[What Is Agile?](#_Toc31105)

[Agile Methodologies](#_Toc17472)

[Agile Components](#_Toc6332)

[Agile Scrum process](#_Toc1105)

[Agile Estimation](#_Toc12522)

[Rally workflow](#_Toc7622)

[Hashing: How Hash Map Works In Java Or How Get() Method Works Internally](#_Toc32044)

[Comparator](#_Toc13112)

[HashSet vs TreeSet](#_Toc460)

[Iterator vs List Iterator](#_Toc15250)

[HashMap vs TreeMap](#_Toc30382)

[Difference between Stack vs Heap in Java](#_Toc16711)

[Difference between ArrayList and LinkedList:](#_Toc3619)

[Strengths](#_Toc71)

[Weakness:](#_Toc17314)

[Why should we hire you](#_Toc109)

[What are Joins?](#_Toc30957)

[Restful Services:](#_Toc30839)

[RESTful Web Services are REST architecture based web services. In REST Architecture everything is a resource. RESTful web services are light weight, highly scalable and maintainable and are very commonly used to create APIs for web based applications.](#_Toc7593)

[Soap and Rest Web Services](#_Toc26070)

[Testing Rest Services](#_Toc15478)

[Stack Memory vs Heap Memory](#_Toc26574)

[Stack is allocated for saving primitive data types](#_Toc25098)

[Heap memory is the memory allocated for Object](#_Toc8382)

[Java Stack memory is used for execution of a thread. They contain method specific values that are short-lived and references to other objects in the heap that are getting referred from the method.](#_Toc13785)

[Stack memory is always referenced in LIFO (Last-In-First-Out) order. Whenever a method is invoked, a new block is created in the stack memory for the method to hold local primitive values and reference to other objects in the method. As soon as method ends, the block becomes unused and become available for next method.](#_Toc13965)

[Stack memory size is very less compared to Heap memory.](#_Toc6484)

[What is spring?](#_Toc5000)

[What is spring bean?](#_Toc2500)

[A bean definition can contain a lot of configuration information, including constructor arguments, property values, and container-specific information such as initialization method, static factory method name, and so on.](#_Toc21306)

[Type of DI](#_Toc8773)

[2) What are the advantages of spring framework?](#_Toc25354)

[3) What are the modules of spring framework?](#_Toc18043)

[Autowiring in Spring](#_Toc23702)

[Advantage of Autowiring](#_Toc1003)

[Autowiring Modes](#_Toc4140)

[Jmeter Interview Questions](#_Toc26400)

[How to create a Jmeter Test Plan](#_Toc10192)

[ Create a Test Plan](#_Toc11017)

[ Create a Thread Group](#_Toc20401)

[o Configure the no of threads](#_Toc7217)

[o Ramp up period](#_Toc30456)

[o Loop Count](#_Toc5820)

[ Under Thread Group](#_Toc5204)

[o Add Samplers like](#_Toc32082)

[o HTTP Request,](#_Toc29622)

[o JDBC Request,](#_Toc14430)

[o LDAP request](#_Toc32535)

[o FTP Request](#_Toc2125)

[o SMTP Sampler](#_Toc16956)

[o JUNIT Request](#_Toc2731)

[o JAVA Request](#_Toc10712)

[ Under Thread Group or Test Plan add listeners for capturing the result](#_Toc4439)

[o Like Summary report](#_Toc16505)

[o View Results in table](#_Toc9250)

[o View Results in tree.](#_Toc37)

[ Performance Parameters](#_Toc2197)

[o No of samples](#_Toc17808)

[o Start time](#_Toc18606)

[o Load Time](#_Toc6737)

[o Sample time](#_Toc14558)

[o Throughput](#_Toc30088)

[o Latency](#_Toc16006)

[o Status](#_Toc20167)

[o Errors](#_Toc25539)

[o Response Code](#_Toc32069)

[Delete vs Truncate vs Drop](#_Toc27218)

[Delete will the rows based on some conditional where class or else deletes the entire table followed by Commit or Rollback and does not clear the table space](#_Toc28063)

[Truncate will delete and clears the Table space](#_Toc24268)

[Drop will remove the table and all the contents.](#_Toc19127)

[Autimation metrics](#_Toc30351)

[What is the difference between preemptive scheduling and time slicing?](#_Toc23151)

**What Is Agile?**

Agile is a methodology that helps teams respond to unpredictability through incremental, iterative approached known as sprints

**Agile Methodologies**

* Agile
* Kanban

|  |  |
| --- | --- |
| **Scrum** | **Kanban** |
| Team commitment | Team commitment is optional |
| Consists of cross functional team | Cross functional is optional. Specialist teams are preferred |
| Items will be broken down and committed to sprint | No item size prescribed |
| Burn down chart prescribed | No particular type of diagram is preferred |
| Estimation prescribed | Estimation Optional |
| Cannot add items in a current iteration | Can add items where ever capacity is available |
| Prescribed ROLES(product Owner/Scrum Master/team) | No prescribed role |
| Product Backlog prioritization is prescribed | Prioritization is Optional |

**Agile Components**

**Product Backlog**

* List all the EPIC
* Break down the epic to User Story

**Sprint Backlog & Sprint Planning**

* Pick and prioritize the user stories from product backlog

***Release Forecast*** *Use the release forecast to review product backlog*

**Sprint Burn down**

**Sprint Demo**

**Retrospective Meeting.**

**Agile Scrum process**

* Derive Product Backlogs
* Define User stories
* Sprint categorize
* Story Sizing & Capacity planning (using poker cards) - Estimation
* Tasks

**Agile Estimation**

* Story cards
* T-shirt sizes

**Rally workflow**

Work Space >> projects >> Plan>Userstory>>Testcases/defects

**Automation in Agile TDD**

**Automation Strategy**

Automation Strategy is the document that is created for the team to guide them from technical front of Automation

1. Scope
2. Kind of Automation testing (performance, UI, Security testing)
3. Automation candidates & Non Automation candidates’ identification criteria
4. Features to be automated and not to automated
5. Identify automation and non-automation candidates
6. Automation framework Architecture
7. Automation Environment
8. Test case design coding guidelines
9. Automation framework and Automation script guidelines
10. Metrics

**Framework considerations**

- Ease of Use

- Object Oriented Framework

- Application-Independent

- Easy Maintenance

- Pluggable framework components

- Reusable components

- Loosely coupled & Independent components

- Encapsulate the Framework implementation team from framework complexities

- Isolate the test data and object repository from the Framework Code

-Modularity

-Reusability

- Error handling

- Detailed Audit logs & Reporting

**AUTOMATION metrics**

Automation coverage

Automation Progress

**Metrics categories:**Most software testing metrics (including the ones presented here) fall into one of three categories:

* **Coverage:**meaningful parameters for measuring test scope and success.
* **Progress:** parameters that help identify test progress to be matched against success criteria. Progress metrics are collected iteratively over time. They can be used to graph the process itself (e.g., time to fix defects, time to test, etc.).
* **Quality:**meaningful measures of testing product quality. Usability, performance, scalability, overall customer satisfaction, and defects reported are a few exaq1mples.

**Tool Evaluation**

Nature of the app (webapp, Windows App, Mobile web, Mobile Native App)

Technology the app has been build3rd party communication

Does it provide the object Oriented framework development support

Reporting support

Support for unit testing integration like Junit & Test NG Integration support

**How to define Automation Framework Architecture**

Based on the nature of the application and the API exposed we need to decide on the Automation framework

Based on Product Specific/Project/Enterprise specific

Framework should be application independent

Framework components should be independent and reusable

Framework should be designed as pluggable framework

Frame work should be designed to be an object oriented for achieving modularity and reusability

Test Data Isolation

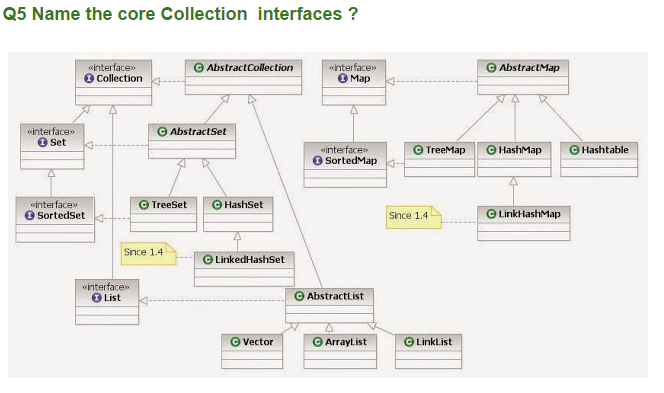
Provision of configuration to OR and Configuration files

CI Integration support

Debugging & Audit log support

Error handling support

Framework Fault Tolerance Support



Important: Collection, Set, Queue, List, and Map

**Q6 what is the difference between List and Set?**

|  |  |
| --- | --- |
| **Set** | **List** |
| Set allows only unique elements | List allows duplicate elements |
| Set is unordered while List is ordered | List maintains the order in which the objects are added |

**Q7 what is the difference between Map and Set?**

|  |  |
| --- | --- |
| **Set** | **Map** |
| Set contains unique keys but does not allow duplicate values | Map contains unique key but allows duplicate values |

**Q8 what are the classes implementing List and Set interface?**

|  |  |
| --- | --- |
| **List** | **Set** |
| ArrayList, Vector, Linked List | HashSet, Tree Set |

**Q9 what is an iterator?**  
Iterator is an interface. It is found in java.util package. It provides methods to iterate over any Collection.

**Q10 what is the difference between Iterator and Enumeration?**

|  |  |
| --- | --- |
| **Iterator** | **Enumeration** |
| Iterator you can remove element(allows read/write operations) | Doesn’t support remove(supports readonly operations) |
|  |  |

**Q12 which methods you need to override to use any object as key in HashMap?**  
  
To use any object as key in HashMap, it needs to implement **equals()** and **hashCode()** method

**Q13 what is the difference between Queue and Stack?**

|  |  |
| --- | --- |
| **Queue** | **Stack** |
| First In First out | Last In First Out |

**Q14 How to reverse the List in Collections?**

Collections.reverse(listobject);

**Q15 How to convert the array of strings into the list?**

asList()

**Q16 what is the difference between ArrayList and Vector?**

|  |  |
| --- | --- |
| **Vector** | **ArrayList** |
| Vector is Synchronized | Array List is not synchronized |
| Vector is slow | ArrayList is fast |
| Vector increases the capacity twice of its initial size | ArrayList increases its ArraySize by 50% |
| Vector supports both Enumeration and Iterator | Array List support only Iterator for traversing |

**Q17 what is the difference between HashMap and Hashtable?**

|  |  |
| --- | --- |
| HashMap | Hashtable |
| HashMap allows null key value insertion | Hashtable does not allow null keys and null values. |
| HashMap is not synchronized or thread-safe | Hashtable is synchronized or thread-safe. |

**Q20 what is the difference between Array and ArrayList in Java?**

|  |  |
| --- | --- |
| **Array** | **ArrayList** |
| Array is static in size | ArrayList is dynamic in size. |
| Array can contain primitive data types | ArrayList cannot contain primitive data types.  If declared primitive data type JVM through Auto boxing(converting primitives to equivalent objects internally) |
| Fast performance | Slow performance |
| does not guarantee ordered  elements. | does not guarantee ordered  elements. |

**inkedList** is fast for adding and deleting elements, but slow to access a specific element. **ArrayList** is fast for accessing a specific element but can be slow to add to either end, and especially slow to delete in the middle. **ArrayList** is essentially an array. **LinkedList** is implemented as a double **linked list**

# **Hashing: How Hash Map Works In Java Or How Get() Method Works Internally**

*HashMap works on the principle of Hashing.  To understand Hashing, we should understand the three terms first   i.e  Hash Function , Hash Value and Bucket .*

*HashMap get(Key k) method calls* ***hashCode*** *method on the key object and applies returned hashValue to its own static hash function to find a* ***bucket*** *location(backing array) where keys and values are stored in form of a nested class called Entry (Map.Entry)*

**HashMap** in **Java** stores both key and value object, in bucket, as an object of Entry class which implements this nested interface Map.Entry. ... When get() method is used to retrieve value, again key object is used to calculate a hash which is used then to find a bucket where that particular key is stored.

|  |  |
| --- | --- |
| **ArrayList** | **Linked list** |
| Manipulation with ArrayList is **slow** | Manipulation with LinkedList is **faster** |
| Ddataccess is faster.Better performance becaue of Index based search | Data access is slower. Performance is slower  As Linked list inherits Double Linked list which requires the traversal through all the elements for searching an element. |
| One should go for Arraylist implemententation when there are more get calls |  |

# **Comparator**

Is an interface which is used for sorting objects in Java.

**Compare to** method

|  |  |
| --- | --- |
| **Comparable** | **Comparator** |
| Comparable provides **single sorting sequence**. In other words, we can sort the collection on the basis of single element such as id or name or price etc. | Comparator provides **multiple sorting sequence**. In other words, we can sort the collection on the basis of multiple elements such as id, name and price etc. |
| Comparable **affects the original class** i.e. actual class is modified. | Comparator **doesn't affect the original class** i.e. actual class is not modified. |
| Comparable is found in **java.lang** package. | Comparator is found in **java.util** package. |
| We can sort the list elements of Comparable type by**Collections.sort(List)** method. | We can sort the list elements of Comparator type by**Collections.sort(List,Comparator)** method. |

# **HashSet vs TreeSet**

|  |  |
| --- | --- |
| HashSet | TreeSet |
| Removes duplicates | Removes duplicates |
| Don’t sort element | Sorts elements in Ascending order |

# **Iterator vs List Iterator**

|  |  |
| --- | --- |
| Iterator | List Iterator |
| Iterator is used for traversing List and Set both. | We can use List Iterator to traverse List only, we cannot traverse Set using ListIterator. |
| We can traverse in only forward direction using Iterator. | Using ListIterator, we can traverse a List in both the directions (forward and Backward) |

# **HashMap vs TreeMap**

|  |  |
| --- | --- |
| **HashMap** | **Tree Map** |
| Return elements as it on how they are added | Return Sorted in Ascending order of Keys |

HashMap vs tree map

Binary tree

Enums

**Hash map usage**

* Create an entry set
* Create a iterator object over entry set
* Then iterate

**LinkedHashMap**

Same as HashMap just preserves the insertion order

**What are concurrentCollectionClasses?**

In jdk1.5 , Java Api developers had introduced new package called **java.util.concurrent** that have thread-safe collection classes as they allow collections to be modified while iterating

Ex : Concurrent Hash Map class

**How will you make Collections readOnly ?**

Collections.unmodifiableMap(Map m)

Collections.unmodifiableList(List l)

Collections.unmodifiableSet(Set s)

## **Difference between Stack vs Heap in Java**

|  |  |
| --- | --- |
| **Stack** | **Heap** |
| Stack memory is used to store variables and functions | Heap memory is used to store Objects |
|  |  |
|  |  |

**What is reflection?**

The ability to examine or modify the runtime behaviour of applications running in the Java virtual machine

* Examine an object's class at runtime
* Construct an object for a class at runtime
* Examine a class's field and method at runtime
* Invoke any method of an object at runtime
* Change accessibility flag of Constructor, Method and Field

**Binary Tree**

A binary tree is a **tree data structure** in which each node has at most two children, which are referred to as the left child and the right child  
[PreOrder traversal](http://www.java2blog.com/2014/07/binary-tree-preorder-traversal-in-java.html)- In PreOrder traversal, each node is processed before either of its sub-trees. In simpler words, Visit each node before its children.  
  
[InOrder traversal](http://www.java2blog.com/2014/07/binary-tree-inorder-traversal-in-java.html): In In-Order traversal, each node is processed between subtrees.In simpler words,Visit left subtree, node and then right subtree.  
  
[PostOrder traversal](http://www.java2blog.com/2014/07/binary-tree-postorder-traversal-in-java.html): In PostOrder traversal,each node is processed after subtrees traversal. In simpler words, Visit left subtree,  right subtree and then node.

[Level order traversal](http://www.java2blog.com/2014/07/binary-tree-level-order-traversal-in.html) : In Level order traversal, tree is traversed by each level. It is same as breadth first search.  
  
[Spiral/Zigzag order traversal](http://www.java2blog.com/2014/08/spiralzigzag-level-order-traversal-of.html): In spiral order traversal, tree is traversed in spiral shape. 

**Binary Search**

Binary search is a fast search algorithm. This search algorithm works on the **principle of divide and conquer.** For this algorithm to work properly the data collection should be in sorted form.

Binary search a particular item by comparing the middle most item of the collection. If match occurs then index of item is returned. If middle item is greater than item then item is searched in sub-array to the right of the middle.

**What is immutable class in Java?**

Immutable classes are those class, whose [object](http://javarevisited.blogspot.com/2012/12/what-is-object-in-java-or-oops-example.html) cannot be modified once created. Immutable and mutable objects are, [**String** and StringBuffer](http://javarevisited.blogspot.com/2011/07/string-vs-stringbuffer-vs-stringbuilder.html).

Class can be made immutable by making it Final.  
**Benefits**

* Thread Safe

**Stack**

Stack is a subclass of Vector collection

Push pop

**Sorting**

**Bubble Sort**

Comparing each pair of adjacent items and swapping them

**Selection Sort**

Tries to identify the smallest and the largest elements and swapping the positions

**Insertion Sort**

**Quick sort in java.**

Quicksort or partition-exchange sort , is a fast sorting algorithm, which is using divide and algorithm. Quicksort first divides a large list into two smaller **sub-lists**: the low elements and the high elements. Quicksort can then recursively sort the sub-lists.

**Merge sort in java.**

Divided into unsorted n no of patterns and

**What is difference between Singly Linked List and Doubly Linked List data structure?**  
Main difference between singly linked list and doubly linked list is ability to traverse. In a single linked list, node only points towards next node, and there is no pointer to previous node, which means you cannot traverse back on a singly linked list. On the other hand doubly linked list maintains two pointers, towards next and previous node, which allows you to navigate in both direction in any linked list.

# **Difference between ArrayList and LinkedList:**

|  |  |
| --- | --- |
| **ArrayList** | **LinkedList** |
| 1) ArrayList internally uses **dynamic array** to store the elements. | LinkedList internally uses **doubly linked list** to store the elements. |
| 1. Manipulation with ArrayList is **slow** because it internally uses array. If any element is removed from the array, all the bits are shifted in memory. 2. One should go for Arraylist implemententation when there are more get calls | Manipulation with LinkedList is **faster** than ArrayList because it uses doubly linked list so no bit shifting is required in memory. |
| 3) 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. |

**Thread.sleep vs Thread.wait vs Thread.Yield**

**Thread.sleep** cannot be revoked

**Thread.wait** can notify thread to continue

**Thread.yieid** will go to runnable/ ready state

Thread.Wait releases the monitor or lock it was holding on that object, but when a thread calls the sleep() method, it never releases the monitor even if it is holding.   
  
Thread.yield() method pauses the currently executing thread temporarily for giving a chance to the remaining waiting threads of the same priority to execute. If there is no waiting thread or all the waiting threads have a lower priority then the same thread will continue its execution

**Volatile**

Any way the **volatile** keyword in **Java** is used as an indicator to **Java** compiler and Thread that do not cache value of this variable and always read it from main memory. ...**Java volatile** keyword cannot be used with method or class and it can only be used with variable.

So if you want to share any variable in which read and write operation is atomic by implementation e.g. read and write in an int or a boolean variable then  you can declare them as volatile variable.  
  
Read more: <http://javarevisited.blogspot.com/2011/06/volatile-keyword-java-example-tutorial.html#ixzz44NxTf9UW>

**Polymorphism**

Polymorphism in Java has two types: Compile time polymorphism (static binding) and Runtime polymorphism (dynamic binding). Method overloading is an example of static polymorphism, while method overriding is an example of dynamic polymorphism.

**Windows scroll**

window.scrollTo(0, document.body.scrollHeight)

Element scroll

WebElement element = driver.findElement(By.linkText("Import/Export")); ((JavascriptExecutor) driver).executeScript(                 "arguments[0].scrollIntoView();", element);

|  |
| --- |
| @Test |
|  | public void testForSoftAssertionFailure() { |
|  | softAssert.assertTrue(false); |
|  | softAssert.assertEquals(1, 2); |
|  | softAssert.assertAll(); |
|  | } |

Technical questions are  
1**. Tell About yourself**

My self Kiran Kumar Kanumuri. Coming to my professional career, I have 11 + years of experience in the software field. I’m with EPAM India formerly Alliance Global Services. From the past 5 yrs.

Currently I’m a Senior Architect involved in design and development of Automation Framework. Currently leading Mobile Automation COE group

Throughout my career, I’m an individual contributor and am passionate about exploring and learning new things in terms of technology and keep myself updated with latest Automation tools & Technologies

I’’ expertize in Open Source tools like webdriver, Appium, I have worked on building Framework using Java and dotnet

**Personal Interests:**

I have keen interest in taking care of Dogs. I really love them

I’m a big fan of Formula1 race. I never miss the live Race on TV (Mika Hakkinen, Fernando Alonso, Kimi Raikkonen and Juan Pablo Montoya)

2. What are your roles & responsibilities?

At EPAM India, My role is an Architect, and leading mobile Automation COE

**Responsibilities**

Create POC, Contribute towards MCOE group in implementing latest mobile automation tools, and support Projects in getting the Automation framework implemented

Design and Develop automation framework  
3. What is your module?  
4. Where did you face difficulty?  
5. Tell me major requirements in your module?  
6. What is process model?  
7. If you are team lead you are having 5 team members, among that 2 people are not doing job perfectly, in that situation what will u do?  
8. How many test cases u wrote for this project?  
9. How many times u take NCR?  
10. How do u perform UAT?  
11. As a client I gave you a dynamic list box how do you test, what type of test cases you will write?  
12. For my above query how can u retrieve information from list box?  
13. What is current project environment?  
14. Why should you perform database testing?  
15. What are the shortest testing techniques to test a page?  
(We have to tell BVA, ECP, Response time).  
16. What is scrum & sprint?

**Why should we hire you?**

I think you should hire me because, I realize that there are likely other candidates who also have the ability to do this job. Yet I bring an additional quality that makes me the best person for the job—my passion for excellence. I am passionately committed to producing truly world class results and I am a fast Learner, pay close attention for details, and have a willingness & readiness to learn and help

others. I’m willing to Learn and Perform and work co-operatively with others in order to achieve a common goal. I always aim at delivering quality work within timelines and I believe in adding value to the organisation and I take full accountability for my deliverables. I have the ability to accept change and adapt to my approach to maintain effectiveness within a variety of situations. I am willing to bring personal value which include honesty, respect, responsibility and a strong work ethic.

**Reason: Strong Technical Skills and Education:**

**I believe my strong technical experience and education will make me a very competitive candidate for a suitable position in your company.** Added to my diligence in paying close attention to detail,

## Strengths

1. Hard worker
2. Determined
3. Always Helpful
4. Flexible
5. Well Adaptable
6. Enthusiastic
7. Functional
8. Honest
9. Complete tasks on time
10. Self-Motivated
11. Ready to help others

# Weakness:

Short Tempered

Sentimental

I am a hard-working self-starter who works equally well in a team environment or individually. I am constantly searching for new experiences and new sources of knowledge, and often seek ways to utilize my creativity passionately. Furthermore, I strive for continued excellence.

“What is your greatest weakness” Never answer this question in a negative way. You could maybe say that you tend to work very late spend less time with your family. However, you are working towards leaving a little earlier to spend more time with them. That way, it is a weakness, but you are showing that there is a positive outcome.

# Why should we hire you

* I do the work and deliver exceptional results
* I am very much passionate about my job.
* I can work under stress and in a multi-cultural environment.
* I do work with the team working towards a common goal
* In my own perspective point of view, a new employer is someone that can give fresh ideas that can improve to the company which I think every company needed so I think your company deserves more interesting and new ideas and I think I am the best person that suit to it.

# What are Joins?

* **INNER JOIN**: Returns all rows when there is at least one match in BOTH tables
* **LEFT JOIN**: Return all rows from the left table, and the matched rows from the right table
* **RIGHT JOIN**: Return all rows from the right table, and the matched rows from the left table
* **FULL JOIN**: Return all rows when there is a match in ONE of the tables

### Restful Services:

### RESTful Web Services are REST architecture based web services. In REST Architecture everything is a resource. RESTful web services are light weight, highly scalable and maintainable and are very commonly used to create APIs for web based applications.

### Soap and Rest Web Services

|  |  |  |
| --- | --- | --- |
| **No.** | **SOAP** | **REST** |
| 1) | SOAP is a **protocol**. | REST is an **architectural style**. |
| 2) | SOAP stands for **Simple Object Access Protocol**. | REST stands for **Representational State Transfer**. |
| 3) | SOAP **can't use REST** because it is a protocol. | REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP. |
| 4) | SOAP **uses services interfaces to expose the business logic**. | REST **uses URI (Uniform Resource locator) to expose business logic**. |
| 5) | SOAP **defines standards**to be strictly followed. | REST does not define too much standards like SOAP. |
| 6) | SOAP **permits XML** data format only. | REST **permits different** data format such as Plain text, HTML, XML, JSON etc. |
|  |  | REST web services call can be cached to improve performance |

### Testing Rest Services

Testing a RESTful Web service includes the following checks:

1. That URL addresses are constituted correctly based on the service deployment end-point and the method annotations.
2. That the generated server requests call the corresponding methods.
3. That the methods return acceptable data.

### Stack Memory vs Heap Memory

### Stack is allocated for saving primitive data types

### Heap memory is the memory allocated for Object

### Java Stack memory is used for execution of a thread. They contain method specific values that are short-lived and references to other objects in the heap that are getting referred from the method.

### Stack memory is always referenced in LIFO (Last-In-First-Out) order. Whenever a method is invoked, a new block is created in the stack memory for the method to hold local primitive values and reference to other objects in the method. As soon as method ends, the block becomes unused and become available for next method.

### Stack memory size is very less compared to Heap memory.

### What is spring?

It is a lightweight, loosely coupled and integrated framework for developing enterprise applications in java.

### What is spring bean?

### A bean definition can contain a lot of configuration information, including constructor arguments, property values, and container-specific information such as initialization method, static factory method name, and so on.

### Type of DI

|  |
| --- |
| **Dependency Injection Type & Description** |
| 1 | [**Constructor-based dependency injection**](http://www.tutorialspoint.com/spring/constructor_based_dependency_injection.htm)  Constructor-based DI is accomplished when the container invokes a class constructor with a number of arguments, each representing a dependency on other class. |
| 2 | [**Setter-based dependency injection**](http://www.tutorialspoint.com/spring/setter_based_dependency_injection.htm)  Setter-based DI is accomplished by the container calling setter methods on your beans after invoking a no-argument constructor or no-argument static factory method to instantiate your bean. |

2) What are the advantages of spring framework?

1. ~~Predefined Templates~~
2. Loose Coupling
3. Easy to test
4. Lightweight
5. Fast Development
6. Powerful Abstraction
7. ~~Declarative support~~

### 3) What are the modules of spring framework?

* **Spring Core Container**

The Spring Core container contains **core**, **beans**, **context** and **expression language (EL)** modules.

***Core*** *and* ***Beans***modules provide IOC and Dependency Injection features.

**Inversion of control** – Done through dependency Injection

We will be configuration Beans in XML files and get the bean from XML file and injected with all its dependencies

**ORM – Object relational Mapping**

Mapping POJO objects with Relational Database

Bean will talk to POJO and pojo communicates to RDB

OXM – Object to XML mapping (Marshalling and UnMarshalling)

**Dependency Injection**

**IOC:**

**IOC (**Inversion of Control)**:**  is a programming technique in which object coupling is bound at run time by an assembler object and is typically not known at compile time using static analysis.

**DI** is the process of providing the dependencies of an object at run time by using setter injection or constructor injection.

# **Autowiring in Spring**

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.

## **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 calls setter 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. |

### Jmeter Interview Questions

### How to create a Jmeter Test Plan

### Create a Test Plan

### Create a Thread Group

### Configure the no of threads

### Ramp up period

### Loop Count

### Under Thread Group

### Add Samplers like

### HTTP Request,

### JDBC Request,

### LDAP request

### FTP Request

### SMTP Sampler

### JUNIT Request

### JAVA Request

### Under Thread Group or Test Plan add listeners for capturing the result

### Like Summary report

### View Results in table

### View Results in tree.

### Performance Parameters

### No of samples

### Start time

### Load Time

### Sample time

### Throughput

### Latency

### Status

### Errors

### Response Code

### Delete vs Truncate vs Drop

### Delete will the rows based on some conditional where class or else deletes the entire table followed by Commit or Rollback and does not clear the table space

### Truncate will delete and clears the Table space

### Drop will remove the table and all the contents.

### Autimation metrics

Total Automation execution

**Automation Progress :**

AA No. of test cases automated

AP (%) = -------- = (-------------------------------------- )

ATC No. of test cases automatable

**Percent of Automated Test Coverage:**

AC Automation coverage

PTC (%) = ------- = (------------------------------- )

C Total coverage

**Percentage Automatable or Automation Index :**

ATC No. of test cases automatable

PA (%) = -------- = (----------------------------------- )

TC Total no. of test cases

Comparison Meztrics:

* Test execution time
*  Test analysis time (expect to be low)
* Number of times test is to be executed during next 12 months
*  Requirement(s) covered by each test
*  Defects reported
* Total number of test cases
*  Total number of regression test cases
*  Regression test cases executed (on average)
* during regression run
*  Regression test cases NOT executed (on average) during regression run
*  Defects reported (on average) during regression run
*  Defects rejected (due to tester error)
*  Regression testing time

ROI:

ROI is one of the measures to Evalute the investments made

ROI= GainfromInvesment - CostofInvestment/CostofInvestment

**Capacity testing:**Capacity testing is complementary to load testing and it determines your server’s ultimate failure point

**Volume testing:** This is done to test how the system handles  
when there is a need for huge volumes of data.

**MultiThreading**

### What is the difference between preemptive scheduling and time slicing?

Under preemptive scheduling, the highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence. Under time slicing, a task executes for a predefined slice of time and then reenters the pool of ready tasks. The scheduler then determines which task should execute next, based on priority and other factors.