and exception in java?
2. **Difference Between Error and Exception in Java**. “Throwable” act as the root for **Java's error and exception** hierarchy. “**Error**” is a critical condition that cannot be handled by the code of the program. “**Exception**” is the exceptional situation that can be handled by the code of the program

4) Can we keep other statements in between try, catch and finally blocks?

No, **we** cannot write any **statements in between try**, **catch and finally blocks** and these **blocks** form one unit. ... If **we try** to put any **statements between** these **blocks**, it **will** throw a compile-time error.

5) Can we write only try block without catch and finally blocks?

**No**, It shows compilation error. The try block must be followed by either catch or finally block. You can remove either catch block or finally block but not both. Following are the valid combinations.  
6) There are three statements in a try block – statement1, statement2 and statement3. After that there is a catch block to catch the exceptions occurred in the try block. Assume that exception has occurred in statement2. Does statement3 get executed or not?  
7) What is unreachable catch block error?

## Featured snippet from the web

When we are keeping multiple **catch blocks**, the order of **catch blocks** must be from most specific to most general ones. i.e subclasses of Exception must come first and superclasses later. If we keep superclasses first and subclasses later, the compiler will throw an **unreachable catch block error**.

try {

   // statements

} catch(Exception e) {

   System.out.println(e);

} catch(NumberFormatException nfe) { **//unreachable block. Not supported by Java, leads to an error.**

   System.out.println(nfe);

}

8) Explain the hierarchy of exceptions in java?  
9) What are run time exceptions in java. Give example?  
10) What is OutOfMemoryError in java?

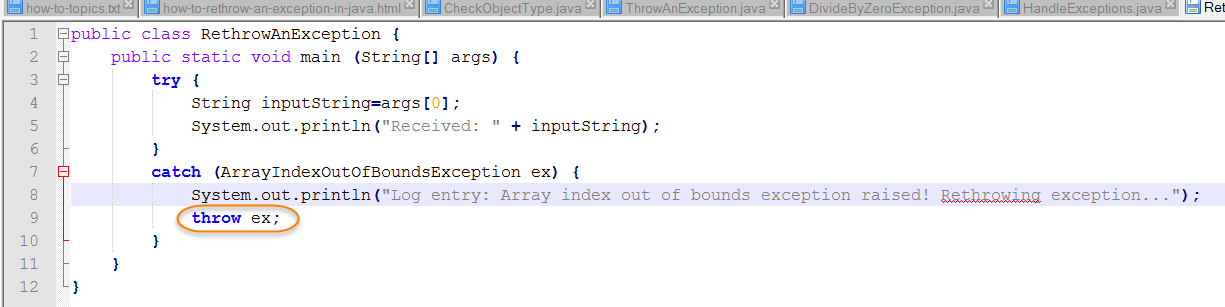
The **OutOfMemoryError** Exception in **Java** looks like this: Exception in thread "main" **java**.lang.**OutOfMemoryError**: **Java** heap space. Usually, this error is thrown when the **Java** Virtual Machine cannot allocate an object because it is out of memory, and no more memory could be made available by the garbage collector.  
11) what are checked and unchecked exceptions in java?  
12) What is the difference between ClassNotFoundException and NoClassDefFoundError in java?

**ClassNotFoundException** is an exception that occurs when you try to load a class at run time using Class. forName() or loadClass() methods and mentioned classes are not found **in the** classpath. **NoClassDefFoundError** is an error that occurs when a particular class is present at compile time, but was missing at run time.

13) Can we keep the statements after finally block If the control is returning from the finally block itself?

**If finally block** does not **return** a value then both try and catch **blocks** must **return** a value. **If** try-catch-**finally blocks** are **returning** a value according to above rules, then **you** should not **keep** any **statements after finally block**. Because they become unreachable and in Java, Unreachable code gives compile time error.  
14) Does finally block get executed If either try or catch blocks are returning the control?  
15) Can we throw an exception manually? If yes, how?  
16) What is Re-throwing an exception in java?

An exception can be rethrown in a catch block. This action will cause the exception to be passed to the calling method. If the rethrow operation occurs in the main method then the exception is passed to the JVM and displayed on the console. The purpose of the rethrow operation is to get the attention of the outside world that an exception has occurred and at the same time perform any contingency logic (such as logging) in the catch block. To learn how to rethrow an exception in Java, follow these four steps.



17) What is the use of throws keyword in java?  
18) Why it is always recommended that clean up operations like closing the DB resources to keep inside a finally block?

Because **finally block** is **always** executed whether exceptions are raised in the try **block** or not and raised exceptions are caught in the catch **block** or not.

19) What is the difference between final, finally and finalize in java?  
20) How do you create customized exceptions in java?  
21) What is ClassCastException in java?

**ClassCastException** is a runtime exception raised in **Java** when we try to improperly cast a class from one type to another. It's thrown to indicate that the code has attempted to cast an object to a related class, but of which it is not an instance.

22) What is the difference between throw, throws and throwable in java?  
23) What is StackOverflowError in java?

A **StackOverflowError** is a runtime error in **java**. It is thrown when the amount of call stack memory allocated by JVM is exceeded. A common case of a **StackOverflowError** being thrown, is when call stack exceeds due to excessive deep or infinite recursion.

24) Can we override a super class method which is throwing an unchecked exception with checked exception in the sub class?

There are few things to remember when **overriding** a **method** with **exception** handling. If **super class method does** not declare any **exception**, then **sub class** overriden **method** cannot declare **checked exception** but it **can** declare **unchecked exceptions**.  
25) What are chained exceptions in java?

**Chained Exception** helps to identify a situation in which one **exception** causes another **Exception** in an application. For instance, consider a method which throws an ArithmeticException because of an attempt to divide by zero but the actual cause of **exception** was an I/O error which caused the divisor to be zero.  
26) Which class is the super class for all types of errors and exceptions in java?  
27) What are the legal combinations of try, catch and finally blocks?  
28) What is the use of printStackTrace() method?

# **Search Results**

## Featured snippet from the web

The **printStackTrace() method** in Java is a tool **used** to handle exceptions and errors. It is a **method** of Java's throwable class which prints the throwable along with other details like the line number and class name where the exception occurred. **printStackTrace()** is very useful in diagnosing exceptions.

29) Give some examples to checked exceptions?  
30) Give some examples to unchecked exceptions?

Q1) What are the advantages of using Spring boot over spring framework?  
Answer:  
Some advantage points are listed below:  
Spring boot provides a configuration which is opinionated, thus avoiding lots of boilerplate code and configuration.  
Its applications can be easily integrated into the Spring ecosystem like Spring Security, Spring AOP, Spring transaction, and cache etc.  
It provides embedded HTTP servers like Tomcat etc. to enhance the development process.  
Spring boot offers a command-line interface i.e. CLI tool to develop and test the application, which is prompt and less intrusive.  
Spring boot reduces a lot of development time, by reducing the configuration to minimal or “no-configuration” approach, thus enhances productivity.  
Q2) What is @SpringBootApplication annotation?  
Answer:  
Before Spring BOOT 1.2, it was very common to use annotations like @Configuration, @EnableAutoConfiguration, @ComponentScan. The @SpringBootApplication annotation is equal to all of the three annotations mentioned before with their default attributes. Means a single annotation is enough now for multiple features like enabling auto configuration and performing a component scan of beans.  
@SpringBootApplication  
public class MyApp {  
……….  
}  
Q3) Explain about Spring boot starter POM file?

Answer:  
Starter POM file actually contains a lot of dependencies, so that project can be up and running quickly within in a very short span of time. It is basically a combination of dependency descriptors that anyone can include in their application and automatically all project related dependency would be available. Starter POM files also manage transitive dependencies of the project. POM file structure is arriving from Maven based application. In other words, a developer creating a project that uses Spring REST for creating rest APIs, just have to include relevant starter POM file, which will import all required dependencies for Spring rest application. All the tedious task of searching and configuring dependencies, required for a framework is now no more needed.  
Q4) Explain the Actuator in Spring Boot?

Answer:  
Actuator brings production-ready features to the table for Spring boot application. Production-ready features like application monitoring, gathering metrics, understanding traffic, and the database state, become very crucial to keep the application in a healthy state. A major benefit of utilizing Actuator like a library is that a developer can have access to production-grade tools, without having to implement any one of these tools. To enable Spring Boot Actuator dependency to our package manager, add below to your POM file

<dependency>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-actuator</artifactId>  
</dependency>

Once this dependency is in the classpath, multiple endpoints are available with the developer.

Q5) What is the way to reload changes on Spring boot without server-restart?

Answer:  
Any changes are reloaded in spring boot without starting the server by using dev tools.

<dependency>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-devtools</artifactId>  
<optional>true</optional>  
</dependency>

Spring boot provides module called DevTools, which can enhance the productivity of a spring boot developer. It can auto-deploy the changes to the server, with auto-restarting the server. This is the common Spring Boot Interview Questions asked in an interview. Thus, a developer can reload his changes on Spring boot without having to restart the server. This package is provided in development mode only, but not in production mode.

Q6) What is the way to run the Spring boot application on a custom port?

Answer:  
There is a file called application.properties in Spring boot. This file can be customized to bring in any change, to alter the behavior of a running spring boot application. If a developer wants to run a spring boot application on a custom port, he can specify the port in an application.properties file:

server.port = 8080

This entry will ensure that application would run on 8080 port.

Q7) What is the way to implement Spring batch in Spring boot?

Answer:  
batch processing involves large volumes of data records processing. Spring boot batch provides a function which can be reused, and essential for doing batch processing. It also provides services and features, which helps in optimization and partition techniques, resulting in high volume and high-performance batch jobs.

<dependency>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-batch</artifactId>  
<optional>true</optional>  
</dependency>

Above changes in POM file, will include the necessary packages in an application, which are required to do the batch processing in spring boot project.

Let us move to the next Spring Boot Interview Questions

Q8) What is the way to configure logging in Spring boot application?

Answer:  
The developer can easily specify logging level in an application.properties files:  
Logging.level.spring.framework = Debug  
This single line in application properties file will let the spring framework logs to the debug level.  
In case, a developer wants to put logs to the file, he can specify logger.file in application properties

Logging.file = ${java.io.tempdirectory}/sample.log

Apart from the above two approaches, a developer can also create a logback.xml file under main/java/resources and specify the logging configuration in the file. Spring boot will automatically pick this file.

Q9) What is a benefit of including spring-boot-maven-plugin?

Answer:  
This is the advanced Spring Boot Interview Questions asked in an interview. spring-boot-maven-plugin provides a list of commands which are helpful in enabling the developer to package the code as a jar file to run the application.  
Spring-boot: run, it will run spring boot application.  
Spring-boot: repackage, it will repackage jar or war file  
Spring-boot: build, generate build information  
Spring-boot: start, stop – to manage the lifecycle of spring boot application.

Q10) What is the way to add custom JS code in Spring boot application?

Answer:  
A developer can create a folder by the name of “static”, under the resources folder. Then all the static content can be put into this folder.  
Any JavaScript file i.e. test.js would reside in /resources/static/js/test.js  
Developer can then refer to this file in JSP like:  
<script src = “/js/test.js”></script>

Interview Experience with @Capgemini thought it would be help to someone who is looking out the same.  
Below are the list of interview questions asked as part of capgemini.  
1.Explain project architect.  
2. What are solid principals  
3. What are Oops concept.  
4. Explain comparable and comparator.  
5. Why return type doesn't take part in overriding.

Yes. It **is** possible for **overridden** methods to have different **return type** . But the limitations are that the **overridden** method must have a **return type** that **is** more specific **type** of the **return type** of the actual method  
6.spring interview questions.  
7. spring boot interview questions.  
8. Apache camel interview questions

#Hexaware Telephonic Round interview questions  
-------------------------------------------------------------------------  
1.What is difference between Object oriented and object based langugae.

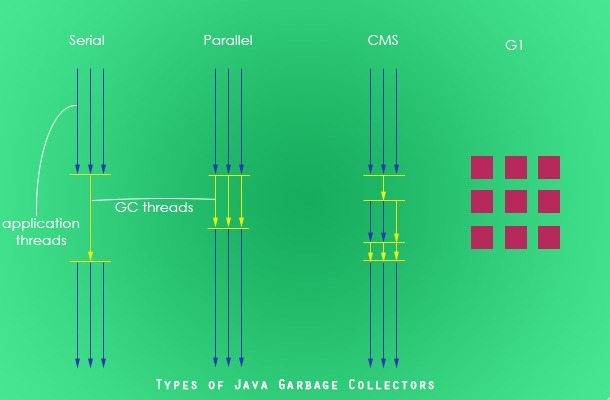
**Difference between Object**-**oriented and Object**-**based languages**. **Object**-**oriented language** supports all the features of OOPs (Abstraction, Encapsulation, Inheritance, Polymorphism). **Object**-**based languages** don't support all the features of OOPs like polymorphism or inheritance.

2.Difference between List and set.  
3.What is serialization?.  
4.Difference between serial garbage collector and parallel garbage collector.

Java has **four types of garbage collectors**,

1. [Serial Garbage Collector](https://javapapers.com/java/types-of-java-garbage-collectors/#serial-garbage-collector)
2. [Parallel Garbage Collector](https://javapapers.com/java/types-of-java-garbage-collectors/#parallel-garbage-collector)
3. [CMS Garbage Collector](https://javapapers.com/java/types-of-java-garbage-collectors/#cms-garbage-collector)
4. [G1 Garbage Collector](https://javapapers.com/java/types-of-java-garbage-collectors/#g1-garbage-collector)

Each of these four types has its own advantages and disadvantages. Most importantly, we the programmers can choose the type of garbage collector to be used by the JVM. We can choose them by passing the choice as JVM argument. Each of these types differ largely and can provide completely different application performance. It is critical to understand each of these types of garbage collectors and use it rightly based on the application.



## 1. Serial Garbage Collector

Serial garbage collector works by holding all the application threads. It is designed for the single-threaded environments. It uses just a single thread for garbage collection. The way it works by freezing all the application threads while doing garbage collection may not be suitable for a server environment. It is best suited for simple command-line programs.

Turn on the -XX:+UseSerialGC JVM argument to use the serial garbage collector.

## 2. Parallel Garbage Collector

Parallel garbage collector is also called as throughput collector. It is the default garbage collector of the JVM. Unlike serial garbage collector, this uses multiple threads for garbage collection. Similar to serial garbage collector this also freezes all the application threads while performing garbage collection.

## 3. CMS Garbage Collector

Concurrent Mark Sweep (CMS) garbage collector uses multiple threads to scan the heap memory to mark instances for eviction and then sweep the marked instances. CMS garbage collector holds all the application threads in the following two scenarios only,

1. while marking the referenced objects in the tenured generation space.
2. if there is a change in heap memory in parallel while doing the garbage collection.

In comparison with parallel garbage collector, CMS collector uses more CPU to ensure better application throughput. If we can allocate more CPU for better performance then CMS garbage collector is the preferred choice over the parallel collector.

Turn on the XX:+USeParNewGC JVM argument to use the CMS garbage collector.

## 4. G1 Garbage Collector

G1 garbage collector is used for large heap memory areas. It separates the heap memory into regions and does collection within them in parallel. G1 also does compacts the free heap space on the go just after reclaiming the memory. But CMS garbage collector compacts the memory on stop the world (STW) situations. G1 collector prioritizes the region based on most garbage first.

Turn on the –XX:+UseG1GC JVM argument to use the G1 garbage collector.

### Java 8 Improvement

Turn on the -XX:+UseStringDeduplication JVM argument while using G1 garbage collector. This optimizes the heap memory by removing duplicate String values to a single char[] array. This option is introduced in [Java 8](https://javapapers.com/java/java-8-features/) u 20.

Given all the above four types of Java garbage collectors, which one to use depends on the application scenario, hardware available and the throughput requirements.

## Garbage Collection JVM Options

Following are the key JVM options that are related to Java garbage collection.

### Type of Garbage Collector to run

|  |  |
| --- | --- |
| **Option** | **Description** |
| -XX:+UseSerialGC | Serial Garbage Collector |
| -XX:+UseParallelGC | Parallel Garbage Collector |
| -XX:+UseConcMarkSweepGC | CMS Garbage Collector |
| -XX:ParallelCMSThreads= | CMS Collector – number of threads to use |
| -XX:+UseG1GC | G1 Gargbage Collector |

5.How we can write our own immutable class.  
6.What all are synchroized objects in collections.

7

Thread safe Collections -

1. **ConcurrentHashMap**

Thread safe without having to synchronize the whole map Very fast reads while write is done with a lock No locking at the object level Uses multitude of locks.

1. **SynchronizedHashMap**

Object level synchronization Both read and writes acquire a lock Locking the collection has a performance drawback May cause contention

1. **Vector**
2. **HashTable**
3. **CopyOnWriteArrayList**
4. **CopyOnWriteArraySet**
5. **Stack**

7.Other than string what all are immutable classes.

**LocalDate**, **LocalTime** and **LocalDateTime** classes (since 1.8) are also immutable. In fact, this very subject is on the OCAJSE8 (1Z0-808) exam, and that's precisely why I decided to treat it as not a mere comment.  
  
All primitive wrapper classes (such as **Boolean**, **Character**, **Byte**, **Short**, **Integer**, **Long**, **Float**, and **Double**) are immutable.  
  
**Money** and **Currency** API (slated for Java9) should be immutable, too.  
  
Incidentally, the **array-backed Lists** (created by Arrays.asList(myArray)) are structurally-immutable.  
  
Also, there are some border-line cases such as **java.util.Optional** (featured on the OCP exam, 1Z0-809), which is immutable if the contained element is itself immutable.

8.What is autoboxing and autounboxing.

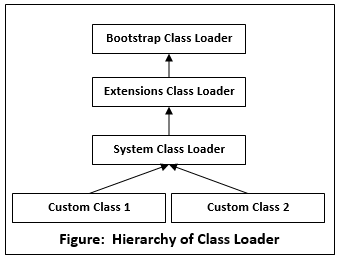
The automatic conversion of primitive data types into its equivalent Wrapper type is known as boxing and opposite operation is known as unboxing. This is the new feature of Java5. So java programmer doesn't need to write the conversion code.

1. **int** a=50;
2. Integer a2=**new** Integer(a);//Boxing
4. Integer a3=5;//Boxing
5. Integer i=**new** Integer(50);
7. **if**(i<100){            //unboxing internally
8. System.out.println(i);
9. }

9.What is class loader and how .class will be loaded.

## **Java ClassLoader**

Java ClassLoader is an abstract class. It belongs to a **java.lang** package. It loads classes from different resources. Java ClassLoader is used to load the classes at run time. In other words, JVM performs the linking process at runtime. Classes are loaded into the JVM according to need. If a loaded class depends on another class, that class is loaded as well. When we request to load a class, it delegates the class to its parent. In this way, uniqueness is maintained in the runtime environment. It is essential to execute a Java program.



Java ClassLoader is based on three principles: **Delegation**, **Visibility**, and **Uniqueness**.

* **Delegation principle:** It forwards the request for class loading to parent class loader. It only loads the class if the parent does not find or load the class.
* **Visibility principle:** It allows child class loader to see all the classes loaded by parent ClassLoader. But the parent class loader cannot see classes loaded by the child class loader.
* **Uniqueness principle:** It allows to load a class once. It is achieved by delegation principle. It ensures that child ClassLoader doesn't reload the class, which is already loaded by the parent.

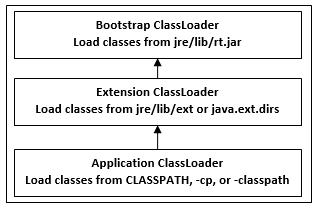
## **Types of ClassLoader**

In Java, every ClassLoader has a predefined location from where they load class files. There are following types of ClassLoader in Java:

**Bootstrap Class Loader:** It loads standard JDK class files from rt.jar and other core classes. It is a parent of all class loaders. It doesn't have any parent. When we call String.class.getClassLoader() it returns null, and any code based on it throws NullPointerException. It is also called Primordial ClassLoader. It loads class files from jre/lib/rt.jar. For example, java.lang package class.

**Extensions Class Loader:** It delegates class loading request to its parent. If the loading of a class is unsuccessful, it loads classes from jre/lib/ext directory or any other directory as java.ext.dirs. It is implemented by sun.misc.Launcher$ExtClassLoader in JVM.

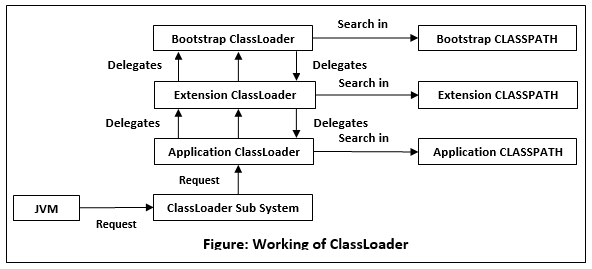
**System Class Loader:** It loads application specific classes from the CLASSPATH environment variable. It can be set while invoking program using -cp or classpath command line options. It is a child of Extension ClassLoader. It is implemented by sun.misc.Launcher$AppClassLoader class. All Java ClassLoader implements java.lang.ClassLoader.



## **How ClassLoader works in Java**

When JVM request for a class, it invokes a loadClass() method of the java.lang.ClassLoader class by passing the fully classified name of the class. The loadClass() method calls for findLoadedClass() method to check that the class has been already loaded or not. It is required to avoid loading the class multiple times.

If the class is already loaded, it delegates the request to parent ClassLoader to load the class. If the ClassLoader is not finding the class, it invokes the findClass() method to look for the classes in the file system. The following diagram shows how ClassLoader loads class in Java using delegation.



Suppose that we have an application specific class Demo.class. The request for loading of this class files transfers to Application ClassLoader. It delegates to its parent Extension ClassLoader. Further, it delegates to Bootstrap ClassLoader. Bootstrap search that class in rt.jar and since that class is not there. Now request transfer to Extension ClassLoader which searches for the directory jre/lib/ext and tries to locate this class there. If the class is found there, Extension ClassLoader loads that class. Application ClassLoader never loads that class. When the extension ClassLoader does not load it, then Application ClaasLoader loads it from CLASSPATH in Java.

Visibility principle states that child ClassLoader can see the class loaded by the parent ClassLoader, but vice versa is not true. It means if Application ClassLoader loads Demo.class, in such case, trying to load Demo.class explicitly using Extension ClassLoader throws java.lang.ClassNotFoundException.

According to the uniqueness principle, a class loaded by the parent should not be loaded by Child ClassLoader again. So, it is possible to write class loader which violates delegation and uniqueness principles and loads class by itself.

In short, class loader follows the following rule:

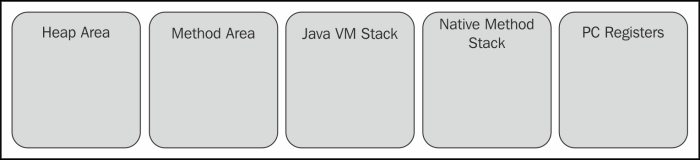
* It checks if the class is already loaded.
* If the class is not loaded, ask parent class loader to load the class.
* If parent class loader cannot load class, attempt to load it in this class loader.

10.What are the various memory areas in JVM.

## JVM Memory Model / Structure

The Java Virtual Machine defines various **run-time data areas** that are used during execution of a program. Some of these data areas are created on Java Virtual Machine start-up and are destroyed only when the Java Virtual Machine exits. Other data areas are per thread. Per-thread data areas are created when a thread is created and destroyed when the thread exits.

Let’s look at the most basic categorization of various parts inside runtime memory.

[](https://howtodoinjava.com/wp-content/uploads/2014/08/JVM-Memory-Area-Parts.jpg)JVM Memory Area Parts

Let’s have a quick look at each of these components according to what is mentioned in the JVM specifications.

## Heap area

The heap area represents the runtime data area, from which the memory is allocated for all class instances and arrays, and is created during the virtual machine startup.

The heap storage for objects is reclaimed by an automatic storage management system. The heap may be of a fixed or dynamic size (based on system’s configuration), and the memory allocated for the heap area does not need to be contiguous.

*A Java Virtual Machine implementation may provide the programmer or the user control over the initial size of the heap, as well as, if the heap can be dynamically expanded or contracted, control over the maximum and minimum heap size.*

If a computation requires more heap than can be made available by the automatic storage management system, the Java Virtual Machine throws an OutOfMemoryError.

## Method area and runtime constant pool

Method area stores per-class structures such as the runtime constant pool; field and method data; the code for methods and constructors, including the special methods used in class, instance, and interface initialization.

The method area is created on the virtual machine startup. Although it is logically a part of the heap but it can or cannot be garbage collected, whereas we already read that garbage collection in heap is not optional; it’s mandatory. The method area may be of a fixed size or may be expanded as required by the computation and may be contracted if a larger method area becomes unnecessary. The memory for the method area does not need to be contiguous.

If memory in the method area cannot be made available to satisfy an allocation request, the Java Virtual Machine throws an OutOfMemoryError.

## JVM Stacks

Each of the JVM threads has a private stack created at the same time as that of the thread. The stack stores frames. A frame is used to store data and partial results and to perform dynamic linking, return values for methods, and dispatch exceptions.

It holds local variables and partial results and plays a part in the method invocation and return. Because this stack is never manipulated directly, except to push and pop frames, the frames may be heap allocated. Similar to the heap, the memory for this stack does not need to be contiguous.

This specification permits that stacks can be either of a fixed or dynamic size. If it is of a fixed size, the size of each stack may be chosen independently when that stack is created.

If the computation in a thread requires a larger Java Virtual Machine stack than is permitted, the Java Virtual Machine throws a StackOverflowError.

If Java Virtual Machine stacks can be dynamically expanded, and expansion is attempted but insufficient memory can be made available to effect the expansion, or if insufficient memory can be made available to create the initial Java Virtual Machine stack for a new thread, the Java Virtual Machine throws an OutOfMemoryError.

## Native method stacks

Native method stacks is called C stacks; it support native methods (methods written in a language other than the Java programming language), typically allocated per each thread when each thread is created. Java Virtual Machine implementations that cannot load native methods and that do not themselves rely on conventional stacks need not supply native method stacks.

The size of native method stacks can be either fixed or dynamic.

If the computation in a thread requires a larger native method stack than is permitted, the Java Virtual Machine throws a StackOverflowError.

If native method stacks can be dynamically expanded and native method stack expansion is attempted but insufficient memory can be made available, or if insufficient memory can be made available to create the initial native method stack for a new thread, the Java Virtual Machine throws an OutOfMemoryError.

## PC registers

Each of the JVM threads has its own program counter (pc) register. At any point, each of the JVM threads is executing the code of a single method, namely the current method for that thread.

As the Java applications can contain some native code (for example, using native libraries), we have two different ways for native and non-native methods. If the method is not native (that is, a Java code), the PC register contains the address of the JVM instruction currently being executed. If the method is native, the value of the JVM’s PC register is undefined.

The Java Virtual Machine’s pc register is wide enough to hold a return address or a native pointer on the specific platform.

11.Difference between composition and aggregarion.  
12.Java 1.7 features.

###### FOLLOWING IS THE LIST JDK 1.7 FEATURES ADDED, OF DAILY IMPORTANCE.

**1.** String in Switch Expression  
**2.** Underscores Between Digits in Numeric Literals  
**3.** Integral Types as Binary Literals  
**4.** Handling multiple exceptions in a single catch block  
**5.** Try-with-resources Statement  
**6.** Automatic Type Inference in Generic object instantiation

13.What is package for these primitive data types and collections.  
14.What is marker interface in java and examples.  
15.Methods in object class.

|  |
| --- |
| The Object class provides many methods. They are as follows: |

|  |  |
| --- | --- |
| **Method** | **Description** |
| public final Class getClass() | returns the Class class object of this object. The Class  class can further be used to get the metadata of this  class. |
| public int hashCode() | returns the hashcode number for this object. |
| public boolean equals(Object obj) | compares the given object to this object. |
| protected Object clone() throws CloneNotSupportedException | creates and returns the exact copy (clone) of this object. |
| public String toString() | returns the string representation of this object. |
| public final void notify() | wakes up single thread, waiting on this object's monitor. |
| public final void notifyAll() | wakes up all the threads, waiting on this object's monitor. |
| public final void wait(long timeout)throws InterruptedException | causes the current thread to wait for the specified  milliseconds, until another thread notifies (invokes  notify() or notifyAll() method). |
| public final void wait(long timeout,int nanos)throws InterruptedException | causes the current thread to wait for the specified  milliseconds and nanoseconds, until another thread  notifies (invokes notify() or notifyAll() method). |
| public final void wait()throws InterruptedException | causes the current thread to wait, until another thread  notifies (invokes notify() or notifyAll() method). |
| protected void finalize()throws Throwable | is invoked by the garbage collector before object is being  garbage collected. |

16.What are the design patterns used in ur project.

[#Sify](https://www.facebook.com/hashtag/sify?source=feed_text&epa=HASHTAG) Technologies Telephonic Round Interview questions  
---------------------------------------------------------------------------------  
1.Difference between throw and throws  
2.How can you write your immutable class  
3.Difference between abstrat class and interface  
4.How are you implementing abstract class and interface in ur project  
5.Difference between exception and error.  
6.Spring MVC  
7.What is synchronized and how u are implementing the same in ur project  
8.What is map and what is concurrent hash map  
9.If 3 users are sending the request how many instances of servltes will be created?

Servlets are managed resources in the servlet container, whenever a request comes for the servlet for the first time, servlet is loaded and instantiated and used for request processing.

A Servlet is instantiated only once in the container, and this Servelt object is used for any further request processing, be it another 15-20 different requests, this servlet object is shared among the different requests.

per-request new thread is created to handle the request not the Servlet object.

10.What methods you will use to return a value in JSP.

1. **getParameter(String name)**– This method is used to get the value of a request’s parameter. For example at login page user enters user-id and password and once the credentials are verified the login page gets redirected to user information page, then using request.getParameter we can get the value of user-id and password which user has input at the login page.
2. String Uid= request.getParameter("user-id");

String Pass= request.getParameter("password");

1. **getParameterNames() –** It returns enumeration of all the parameter names associated to the request.

Enumeration e= request.getParameterNames();

1. **getParameterValues(String name) –**It returns the array of parameter values.

String[] allpasswords = request.getParameterValues("password");

1. **getAttribute(String name) –**Used to get the attribute value.  request.getAttribute(“admin”) would give you the value of attribute admin.
2. **getAttributeNames() –**It is generally used to get the attribute names associated to the current session. It returns the enumeration of attribute names present in session.

Enumerator e = request.getAttributeNames();

1. **setAttribute(String,Object) –**It assigns an object’s value to the attribute. For example I have an attribute **password**and a String object str which has a value **“admin”**then calling request.setAttribute(“password”, str) would assign a value **admin** to the attribute **password**.
2. **removeAttribute(String) –**By using this method a attribute can be removed and cannot be used further. For e.g. If you have a statement **request.removeAttribute(“userid”)** on a JSP page then the userid attribute would be completely removed and request.getAttribute(“userid”) would return **NULL** if used after the removeAttribute method.
3. **getCookies() –**It returns an array of cookie objects received from the client. This method is mainly used when dealing with cookies in JSP.
4. **getHeader(String name) –**This method is used to get the header information of the request.
5. **getHeaderNames() –**Returns enumerator of all header names. Below code snippet would display all the header names associated with the request.
6. Enumeration e = request.getHeaderNames();
7. while (enumeration.hasMoreElements()) {
8. String str = (String)e.nextElement();
9. out.println(str);

}

1. **getRequestURI() –**This method (request.getRequestURI()) returns the URL of current JSP page.
2. **getMethod() –** It returns HTTP request method. request.getMethod(). For example it will return GET for a Get request and POST for a Post Request.
3. **getQueryString() –**Used for getting the query string associated to the JSP page URL. It is the string associted to the URL after question mark sign (?).

11.2nd highest salary

SELECT name, MAX(salary) AS salary

FROM employee

WHERE salary < (SELECT MAX(salary)

FROM employee);

12.How to find duplicate values in a string by using collection.  
13.Difference between comparable and comparator.  
14.What is the error you will get when you are trying to create instance of interface  
A:Cannot instantiate the type List

---------------------------------------------------  
#Cognizant:  
-------------  
1.Difference between Soap and Rest.  
2.Difference between setter injection and constructor injection.  
3.What is design pattern why we need design patterns?.  
4.Caching mechanism in hibernate?.

<property name = "hibernate.cache.provider\_class">

org.hibernate.cache.EhCacheProvider

</property>

<diskStore path="java.io.tmpdir"/>

<defaultCache

maxElementsInMemory = "1000"

eternal = "false"

timeToIdleSeconds = "120"

timeToLiveSeconds = "120"

overflowToDisk = "true"

/>

<cache name = "Employee"

maxElementsInMemory = "500"

eternal = "true"

timeToIdleSeconds = "0"

timeToLiveSeconds = "0"

overflowToDisk = "false"

/>

Session session = SessionFactory.openSession();

Query query = session.createQuery("FROM EMPLOYEE");

query.setCacheable(true);

List users = query.list();

SessionFactory.closeSession();

Session session = SessionFactory.openSession();

Query query = session.createQuery("FROM EMPLOYEE");

query.setCacheable(true);

query.setCacheRegion("employee");

List users = query.list();

SessionFactory.closeSession();

5.When does the cache expires in hibernate?.

The following attributes and elements are optional.

timeToIdleSeconds:

Sets the time to idle for an element before it expires.

i.e. The maximum amount of time between accesses before an element expires

Is only used if the element is not eternal.

Optional attribute. A value of 0 means that an Element can idle for infinity.

The default value is 0.

timeToLiveSeconds:

Sets the time to live for an element before it expires.

i.e. The maximum time between creation time and when an element expires.

Is only used if the element is not eternal.

Optional attribute. A value of 0 means that and Element can live for infinity.

The default value is 0.

6.ORM vs JDBC  
7.What is dialect in hibernate?.

**Hibernate** uses "**dialect**" configuration to know which database you are using so that it can switch to the database specific SQL generator code wherever/whenever necessary.

8.Immutable classes in java.  
9.Why we need immutable classes?.  
10.Hash code and equals methods.  
11.Concurrent Hash Map.

--------------------------------------------------  
#Cisco:  
-----  
1.Bean Scope in spring.  
2.Single ton and global session.  
3.Wait vs Sleep  
4.Immutable class  
5.Prototype and request.  
6.Lazy Loading in hibernate.  
7.Cascade attribute in hibernate and what are the values for cascade.

21

Cascading is about persistence actions involving one object propagating to other objects via an association. Cascading can apply to a variety of Hibernate actions, and it is typically transitive. The "cascade=..." attribute of the annotation that defines the association says what actions should cascade for that association.

Cascade = "all" means to apply all primary cascade types. As of Hibernate 5.3, these types are:

* "delete" / "remove",
* "detach" / "evict",
* "merge",
* "lock",
* "persist",
* "refresh",
* "replicate",
* "save\_update" / "update"

(Some of those cascade type names are old and/or deprecated.)

There are three more compound types:

* "all\_delete\_orphan" - means the same as "all" plus enabling the deletion of entities that are orphaned by the cascading.
* "delete\_orphan" - means "delete" plus orphan deletion.
* "none" - means no cascading.

8.

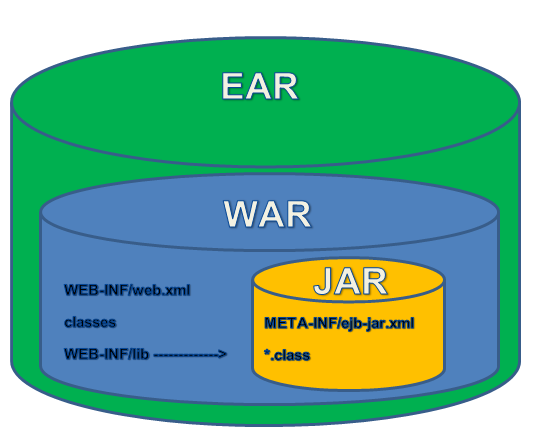
----------------------------------------------------  
#NIIT  
-----

1.About project  
2.Web services  
3.JDK vs JRE.  
4.Jar vs WAR

33

**WAR** stands for **Web application ARchive**

**JAR** stands for **Java ARchive**

[](https://i.stack.imgur.com/ZLdF7.png)

5.Checked vs unchecked  
6.Synchronizaction in thread.  
7.Runnable or thread.  
8.Auto Boxing and unboxing.  
9.java 5 features  
Java 6 features  
Java 7 features.  
10.Design patterns  
Ex:Single ton,Front controller,MVC.  
11.How to create single ton.  
12.What is WSDL file?.  
13.Soap Vs Rest?  
14.Servlet context vs servlet config.  
15.9 implict objects.  
16.Actions in struts.  
[#Programs](https://www.facebook.com/hashtag/programs?source=feed_text&epa=HASHTAG)  
1-Write a program to revrse a string without using StringAPI.  
2-Change Order of vowels in the string;(eg-"you are cool boy"="yuo era cool boy")  
3-Find Missing alfabates in the string.(eg-qwer tyu op asdf gh jl xcv nm ,missing are =ikzb)

technical interview qsn-  
1-Different Between Hashmap And Concurrent Hashmap?  
2-whats are the enhancements in multithreading in 1.7v of java?  
3-what is lambda expression in java?  
4-What is Deadlock?  
5-What is work of fork/join class in java?  
6-What is "WITH" clause in SQL?

The **SQL WITH clause** allows you to give a sub-query block a name (a process also called sub-query refactoring), which can be referenced in several places within the main **SQL** query. ... The **SQL WITH clause** was introduced by Oracle in the Oracle 9i release 2 database.  
7-Spring IOC container?  
8-Bean scopes?  
9-what are the interfaces spring impliments?  
10-difference between @path & URL?

path is location of a resource (file/directory) in a file system. Just like iOS File System, other environments file system can be Windows file system, Unix etc. Path can have spaces like /docs/random doc/. (between random and doc)

URL is is a reference to a resource anywhere (file system, web HTTP, FTP etc). URL can not have spaces like path.

Web URL: http://stackoverflow.com/  
file URL: file://localhost/Users/username/docs/random%20docs/  
path for above mentioned file URL: /Users/username/docs/random%20docs/

in layman terms:

URL = protocol (http, file etc) + host (domain name or IP or localhost) + path

11-Design Pattern catagories ?  
12-what is Factory & Singleton Design Pattern& in which catagory they fallunder?  
13-How to create Factory object in hibernate?

**SessionFactory** is an interface. **SessionFactory** can be created by providing **Configuration** object, which will contain all DB related property details pulled from either **hibernate**. cfg.

p.s- i dont remember others questions..  
--------------------------------------------  
[#Mphasis](https://www.facebook.com/hashtag/mphasis?source=feed_text&epa=HASHTAG):  
---------  
1.Various states of hibernate.  
------------------------------------------------------------------  
#Infosys:  
---------  
1.How do you handle Runtime Exception in JSP.

The errorPage attribute of the page directive can be used to redirects the browser to an error processing page, when uncaught **exception** is encountered. redirects the browser to the **JSP** page error. **jsp** if an uncaught **exception** is encountered during the request processing.

2.What is IOC.  
3.Setter injection vs constructor injection.  
4.What is WSDL?.  
5.What is XML?.  
---------------------------------------------------------------------

1.Difference between exception and error?  
2.Serialization and deserialization?  
3.how can you increase memory size in java.

4.how can you increase heap size in java?  
A:

## How to increase heap size in Java

Default size of Heap space in Java is 128MB on most of 32 bit Sun's [JVM](http://javarevisited.blogspot.sg/2011/12/jre-jvm-jdk-jit-in-java-programming.html) but its highly varies from JVM to JVM e.g. default maximum and start heap size for the 32-bit Solaris Operating System (SPARC Platform Edition) is -Xms=3670K and -Xmx=64M and Default values of heap size parameters on 64-bit systems have been increased up by approximately 30%.  
  
Also, if you are using throughput garbage collector in Java 1.5 default maximum heap size of JVM would be Physical Memory/4 and default initial heap size would be Physical Memory/16. Another way to find default heap size of JVM is to start an application with default heap parameters and monitor in using JConsole which is available on JDK 1.5 onwards, on VMSummary tab you will be able to see maximum heap size.  
  
-Xms<size> set initial Java heap size  
-Xmx<size> set maximum Java heap size  
-Xss<size> set java thread stack size  
java -Xms16m -Xmx64m ClassName.

5.what data base you will use trigger or index?  
6.What is singleton and where it is used?

In Java the **Singleton** pattern will ensure that there is only one instance of a class is created in the Java Virtual Machine. It is **used** to provide global point of access to the object. In terms of practical use **Singleton** patterns are **used** in logging, caches, thread pools, configuration settings, device driver objects

7.Key value pair?  
8.How can you compare two strings?  
9.Can main method be overloaded?.  
10.How many ways we can define a Thread.

----------------------------------------------------------------------------  
1.What is hashing in java?  
2.What is polymorphism in java?.  
3.class Test3  
{  
public void m1()  
{  
System.out.println(" I am in parent class instance method m1 ");  
}  
public static void m2()  
{  
System.out.println(" I am in parent class instance method m2");  
}  
}

class Test extends Test3  
{  
public void m1()  
{  
System.out.println(" I am in child class instance method m1 ");  
}  
public static void m2()  
{  
System.out.println(" I am in child class instance method m2");  
}  
public static void main(String [] args)  
{  
Test3 t3=null;  
//t3.m1();  
t3.m2();  
Test3 t4= new Test();  
t4.m1();  
t4.m2();  
Test3 t5= new Test3();  
t5.m1();  
t5.m2();  
}  
}  
o/p:  
4.Annotations in Hibernate?

### JPA Annotations for mapping java object to database table

Let us look at some of the important JPA annotations. Note that these annotations are present in javax.persistence package.

1. javax.persistence.Entity: Specifies that the class is an entity. This annotation can be applied on Class, Interface of Enums.
2. import javax.persistence.Entity;
3. @Entity
4. public class Employee implements Serializable {
5. }
6. @Table: It specifies the table in the database with which this entity is mapped. In the example below the data will be stores in the “employee” table. Name attribute of @Table annotation is used to specify the table name.
7. import javax.persistence.Entity;
8. import javax.persistence.Table;
9. @Entity
10. @Table(name = "employee")
11. public class Employee implements Serializable {
12. }
13. @Column: Specify the column mapping using @Column annotation. Name attribute of this annotation is used for specifying the table’s column name.
14. import javax.persistence.Column;
15. import javax.persistence.Entity;
16. import javax.persistence.Table;
17. @Entity
18. @Table(name = "employee")
19. public class Employee implements Serializable {
21. @Column(name = "employee\_name")
22. private String employeeName;
23. }
24. @Id: This annotation specifies the primary key of the entity.
25. import javax.persistence.\*;
26. @Entity
27. @Table(name = "employee")
28. public class Employee implements Serializable {
29. @Id
30. @Column(name = "id")
31. private int id;
32. }
33. @GeneratedValue: This annotation specifies the generation strategies for the values of primary keys.
34. import javax.persistence.\*;
35. @Entity
36. @Table(name = "employee")
37. public class Employee implements Serializable {
39. @Id
40. @Column(name = "id")
41. @GeneratedValue(strategy=SEQUENCE, generator="ID\_SEQ")
42. private int id;
43. }
44. @Version: We can control versioning or [concurrency](https://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers) using this annotation.
45. import javax.persistence.\*;
46. @Entity
47. @Table(name = "employee")
48. public class Employee implements Serializable {
49. @Version
50. @Column(name = "version")
51. private Date version;
52. }
53. @OrderBy: Sort your data using @OrderBy annotation. In example below, it will sort all employees\_address by their id in ascending order.
54. @OrderBy("id asc")
55. private Set employee\_address;
56. @Transient: Every non static and non-transient property of an entity is considered persistent, unless you annotate it as @Transient.
58. @Transient
59. Private int employeePhone;
60. @Lob: Large objects are declared with @Lob.
62. @Lob
63. public String getEmployeeAddress() {
64. return employeeAddress;
65. }

The above set of annotation are most commonly used JPA annotations to define an entity.

### Hibernate Annotations for Mapping between tables

We have another set of annotations that are used to specify the association mapping between different tables and entities.

We will take an example considering the below mentioned scenario.

* Tables ’employee’ and ’employeeDetail’ have one-to-one association and they share the same primary key.
* Tables ‘communication’ and ‘communicationDetail’ are linked by a foreign key. It is also a one to one association.
* Tables ‘communication’ and ’employee’ are linked using a foreign key in many-to-one association with communication being the owner.
* Tables ’employee’ and ’employeeStatus’ are linked through a foreign key in many-to-one association with employee being the owner.

@OneToOne  
Employee and EmployeeDetail entities share the same primary key and we can associate them using @OneToOne and @PrimaryKeyJoinColumn.  
In this case the id property of EmployeeDetail is not annotated with @GeneratedValue. The id value of Employee will be used for used for id of EmployeeDetail.

@Entity

@Table(name = "employee")

public class Employee implements Serializable {

@Id

@Column(name = "id")

@GeneratedValue

private int id;

@OneToOne(cascade = CascadeType.MERGE)

@PrimaryKeyJoinColumn

private EmployeeDetail employeeDetail;

}

@Entity

@Table(name = "employeeDetail")

public class EmployeeDetail implements Serializable {

@Id

@Column(name = "id")

private int id;

}

Points to note:

* @PrimaryKeyJoinColumn should be used for associated entities sharing the same primary key.
* @JoinColumn & @OneToOne should be mappedBy attribute when foreign key is held by one of the entities.

Communication and CommunicationDetail are linked through a foreign key, so @OneToOne and @JoinColumn annotations can be used. In snippet mentioned below, the id genereated for Communication will be mapped to ‘communication\_id’ column of CommunicationDetail table. @MapsId is used for the same.

@Entity

@Table(name = "communicationDetail")

public class CommunicationDetail implements Serializable {

@Id

@Column(name = "id")

@GeneratedValue

private int id;

@OneToOne

@MapsId

@JoinColumn(name = "communicationId")

private Communication communication;

}

@Entity

@Table(name = "communication")

public class Communication implements Serializable {

@Id

@Column(name = "ID")

@GeneratedValue

private Integer id;

@OneToOne(mappedBy = "communication", cascade = CascadeType.ALL)

private CommunicationDetail communicationDetail;

}

@ManyToOne  
Many employees can share the same status. So, employee to employeeStatus is a many to one relation. @ManyToOne annotation can be used for the same.

@Entity

@Table(name = "employee")

public class Employee implements Serializable {

@ManyToOne

@JoinColumn(name = "statusId")

private EmployeeStatus status;

}

@OneToMany  
Employee to Communication will be a one-to-many relationship. The owner of this relationship is Communication so, we will use ‘mappedBy’ attribute in Employee to make it bi-directional relationship.

@Entity

@Table(name = "employee")

public class Employee implements Serializable {

@OneToMany(mappedBy = "employee", fetch = FetchType.EAGER)

@OrderBy("firstName asc")

private Set communications;

}

@PrimaryKeyJoinColumn  
This annotation is used to associate entities sharing the same primary key.

@Entity

@Table(name = "employee")

public class Employee implements Serializable {

@Id

@Column(name = "id")

@GeneratedValue

private int id;

@OneToOne(cascade = CascadeType.MERGE)

@PrimaryKeyJoinColumn

private EmployeeDetail employeeDetail;

}

@JoinColumn  
@JoinColumn annotation is used for one-to-one or many-to-one associations when foreign key is held by one of the entities.

@ManyToOne

@JoinColumn(name = "statusId")

private EmployeeStatus status;

@JoinTable: @JoinTable and mappedBy should be used for entities linked through an association table.

@MapsId: Two entities with shared key can be persisted using @MapsId annotation.

@OneToOne

@MapsId

@JoinColumn(name = "communicationId")

private Communication communication;

### Hibernate Annotations for inheritance mapping

Now let us try to understand the inheritance mapping annotation in Hibernate.

Hibernate supports the three basic inheritance mapping strategies:

* table per class hierarchy
* table per subclass
* table per concrete class

we will consider example for each type.

1. Table per class hierarchy – single table per Class Hierarchy Strategy.
2. @Entity
3. @Inheritance(strategy=InheritanceType.SINGLE\_TABLE)
4. @DiscriminatorColumn(name="cartype", discriminatorType=DiscriminatorType.STRING )
6. @DiscriminatorValue("Car")
7. public class Car { }
9. @Entity
10. @DiscriminatorValue("BMW")
11. public class BMW extends Car { }
12. Table per class/subclass – joined subclass Strategy.
13. @Entity
14. @Inheritance(strategy=InheritanceType.JOINED)
15. public class Ship implements Serializable {}
17. @Entity
18. @PrimaryKeyJoinColumn
19. public class Titanic extends Ship {}
20. Table per concrete class.
21. @Entity
22. @Inheritance(strategy = InheritanceType.TABLE\_PER\_CLASS)
23. public class Aeroplane implements Serializable {}
24. @DiscriminatorColumn: As the name suggests this column is the descriminator and this annotation specifies the discriminator column for the SINGLE\_TABLE and JOINED Inheritance mapping strategies.
25. @Entity
26. @Inheritance(strategy=InheritanceType.SINGLE\_TABLE)
27. @DiscriminatorColumn(name="cartype", discriminatorType=DiscriminatorType.STRING )

That’s all for JPA and Hibernate annotations.

5.What is underlying data structure of hash map?.

Implement a hash-map data structure from scratch , where both key and value are of string data type.

The details of how a hash works can be found in Chapter 11 of the book "Introduction to

Algorithms" by Cormen, Leiserson, Rivest. You should also refer to Section 3.7 of "The

Algorithm Design Manual" by Steven Skiena. For the required hash-map implementation,

the following conditions hold true:

1. The key is made up of lower-case english alphabets only (a,b,c...z). It can be of any

length.

2. Values are of string data type.

3. The hash function to be used is the one given in Section 3.7 of Skiena.

4. Choose a suitable size of the hash-map, that is, the number of buckets. It should be

greater than 100.

4. Collisions will be resolved using Chaining. Doubly linked lists will be used to store

colliding entries.

You will have to implement the following operations on the hash-map:

a. Create an empty hash-map.

b. Insert a (key, value) pair into the hash-map.

c. Delete a (key) from the hash-map (if present).

d. Search for a (key) in the hash-map, and if present return its value. Else return null.

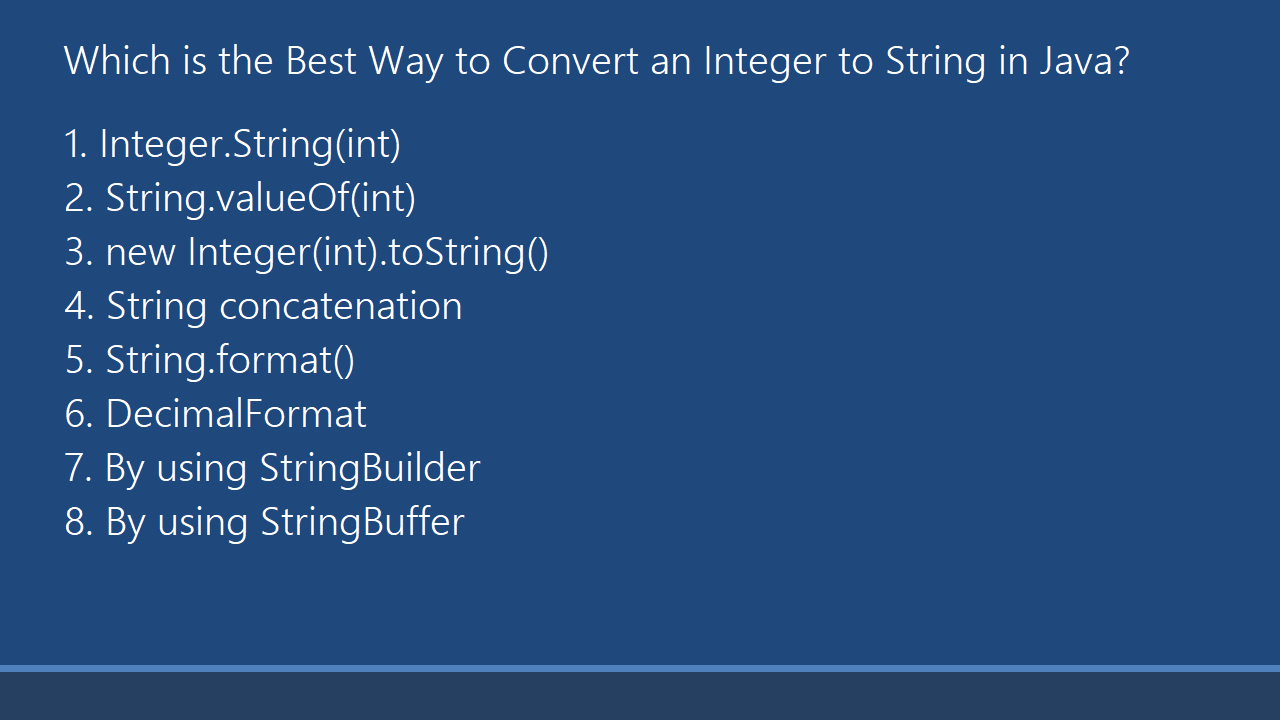
Thanks in advance.

6.What are the collections you use?.  
7.What is underlying data structure of arraylist?

**ArrayList**. The **ArrayList** class is an array-based implementation of the List interface. Specifically, all elements of an **ArrayList** are stored in a Java array. For example, an **ArrayList** named words has its **underlying** array of the size n.

8.Checked vs unchecked exceptions?  
9.how do you validate the user is valid or not to your application?.  
10.What are the design patterns you used in java?.

#Carlon Technologies:  
---------------------  
1)how to convert integer to string in java with example?

  
2)What is polymorphism in java?  
3)What is connection pooling?.  
4)What is HQL?.  
5)What are types of exception in java?  
6)What is difference between IOC and dependency injection?  
7)What is autowiring?.  
8)What is immutable?.

#Capgemini:  
-----------------  
1.What is polymorphism?  
2.class Test  
{  
public int m1()  
{  
System.out.println("Exception");  
return 10;  
}  
public int m1(int x)  
{  
System.out.println("Exception");  
return 10;  
}  
3.What is exception and how many types of exceptions are availble?  
4.if we insert null in hash map as key and while retriving will it give any Run time error?

Work fine

#SLK software:  
------------------  
1.What is composition and aggregation?  
2.What is the super class of java?  
3.Internal implemenation of hashmap?  
4.What will happen if there is collision in values of map?  
5.What is static key word in java?  
6.What are JSP implict objects?  
7.Spring MVC flow  
8.How can configure spring in web.xml?

<servlet>

    <servlet-name>SpringController</servlet-name>

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

    <init-param>

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

        <param-value>/WEB-INF/spring-mvc.xml</param-value>

    </init-param>

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

</servlet>

<servlet-mapping>

    <servlet-name>SpringController</servlet-name>

    <url-pattern>\*.htm</url-pattern>

</servlet-mapping>

9.What is dependency injection?  
10.How can you create your own immutable class?.  
11.Scopes of spring?  
12.Is the class needs to be singleton?.

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

1.What is difference between hashmap and concurrenthashmap?  
2.Methods present in java.lang.Object?  
3.Different types of clonning?.  
4.What is blocking Queue?.  
5.Differences between comparable and comparator?.  
6.Ways of implementing threads?.  
7.Exceutor service?.  
8.Thread Local?.  
9.What is the use of thread pool?  
10.What happens if i clone singleton object?.  
11.How can you make multi threaded as synchronized in singleton object?.  
12.What is serialization and how we can make use of transisent keyword?  
13.What will happen if i deserialize the variable which is already serialized?.  
14.Exception Hierachy?.  
15.Throw vs throws?  
16.How can you create our own customized exception?.  
17.Name some of the checked exception?.  
18.Classloaders in java?.  
19.Pattern and matcher?  
20.What is fail safe and fail fast means?  
21.What will happen if i insert duplicate keys in hashmap?.

Spring:  
-------  
1.What is DI?  
2.What are the scopes?.  
3.Difference between request and prototype?.  
4.Difference between singleton and prototype?.  
5.How can you configure multiple properties file?.

Loading multiple properties files in spring framework can be accomplished in following ways.

* JavaConfig approach using @PropertySources annotation
* XML configuration approach using context:property-placeholder

## **Sample Property files**

**env.properties:**

jdbc.url=127.0.0.1

jdbc.userid=

jdbc.userid.default=admin

jdbc.port=1521

**env2.properties:**

jdbc.userid=codesimplify

jdbc.port=1111

## **1. JavaConfig approach using @PropertySources annotation**

### **1.1 Define @PropertySources annotation**

@PropertySources annotation is expected to be defined on configuration class(@Configuration annotated class). @PropertySources acts as container annotation that aggregates multiple [@PropertySource](http://www.codesimplify.com/spring-core/spring-javaconfig-propertysource-example/) annotations.

package com.codesimplify.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.PropertySource;

import org.springframework.context.annotation.PropertySources;

@Configuration

@PropertySources({

@PropertySource("classpath:env.properties"),

@PropertySource("classpath:env2.properties")

})

public class AppConfig {

}

**NOTE:**If you wanted to use Property placeholder ‘$’, you are expected to define **PropertySourcesPlaceholderConfigurer**bean. For more information, see section 2.2 in [Spring @Value annotation usage using SpringEL](http://www.codesimplify.com/spring-core/spring-value-usage-using-springel/)

6.What is autowiring?.  
7.What will happen if i make @autowired as false?  
8.What is application context?.  
9.What is Advice?  
10.How do you write a programm for AOP?.

Webservices:  
--------------  
1.Difference between soap and rest?.  
2.How can you declare different response types in REST?.  
3.How we can implement security in webservices?.  
4.What implementations did you use for soap?

#Vodafone:  
----------  
1.How can you integrate Spring and hibernate?.  
2.How hashmap works?.  
3.What is difference between get and load?.  
4.How can you implement transactions?.

1) Explian Your Project Arch?

2)Explain about Layers used in your project?

3)In which layer you are involed?

4)What are technologies are used in your Project?  
//scamstandup mettings,  
5)Are your attended any meetings?  
If your attending then how much time ?

6)Your Client name?  
//  
7)What is your HR name and TL names etc...?

8)How security is implemented in your project?

9)What are the exceptions handled and avoided?

10) How many modules are there in your project?

11)How many roles (like admin,customer etc..)  
are there?

12)How many tables are there?

13)What is need of webservice for your project?

14) Which implementation used from webservices?

15)Total Duration of Your Project?

16)What is scrum and scrum master?

17)How the tasks are assigning ?

18)What is sprint and story -points ?

19)what is difference between  
agile and waterfall model?

20) Sample Unit Test cases of Your Project?

21)From Which Repository your checkedout Project?  
(like SVN/CVS/GIT etc..)

22)Some of Design Patterns  
(DAO,Factory,AbstractFactory,  
Front-Controller,BussinessDelegate,VO,BO,Proxy,  
Facade,Command Object,ViewHelper,Singleton,  
ServiceLocator,Strategy )

23)Questions On Tools (Maven,Junit,Log4j,SVN )

\*\*\*\*24) How to resolve the Confilicts in SVN ?

25)svn term : Checkout,commit,update ?

26) what is trunk,branch,tags?

27) What are the issues your solved in your project?

28) Tell me some problems you faced  
while developing the code?

29)Where method overloading  
and method overriding implemented in your project?

30)Is your created any thread in your project?  
if created then what is need to create that Thread?

31) any where synchronoziation is  
implemented in your project?

32)All the collection questions ?

33) String,StringBuffer and string Builder questions?

34) JSTL tags?

35)What is Dependency Injection  
and why dependency inj?

36) How many containers are there in spring?

37) what is IOC?

38) Spring Mvc flow and All  
the components of Spring MVC?

39)Why we are using  
spring-boot and spring-batch?

40) how to implement a cron job?

41)what is difference between  
SOAP and Restfull webservices?

42)Commons questions on Servlet,JSP and JDBC?

43) What is difference between get and load() ?

44) Questions HQL,criteria,Native-SQL?

45)Caching tech'q question from Hibernate?

46)Locking mechanism in Hibernate and DB?

47)SessionFactory and Session in hibernate?

48) HibernatTemplate is better (OR) Plain Hibernate API  
is better?

49) How to increase JVM heap Size  
and How to solve HeapOut of Memory Error?

50) Connection Pooling realeated questions?

51)Transatcion Management related questions?

52) what is root resource and resource method?

53)where we can use GET Where we can use POST?

**1) Explain your project flow? What technologies used in your previous project?**

**2) How many modules are there in your project?**

**3) What is your role and responsibilities in the project?**

**4) What are the corporate trainings you have attended?**

**5) What are the tools that were used in your project?**

**6) Explain about check-in and check-out in version control tools?**

**7) What are the IDEs you have worked on?**

**8) What are the Servers you have worked on?**

**9) Have you ever been involved in Testing?**

**10) What are the different phases in delivering the project during development and during maintenance?**

**(CTS, Patni, Galaxe Systems, General Motors)**

**11) Do you have experience in Production Support and what is the hierarchy in Production Support?**

**(HCL, RBS, Amdocs(Gurgaon), HSBC)**

**12) Have you ever been involved in Design?**

**13) What is the reporting tool in your project?**

**14) What is your team size?**

**15) What was the criticism/difficulties you got during your projects, how did you handle it and what are the measures you        took to avoid repetition of the same?**

**(Mecedes Benz, ORACLE, CA, Pega Systems)**

**16) How do you write Unit Test Cases?**

**(Microsoft, Google, Tanla, Adobe, Bharti Telecom)**

**17) What is STQC testing?**

**(For all Govt. projects)**

**18) How do you perform impact analysis for your code?**

**(Microsoft, EmcSquare, Honeywell, INTEL, Symphony, TCS, CTS, Amazon.com, NOKIA)**

**19) Why Analysis & Testing phases are very important?**

**(Infosys, Mphasis)**

**20) What is the document that should be consulted to know about your project, the activities you do, to know about your        project, the activities you do, your schedules and milestones?**

**(Dell, Qualcomm, HCL, MindTree, PSPL, TechMahindra)**

**21) What is the reporting hierarchy in your project?**

**(all CMM Level 5 companies)**

**22) Tell me some complex code you have written?**

**(convergys)**

**23) Which type of validations you have done in your Project?**

**(convergys)**

**24) What are performance issues / Performance issues you have faced in your project?**

**(Encore software)**

**25) How did you provide security to your projects and some modules?**

**(IBM)**

**26) What versions of softwares you are using in your project?**

**(IBM)**

**27) What are the exceptions you faced in your project?**

**(IBM)**

  1) What difference LinkedList and ArrayList?

           2) What is difference Iterator and Enumeration?

           3) What is an abstraction?

           4) What is an Abstract class and interface?

           5) What is main purpose Garbage Collection ?

  1) What are the oops concepts and explain them?

              2) They can given some programs on Overloading concepts with Exceptions?

              3) Difference between Interface and Abstract Class?

              4) Which version of java we are using? and what are the features in 1.5                   version and explain them?

              5) what is Cloning? why we are using Cloning concept? and How many types                   of cloning?

              6) What is Exception? Difference between Exception and Error?

              7) How many types of Exceptions and explain and given some examples?

              8) How we will handle Exceptions?

              9) How to write Customized Exceptions?

            10) They are given some programs on Try, Catch and Finally?

            11) Explain about Thread life cycle?

            12) What is Difference between Sleep and Wait methods?

            13) What is the Difference between Notify and NotifyAll methods?

            14) What is Yield method?

            15) What is Synchronization?

            16) can you draw the Collection Hierarchy?

            17) What is Difference between list and Set?

            18) What is Difference between ArrayList and Vector?

            19) How we can sort the elements in HashSet?

            20) What is Difference between Comparable and Comparator?

            21) What is Difference between Hashmap and HashTable?

            22) How we can Synchronize List and set?

            23) When we use Enumeration, Iterator and ListIterator?

            24) What is the difference String and StringBuffer and StringBuilder?

            25) what are the methods are available in Object Class?

**After these they come to the Advanced Java(Jdbc, Servlets, jsp)**

             1) How many Drivers are there in Jdbc? Which Driver we are using in your                  project?

             2) Can you write the code for Connecting the database?

             3) What is Connection pooling mechanism? How we can implement?

             4) Difference between Statement and PreparedStatement?

             5) What is Callable Statement?

             6) What is Servlet Life Cycle?

             7) If iam sending the request at in servlet life cycle which method will be                  called?

             8) What is the difference between Forward and include Mechanisms?

             9) What is the difference between Forward and SendRedirect Mechanisms?

           10) What is the difference between Servlet Config and servlet Context?

           11) What are the Implicit objects in jsp?

           12) How we can Handle the Exceptions in Jsp?

           13) What are the scopes in jsp?

**While coming to Struts**

             1) Explain Struts Flow?

             2) What is I18N?

             3) What are the Actions are available in Struts?

             4) What is the difference between DispatchAction and LookupDispatchAction?

             5) What is Switchaction?

             6) What is MVC?

             7) Struts follows which MVC Architecture?

             8) What is FrontContoller?

             9) What is Singleton Design Pattern? can you write some code about this?

           10) What is DAO and Explain?

           11) Which Tags are there in Struts-config.xml file?

           12) Difference between Struts1.2 and Struts2.0?

           13) How you provide the validations?

**While Coming to PROJECT MANAGER ROUND**

             1) Explain about your project?

             2) How many Modules are there in your project?

             3) What's your Role and Responsibilities?

             4) How big is your Team Size?

             5) Tell me Some Exceptions you faced in your project?

             6) Which type of Complexity code you written in your project

             7) Which Version Controller you used?

             8) Which type of validations you done in your Project?

            Etc.

 1.What is inheritance?

         2. What is Abstraction ?

         3. What is difference between C++ and Java?

         4. What are types driver in Jdbc

         5. What is difference between Abstract class and inner class ?

         6. Write code to delete the cookies?

         7. What is servlet and JSP?

         8. What is struts?

         9. What is Jvm?

       10. What is super and this keyword?

       11. What is constructor and how to initialize an Object?

       12. What is function overloading ?

       13. What is class and object?

       14. What is collections?

       15. What is Iterator?

       16. What is final Key word?

       17. What is main purpose of Garbage Collectior?

       18. What is an object-oriented Programming?

**Here is the programs how they asked**

               1) write a program to insert a node at rear end in double linked list?

               2) write a program to reverse the lined list?

               3) write a program to find substring using pointers?

               4) write a program for palindrome using recursion ?

               5) write a program for to find repeated letter in given file?

**some technical questions**

               1.Describe how procedure of c program -like compilation every thing...

               2.what is macro write a program to find largest b/w 3 elements by using                   macro?

               3.diff b/w in declaration and definition?

               4.tell memory structure allocation process and how it works?

               5.what is DMA? what it means?

               6.what is function pointer?

               7.what is difference b/w structure and union?

               8.write a program for conversion of little endian to big endian?

               9.what is splay tress and some question about tress?

              10.can u write a print a statement with out using print function ?

              11.They told me to write a program to find mid point of rectangle by just

  1. Explain features of Java?

             2. Explain about polymorphism?

             3. What is the purpose of ArrayList?

             4. Explain about Hibernate Session Factory?

             5. Explain about inheritance and polymorphism in hibernate?

             6. Explain about Hibernate Architecture?

             7. What is mutable and immutable?

             8. What are performance issues you have to faced in your project?

  1. Explain about Hibernate SessionFactory ?

         2. Explain about inheritance & polymorphism in hibernate?

         3. Explain about Hibernate Architecture ?

         4. Explain about Hibernate mappings ?

         5. What is meant by component mapping?

         6. What are the problems you faced while using hibernate in your project?

         7. Explain about Lazy Loading?

         8. Explain differences between get() and load()?

         9. What is difference between First level cache and Second level cache?

        10. What is the difference Session and SessionFactory?

        11. How can we connect more than one database in hibernate applications?

        12. Difference between Dialect and Driver?

        13. Is it possible to write hibernate application without using hibernate mapping file?

        14. Is it possible to write hibernate application without using hibernate configuration               file ?

        15. Explain about Persistance class life cycle states in hibernate?

        16. Explain about hibernate tools?

        17. what is hibernate?

        18. What is disadvantage using hibernate?

        19. What is ORM?

  1) write a program to insert a node at rear end in double linked list?

               2) write a program to reverse the lined list?

               3) write a program to find substring using pointers?

               4) write a program for palindrome using recursion ?

               5) write a program for to find repeated letter in given file?

**some technical questions**

               1. describe how procedure of c program -like compilation every thing...

              2.what is macro write a program to find largest b/w 3 elements by using                                 macro?

              3.diff b/w in declaration and definition?

              4.tell memory structure allocation process and how it works?

              5.what is DMA? what it means?

              6. what is function pointer?

              7.what is difference b/w structure and union?

              8. write a program for conversion of little endian to big endian?

              9.what is splay tress and some question about tress?

              10.can u write a print a statement with out using print function ?

              11.They told me to write a program to find mid point of rectangle by just

  1. Tell me about your Second project and draw its architecture?

           2. Explain about Oops?

           3. Encapsulation is keyword or class or interface?

           4. How many ways are possible to create a Thread ? which way is better?

           5. Explain about Servlets? what are the life cycle methods explain?

           6. Among Servlets and JSPs which is better? Where JSP Technology is best suitable.

           7. Tell me about jsp Elements?

           8. What is difference b/w scriptlet and declaration?

           9. Explain about struts flow?

         10. What are Action classes available in struts?

         11. What is DispatchAction? where did you use in your project? where it configure               write code?

         12. How to write form bean tags in your struts-configuration file?

         13. I have 4 action classes i want get second action class how can i get it? where are               you configure? can u write code ?

         14. Explain about SingleThreadModel in servlets?

         15. Explain about collection overview?

         16. What is difference b/w HashMap and Hashtable?

         17. What is differance b/w ArrayList and Vector?

         18. What are methods available in Collection interface?

         19. What is serialization? How can you achieve it?where you used in your project?

         20. What is Marker interface?

         21. How can you achieve encapsulation your project?

         22. Explain about Spring?

         23. What is difference b/w Arrays and Collections?

   1. What is the Hibernate? What are files we have requires doing a hibernate               application?

          2. Explain about Hibernate Architecture?

          3. Explain your project flow? What technologies used in your previous project?

          4. Explain about Spring Framework features?

          5. How do you configure using spring mvc with DAO class?

          6. What is Abstraction, inheritance polymorphism?

          7. What are Checked vs Unchecked Exceptions?

          8. What is AutoBoxing and AutoUnboxing ? Can you Explain with real time example?

          9. Explain differences between Abstract class vs interface?

         10. Explain differences between HashMap and Hashtable?

         11. try{ some code }

               catch(Exception e) { some code }

               finally{ some code }

               which one is optional?

         12. try{ some code; system.exit(0); }

               Catch(Exception e){ Some code }

               finally{ some code }

               what will happen?

         13. What is the difference b/w extends Thread class and implements Runnable               interface?

         14. Explain about Servlet life cycle ?

         15. How to include one jsp into anther jsp? Which tag we can use for this purpose ?

         16. Explain about Struts?

         17. Explain Singleton Design pattern?

         18. How to connect more than one database in Hibernate?

         19. I want change my database from Oracle to MySQL in hibernate. Which               configuration changes are required and which classes we have to use?

         20. Which tools you used in project?

         21. What are version control tools used in your project? Explain about check-in and               check-out?

         22. What is your team size?

         23. How to build modules in your project? Are you used any tool for this?

         24. What is Sax parser and Dom parser?