Interview questions

What is view?(SQL)

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

A view is created with the CREATE VIEW statement

Example:

CREATE VIEW [Brazil Customers] AS

SELECT CustomerName, ContactName

FROM Customers

WHERE Country = 'Brazil';

Output:

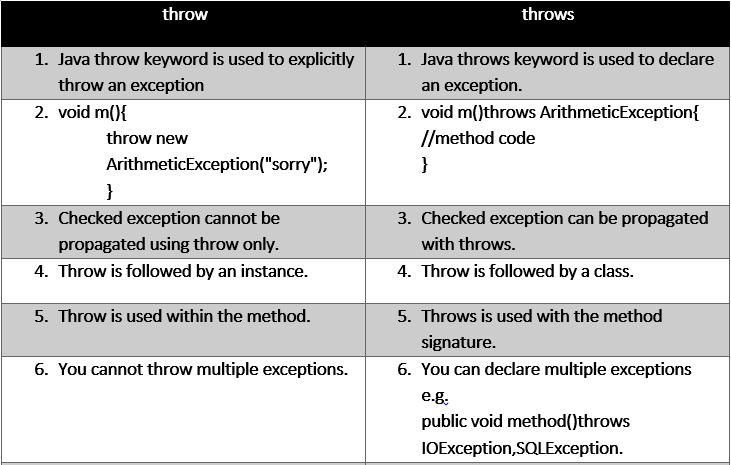
#### Views:

|  |  |
| --- | --- |
| **Name of View** | **Records** |
| Brazil Customers | 9 |

What is View?(MVC)

View Model is a model class that can hold only those properties that is required for a view. It can also contains properties from more than one entities (tables) of the database. As the name suggests, this model is created specific to the View requirements

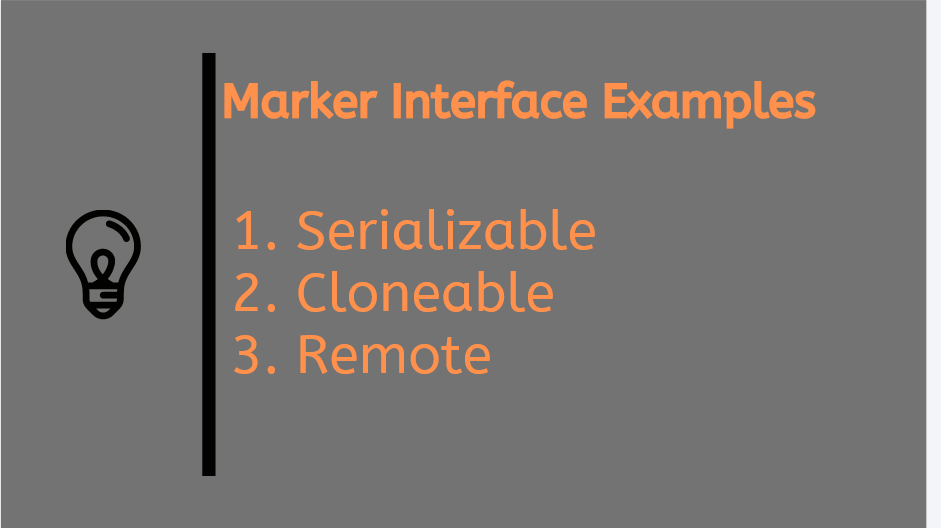
What is difference between throw and throws



**What is marker interface?**

A marker interface is an [interface](https://www.baeldung.com/java-interfaces) 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.



Java has many built-in marker interfaces, such as Serializable, Cloneable, and Remote

Example

**1.Serializable**

The **Serializable**interface is present in **java.io** package. It is a [marker interface](https://www.geeksforgeeks.org/marker-interface-java/). A Marker Interface does not have any methods and fields. Thus classes implementing it do not have to implement any methods. Classes implement it if they want their instances to be Serialized or Deserialized. Serialization is a mechanism of converting the state of an object into a byte stream. Serialization is done using [ObjectOutputStream](https://www.geeksforgeeks.org/java-io-objectoutputstream-class-java-set-1/). Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist the object. Deserialization is done using [ObjectInputStream](https://www.geeksforgeeks.org/java-io-objectinputstream-class-java-set-2/). Thus it can be used to make an eligible for saving its state into a file.

**2.Cloneable**

The **Java.lang.Cloneable**interface is a [marker interface](https://www.geeksforgeeks.org/marker-interface-java/). It was introduced in JDK 1.0. There is a method [clone()](https://www.geeksforgeeks.org/clone-method-in-java-2/) in the Object class. **Cloneable**interface is implemented by a class to make [Object.clone()](https://www.geeksforgeeks.org/clone-method-in-java-2/) method valid thereby making field-for-field copy. This interface allows the implementing class to have its objects to be cloned instead of using a **new**operator

What is index?

The CREATE INDEX statement is used to create indexes in tables.

Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries

CREATE INDEX idx\_lastname  
ON Persons (LastName)

**How many methods present in String?**

## All String Methods

The String class has a set of built-in methods that you can use on strings.

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Return Type** |
| [charAt()](https://www.w3schools.com/java/ref_string_charat.asp) | Returns the character at the specified index (position) | char |
| [codePointAt()](https://www.w3schools.com/java/ref_string_codepointat.asp) | Returns the Unicode of the character at the specified index | int |
| [codePointBefore()](https://www.w3schools.com/java/ref_string_codepointbefore.asp) | Returns the Unicode of the character before the specified index | int |
| [codePointCount()](https://www.w3schools.com/java/ref_string_codepointcount.asp) | Returns the number of Unicode values found in a string. | int |
| [compareTo()](https://www.w3schools.com/java/ref_string_compareto.asp) | Compares two strings lexicographically | int |
| [compareToIgnoreCase()](https://www.w3schools.com/java/ref_string_comparetoignorecase.asp) | Compares two strings lexicographically, ignoring case differences | int |
| [concat()](https://www.w3schools.com/java/ref_string_concat.asp) | Appends a string to the end of another string | String |
| [contains()](https://www.w3schools.com/java/ref_string_contains.asp) | Checks whether a string contains a sequence of characters | boolean |
| [contentEquals()](https://www.w3schools.com/java/ref_string_contentequals.asp) | Checks whether a string contains the exact same sequence of characters of the specified CharSequence or StringBuffer | boolean |
| [copyValueOf()](https://www.w3schools.com/java/ref_string_copyvalueof.asp) | Returns a String that represents the characters of the character array | String |
| [endsWith()](https://www.w3schools.com/java/ref_string_endswith.asp) | Checks whether a string ends with the specified character(s) | boolean |
| [equals()](https://www.w3schools.com/java/ref_string_equals.asp) | Compares two strings. Returns true if the strings are equal, and false if not | boolean |
| [equalsIgnoreCase()](https://www.w3schools.com/java/ref_string_equalsignorecase.asp) | Compares two strings, ignoring case considerations | boolean |
| format() | Returns a formatted string using the specified locale, format string, and arguments | String |
| getBytes() | Encodes this String into a sequence of bytes using the named charset, storing the result into a new byte array | byte[] |
| getChars() | Copies characters from a string to an array of chars | void |
| [hashCode()](https://www.w3schools.com/java/ref_string_hashcode.asp) | Returns the hash code of a string | int |
| [indexOf()](https://www.w3schools.com/java/ref_string_indexof.asp) | Returns the position of the first found occurrence of specified characters in a string | int |
| intern() | Returns the canonical representation for the string object | String |
| [isEmpty()](https://www.w3schools.com/java/ref_string_isempty.asp) | Checks whether a string is empty or not | boolean |
| [lastIndexOf()](https://www.w3schools.com/java/ref_string_lastindexof.asp) | Returns the position of the last found occurrence of specified characters in a string | int |
| [length()](https://www.w3schools.com/java/ref_string_length.asp) | Returns the length of a specified string | int |
| matches() | Searches a string for a match against a regular expression, and returns the matches | boolean |
| offsetByCodePoints() | Returns the index within this String that is offset from the given index by codePointOffset code points | int |
| regionMatches() | Tests if two string regions are equal | boolean |
| [replace()](https://www.w3schools.com/java/ref_string_replace.asp) | Searches a string for a specified value, and returns a new string where the specified values are replaced | String |
| replaceFirst() | Replaces the first occurrence of a substring that matches the given regular expression with the given replacement | String |
| replaceAll() | Replaces each substring of this string that matches the given regular expression with the given replacement | String |
| split() | Splits a string into an array of substrings | String[] |
| [startsWith()](https://www.w3schools.com/java/ref_string_startswith.asp) | Checks whether a string starts with specified characters | boolean |
| subSequence() | Returns a new character sequence that is a subsequence of this sequence | CharSequence |
| substring() | Returns a new string which is the substring of a specified string | String |
| toCharArray() | Converts this string to a new character array | char[] |
| [toLowerCase()](https://www.w3schools.com/java/ref_string_tolowercase.asp) | Converts a string to lower case letters | String |
| toString() | Returns the value of a String object | String |
| [toUpperCase()](https://www.w3schools.com/java/ref_string_touppercase.asp) | Converts a string to upper case letters | String |
| [trim()](https://www.w3schools.com/java/ref_string_trim.asp) | Removes whitespace from both ends of a string | String |
| valueOf() | Returns the string representation of the specified value | String |

**Diff between treemap and hashmap**



What is copy constructor ?

A copy constructor in a Java class is a [constructor](https://www.baeldung.com/java-constructors) that**creates an object using another object of the same Java class**.

That's helpful when we want to copy a complex object that has several fields, or when we want to make a [deep copy](https://www.baeldung.com/java-deep-copy) of an existing object

Exapmple

## How to Create a Copy Constructor

To create a copy constructor, we can first declare a constructor that takes an object of the same type as a parameter:

**public** **class** **Employee** {

**private** **int** id;

**private** String name;

**public** **Employee**(Employee employee) {

}

}

Then, we copy each field of the input object into the new instance:

**public** **class** **Employee** {

**private** **int** id;

**private** String name;

**public** **Employee**(Employee employee) {

this.id = employee.id;

this.name = employee.name;

}

}

What we have here is a shallow copy, which is fine since all of our fields – an int and a String in this case – are either [primitive types](https://www.baeldung.com/java-primitives) or [immutable types](https://www.baeldung.com/java-immutable-object)

**What is default constructor?**

## 3. Java Default Constructor

If we do not create any constructor, the Java compiler automatically create a no-arg constructor during the execution of the program. This constructor is called default constructor.

### Example 5: Default Constructor

class Main {

int a;

boolean b;

public static void main(String[] args) {

// A default constructor is called

Main obj = new Main();

System.out.println("Default Value:");

System.out.println("a = " + obj.a);

System.out.println("b = " + obj.b);

}

}

What is Autoboxing and Autounboxing?

In Java, primitive data types are treated differently so do there comes the introduction of [wrapper classes](https://www.geeksforgeeks.org/wrapper-classes-java/) where two components play a role namely Autoboxing and Unboxing. [Autoboxing](https://www.geeksforgeeks.org/autoboxing-unboxing-java/) refers to the conversion of a primitive value into an object of the corresponding wrapper class is called autoboxing. For example, converting int to Integer class. The Java compiler applies autoboxing when a primitive value is:

* Passed as a parameter to a method that **expects an object** of the corresponding wrapper class.
* Assigned to a variable of the corresponding **wrapper class**.

**Unboxing**on the other hand refers to converting an object of a wrapper type to its corresponding primitive value. For example conversion of Integer to int. The Java compiler applies to unbox when an object of a wrapper class is:

* Passed as a parameter to a method that **expects a value** of the corresponding primitive type.
* Assigned to a variable of the corresponding **primitive type**.

| Primitive Type | Wrapper Class |
| --- | --- |
| boolean | Boolean |
| byte | Byte |
| char | Character |
| float | Float |
| int | Integer |
| long | Long |
| short | Short |
| double | Double |

The following table lists the primitive types and their corresponding wrapper classes, which are used by the Java compiler for autoboxing and unboxing. Now let us discuss a few advantages of autoboxing and unboxing in order to get why we are using it.

* Autoboxing and unboxing lets developers write cleaner code, making it easier to read.
* The technique lets us use primitive types and Wrapper class objects interchangeably and we do not need to perform any typecasting explicitly.

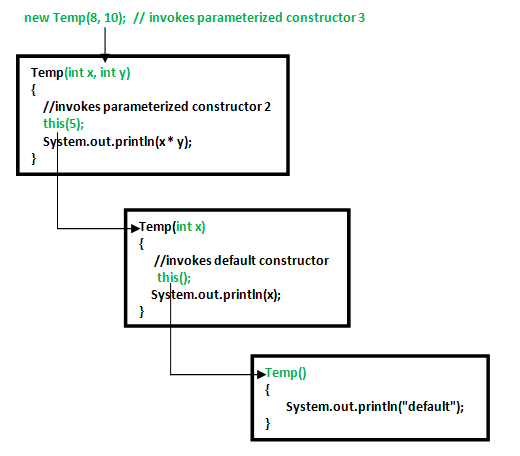
What is constructor chaining ?

Constructor chaining is the process of calling one constructor from another constructor with respect to current object.   
Constructor chaining can be done in two ways: 

* **Within same class**: It can be done using **this()** keyword for constructors in same class
* **From base class:**by using **super()** keyword to call constructor from the base class.

Constructor chaining occurs through **inheritance**. A sub class constructor’s task is to call super class’s constructor first. This ensures that creation of sub class’s object starts with the initialization of the data members of the super class. There could be any numbers of classes in inheritance chain. Every constructor calls up the chain till class at the top is reached.  
**Why do we need constructor chaining ?**   
This process is used when we want to perform multiple tasks in a single constructor rather than creating a code for each task in a single constructor we create a separate constructor for each task and make their chain which makes the program more readable. 

**Constructor Chaining within same class using this() keyword :**



What is Stream Api?

Introduced in Java 8, the Stream API is used to process collections of objects. A stream is a sequence of objects that supports various methods which can be pipelined to produce the desired result.  
The features of Java stream are –

* A stream is not a data structure instead it takes input from the Collections, Arrays or I/O channels.
* Streams don’t change the original data structure, they only provide the result as per the pipelined methods.
* Each intermediate operation is lazily executed and returns a stream as a result, hence various intermediate operations can be pipelined. Terminal operations mark the end of the stream and return the result.

Different Operations On Streams-  
**Intermediate Operations:**

1. **map:**The map method is used to returns a stream consisting of the results of applying the given function to the elements of this stream.  
   List number = Arrays.asList(2,3,4,5);  
   List square = number.stream().map(x->x\*x).collect(Collectors.toList());
2. **filter:** The filter method is used to select elements as per the Predicate passed as argument.  
   List names = Arrays.asList("Reflection","Collection","Stream");  
   List result = names.stream().filter(s->s.startsWith("S")).collect(Collectors.toList());
3. **sorted:** The sorted method is used to sort the stream.  
   List names = Arrays.asList("Reflection","Collection","Stream");  
   List result = names.stream().sorted().collect(Collectors.toList());

**Terminal Operations:**

1. **collect:** The collect method is used to return the result of the intermediate operations performed on the stream.  
   List number = Arrays.asList(2,3,4,5,3);  
   Set square = number.stream().map(x->x\*x).collect(Collectors.toSet());
2. **forEach:** The forEach method is used to iterate through every element of the stream.  
   List number = Arrays.asList(2,3,4,5);  
   number.stream().map(x->x\*x).forEach(y->System.out.println(y));
3. **reduce:** The reduce method is used to reduce the elements of a stream to a single value.  
   The reduce method takes a BinaryOperator as a parameter.

List number = Arrays.asList(2,3,4,5);  
int even = number.stream().filter(x->x%2==0).reduce(0,(ans,i)-> ans+i);

Here ans variable is assigned 0 as the initial value and i is added to it .

What is Lambda Expression?

Lambda expressions basically express instances of [functional interfaces](https://www.geeksforgeeks.org/functional-interfaces-java/) (An interface with single abstract method is called functional interface. An example is java.lang.Runnable). lambda expressions implement the only abstract function and therefore implement functional interfaces

lambda expressions are added in Java 8 and provide below functionalities.

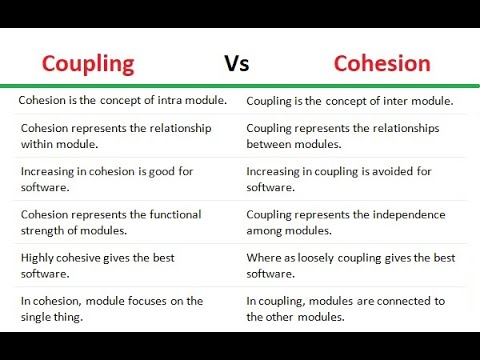
* Enable to treat functionality as a method argument, or code as data.
* A function that can be created without belonging to any class.
* A lambda expression can be passed around as if it was an object and executed on demand.

|  |
| --- |
| // Java program to demonstrate lambda expressions  // to implement a user defined functional interface.    // A sample functional interface (An interface with  // single abstract method  interface FuncInterface  {      // An abstract function      void abstractFun(int x);        // A non-abstract (or default) function      default void normalFun()      {         System.out.println("Hello");      }  }    class Test  {      public static void main(String args[])      {          // lambda expression to implement above          // functional interface. This interface          // by default implements abstractFun()          FuncInterface fobj = (int x)->System.out.println(2\*x);            // This calls above lambda expression and prints 10.          fobj.abstractFun(5);      }  } |

Output:

10

What is cohesion and Coupling?



**Cohesion Coupling**

**Spring, Spring Mvc ,SpringBoot and Spring Ioc**

What is spring boot

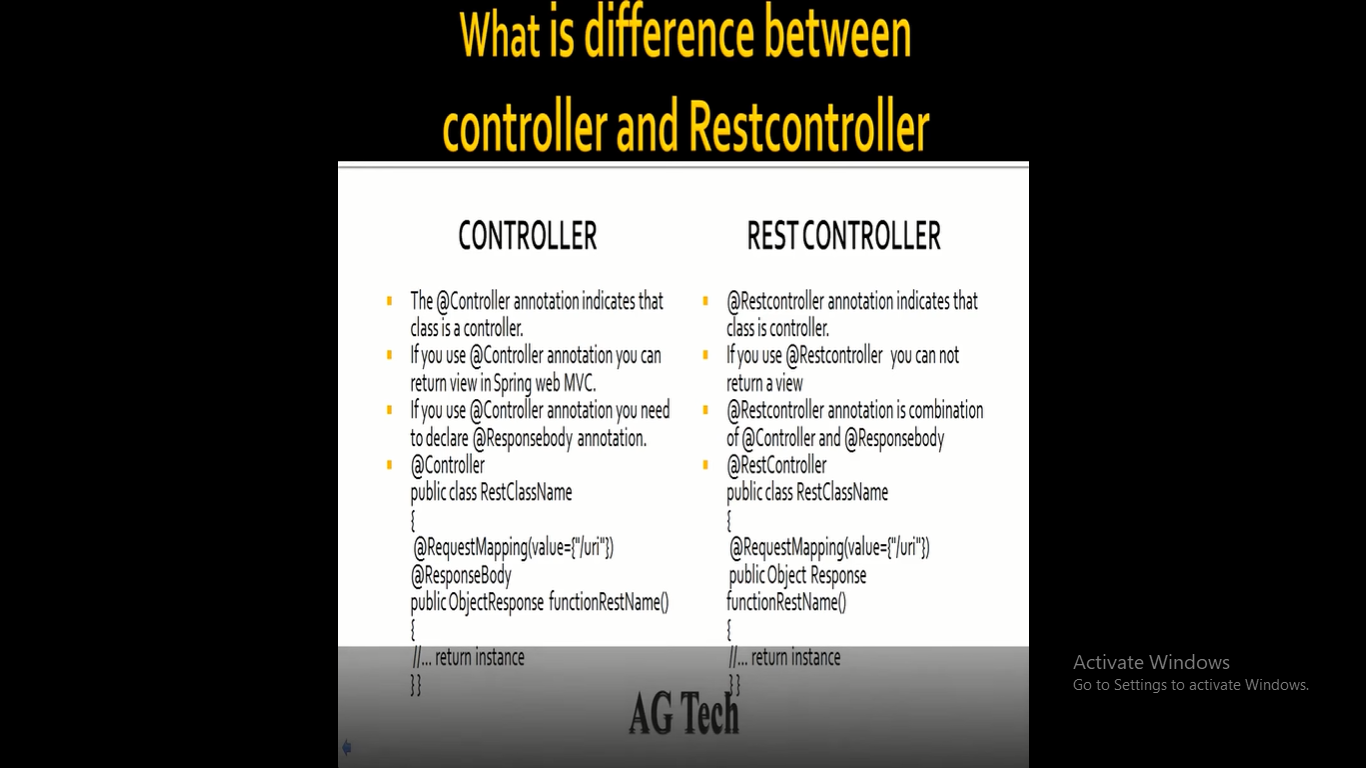
Spring Boot is an open source, microservice-based Java web framework. The  Spring Boot framework creates a fully production-ready environment that is completely configurable using its prebuilt code within its codebase. The microservice architecture provides developers with a fully enclosed application, including embedded application servers.

Advantages of Spring boot

* Spring Framework can be employed on all architectural layers used in the development of web applications;
* Uses the very lightweight POJO model when writing classes;
* Allows you to freely link modules and easily test them;
* Supports declarative programming;
* Eliminates the need to independently create factory and singleton classes;
* Supports various configuration methods;
* Provides middleware-level service.

**Diff bet @RestController and @Controller annotations**

( Representational State Transfer)



**Autowiring and its Types**

* **no**: It’s the default autowiring mode. It means no autowiring.
* **byName**: The byName mode injects the object dependency according to name of the bean. In such a case, the property and bean name should be the same. It internally calls the setter method.
* **byType**: The byType mode injects the object dependency according to type. So it can have a different property and bean name. It internally calls the setter method.
* **constructor**: The constructor mode injects the dependency by calling the constructor of the class. It calls the constructor having a large number of parameters.
* **autodetect**: In this mode, Spring first tries to autowire by the constructor. If this fails, it tries to autowire by using byType.

**Spring bean life cycle**

**init()**

**destroy()**

**2. By using interface**

**InitializingBean**

**afterPropertiesSet()**

**DisposableBean**

**destroy()**

**3. Using Annotation**

**@PostConstruct**

**@PreDestroy**

Steps to create rest API

Follow the below-mentioned steps to build a Spring Boot REST API using Java.

* [Step 1: Initializing a Spring Boot Project](https://hevodata.com/learn/spring-boot-rest-api/#s1)
* [Step 2: Connecting Spring Boot to the Database](https://hevodata.com/learn/spring-boot-rest-api/#s2)
* [Step 3: Creating a User Model](https://hevodata.com/learn/spring-boot-rest-api/#s3)
* [Step 4: Creating Repository Classes](https://hevodata.com/learn/spring-boot-rest-api/#s4)
* [Step 5: Creating a Controller](https://hevodata.com/learn/spring-boot-rest-api/#s5)
* [Step 6: Compile, Build and Run](https://hevodata.com/learn/spring-boot-rest-api/#s6)
* [Step 7: Testing the Spring Boot REST APIs](https://hevodata.com/learn/spring-boot-rest-api/#s7)

**Get, post, put, delete and patch method**

GET

The HTTP GET method is used to \*\*read\*\* (or retrieve) a representation of a resource. In the “happy” (or non-error) path, GET returns a representation in XML or JSON and an HTTP response code of 200 (OK). In an error case, it most often returns a 404 (NOT FOUND) or 400 (BAD REQUEST).

According to the design of the HTTP specification, GET (along with HEAD) requests are used only to read data and not change it. Therefore, when used this way, they are considered safe. That is, they can be called without risk of data modification or corruption—calling it once has the same effect as calling it 10 times, or none at all. Additionally, GET (and HEAD) is idempotent, which means that making multiple identical requests ends up having the same result as a single request.

Do not expose unsafe operations via GET—it should never modify any resources on the server

**POST**

The POST verb is most-often utilized to \*\*create\*\* new resources. In particular, it's used to create subordinate resources. That is, subordinate to some other (e.g. parent) resource. In other words, when creating a new resource, POST to the parent and the service takes care of associating the new resource with the parent, assigning an ID (new resource URI), etc.

On successful creation, return HTTP status 201, returning a Location header with a link to the newly-created resource with the 201 HTTP status.

POST is neither safe nor idempotent. It is therefore recommended for non-idempotent resource requests. Making two identical POST requests will most-likely result in two resources containing the same information.

**PUT** is most-often utilized for \*\*update\*\* capabilities, PUT-ing to a known resource URI with the request body containing the newly-updated representation of the original resource.

However, PUT can also be used to create a resource in the case where the resource ID is chosen by the client instead of by the server. In other words, if the PUT is to a URI that contains the value of a non-existent resource ID. Again, the request body contains a resource representation. Many feel this is convoluted and confusing. Consequently, this method of creation should be used sparingly, if at all.

Alternatively, use POST to create new resources and provide the client-defined ID in the body representation—presumably to a URI that doesn't include the ID of the resource (see POST below).

On successful update, return 200 (or 204 if not returning any content in the body) from a PUT. If using PUT for create, return HTTP status 201 on successful creation. A body in the response is optional—providing one consumes more bandwidth. It is not necessary to return a link via a Location header in the creation case since the client already set the resource ID.

PUT is not a safe operation, in that it modifies (or creates) state on the server, but it is idempotent. In other words, if you create or update a resource using PUT and then make that same call again, the resource is still there and still has the same state as it did with the first call.

If, for instance, calling PUT on a resource increments a counter within the resource, the call is no longer idempotent. Sometimes that happens and it may be enough to document that the call is not idempotent. However, it's recommended to keep PUT requests idempotent. It is strongly recommended to use POST for non-idempotent requests.

**PATCH** is used for \*\*modify\*\* capabilities. The PATCH request only needs to contain the changes to the resource, not the complete resource.

This resembles PUT, but the body contains a set of instructions describing how a resource currently residing on the server should be modified to produce a new version. This means that the PATCH body should not just be a modified part of the resource, but in some kind of patch language like JSON Patch or XML Patch.

PATCH is neither safe nor idempotent. However, a PATCH request can be issued in such a way as to be idempotent, which also helps prevent bad outcomes from collisions between two PATCH requests on the same resource in a similar time frame. Collisions from multiple PATCH requests may be more dangerous than PUT collisions because some patch formats need to operate from a known base-point or else they will corrupt the resource. Clients using this kind of patch application should use a conditional request such that the request will fail if the resource has been updated since the client last accessed the resource. For example, the client can use a strong ETag in an If-Match header on the PATCH request.

DELETE is pretty easy to understand. It is used to \*\*delete\*\* a resource identified by a URI.

On successful deletion, return HTTP status 200 (OK) along with a response body, perhaps the representation of the deleted item (often demands too much bandwidth), or a wrapped response (see Return Values below). Either that or return HTTP status 204 (NO CONTENT) with no response body. In other words, a 204 status with no body, or the JSEND-style response and HTTP status 200 are the recommended responses.

HTTP-spec-wise, DELETE operations are idempotent. If you DELETE a resource, it's removed. Repeatedly calling DELETE on that resource ends up the same: the resource is gone. If calling DELETE say, decrements a counter (within the resource), the DELETE call is no longer idempotent. As mentioned previously, usage statistics and measurements may be updated while still considering the service idempotent as long as no resource data is changed. Using POST for non-idempotent resource requests is recommended.

There is a caveat about DELETE idempotence, however. Calling DELETE on a resource a second time will often return a 404 (NOT FOUND) since it was already removed and therefore is no longer findable. This, by some opinions, makes DELETE operations no longer idempotent, however, the end-state of the resource is the same. Returning a 404 is acceptable and communicates accurately the status of the call.

Diff bet put and patch



**What is IOC**

Spring IoC (Inversion of Control) Container is the core of [Spring Framework](https://www.geeksforgeeks.org/introduction-to-spring-framework/). It creates the objects, configures and assembles their dependencies, manages their entire life cycle. The Container uses Dependency Injection(DI) to manage the components that make up the application

**There are 2 types of IoC containers:**

* [BeanFactory](https://www.geeksforgeeks.org/spring-beanfactory/)
* [ApplicationContext](https://www.geeksforgeeks.org/spring-applicationcontext/)

**What is Dependency Injection and its type(setter, constructor and field)**

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.

**@primary and @qualifier annotations:**

## Spring @Qualifier annotation – select bean

The **@Autowired**annotation does a great job of resolving the dependency. But it throws NoUniqueBeanDefinitionException when it can not resolve a specific bean due to ambiguity. This exception occurs when there is more than one bean of the same type is available in the container.

The @Primary annotation does a great job of giving a higher preference to a bean when there are multiple beans of the same types registered in the container. But it does not allow you to select a specific bean.

If you do not specify the @Primary annotation, there will be ambiguity as there are 2 beans of the same type

Bean ambiguity – NoUniqueBeanDefinitionException Use

@Primary annotation with @Bean

Exception handling in spring

Two types

1 .controlller

@ControllerAdvice

@ExceptionHandler

2.Global

@RestControllerAdvice

@ExceptionHandler

Spring MVC annotations

@ComponentScan

@Component

@Controller

@Service

@Repository

@RestControllerAdvice

@ExceptionHandler

@Bean

@Entity

@Table

@GeneratedValue

@Embeddable

@OneToOne

@OneToMany

@ManyToOne

@ManyToMany

@Autowired

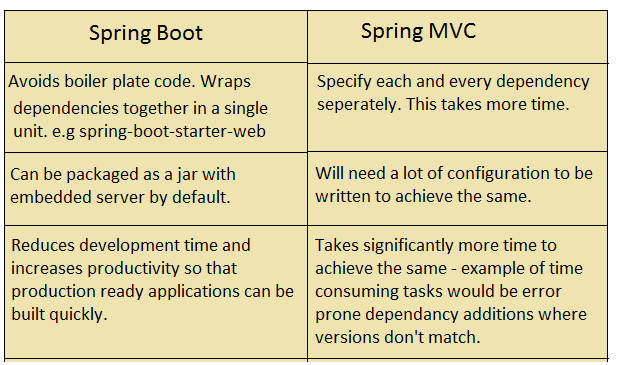
@Data

@SpringBootApplication

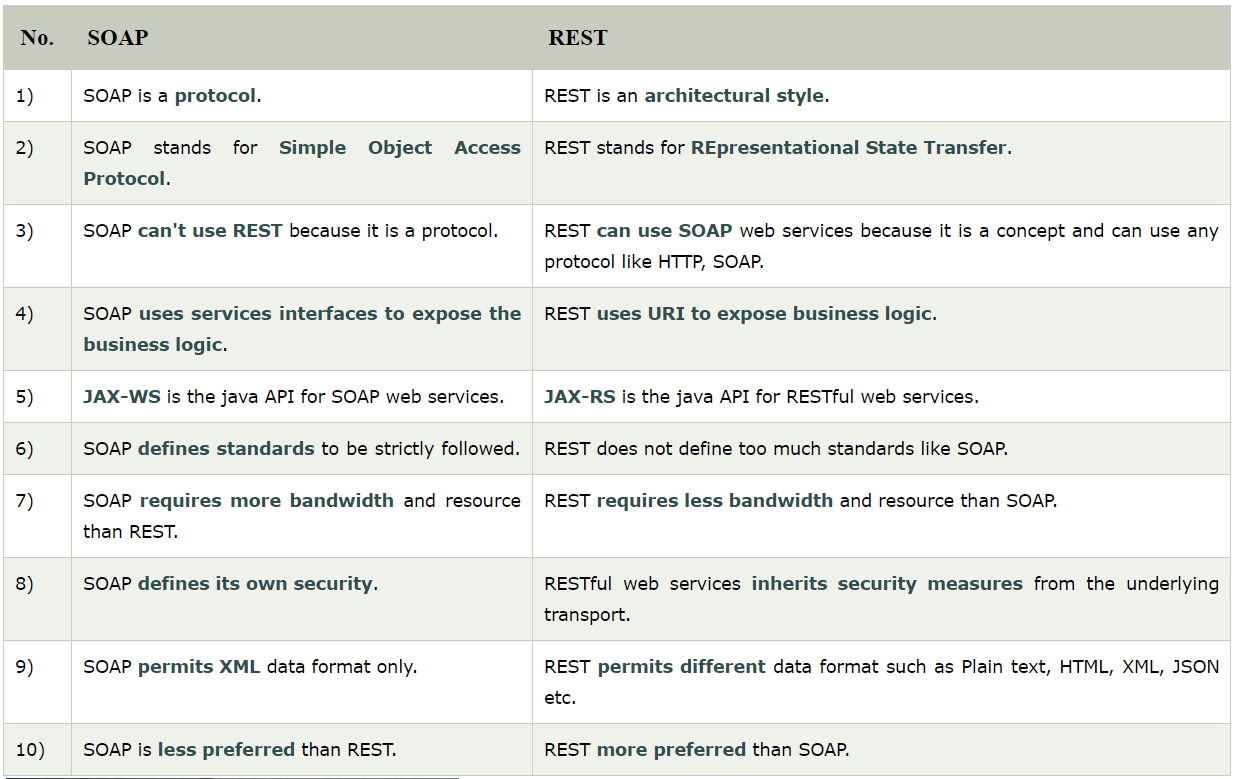
@AutoEnableConfigurations

@configuration

Diff bet spring MVC and Spring Boot



**What is SOAP and RESt diff bet them?**



**What is spring security and Advantages**

**Authorization** is the process to allow authority to perform actions in the application.

We can apply authorization to authorize web request, methods and access to individual domain.

Technologies that support Spring Security Integration

Spring Security framework supports wide range of authentication models. These models either provided by third parties or framework itself. Spring Security supports integration with all of these technologies.

* HTTP BASIC authentication headers
* HTTP Digest authentication headers
* HTTP X.509 client certificate exchange
* LDAP (Lighweight Directory Access Protocol)
* Form-based authentication
* OpenID authentication
* Automatic remember-me authentication
* Kerberos
* JOSSO (Java Open Source Single Sign-On)
* AppFuse
* AndroMDA
* Mule ESB
* DWR(Direct Web Request)

The beauty of this framework is its flexible authentication nature to integrate with any software solution. Sometimes, developers want to integrate it with a legacy system that does not follow any security standard, there Spring Security works nicely.

## Advantages

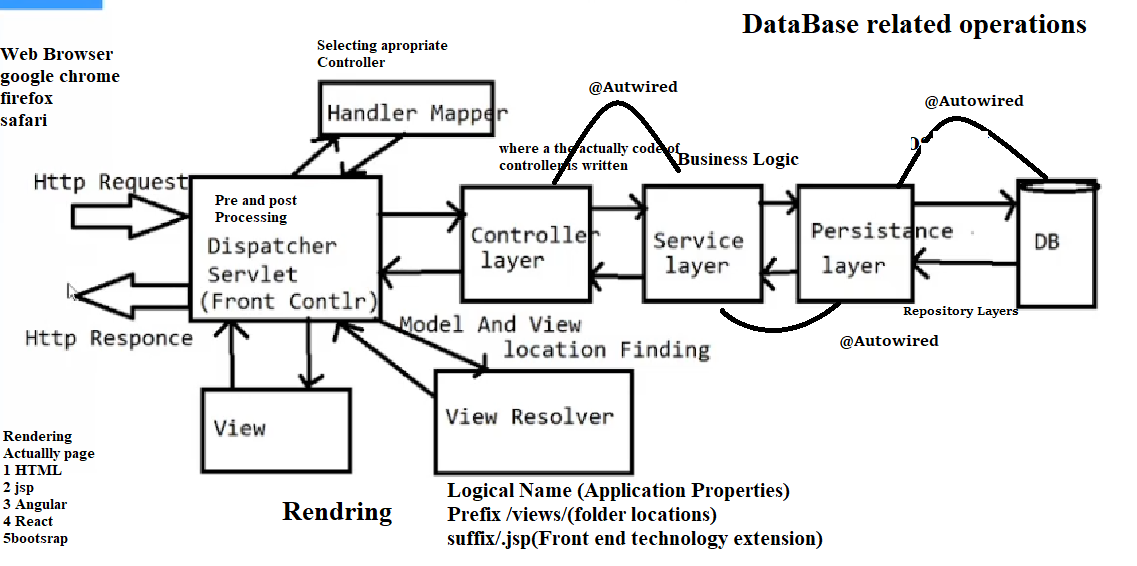
Spring Security has numerous advantages. Some of that are given below.

* Comprehensive support for authentication and authorization.
* Protection against common tasks
* Servlet API integration
* Integration with Spring MVC
* Portability
* CSRF protection
* Java Configuration support

What is webservices or benefits of web services

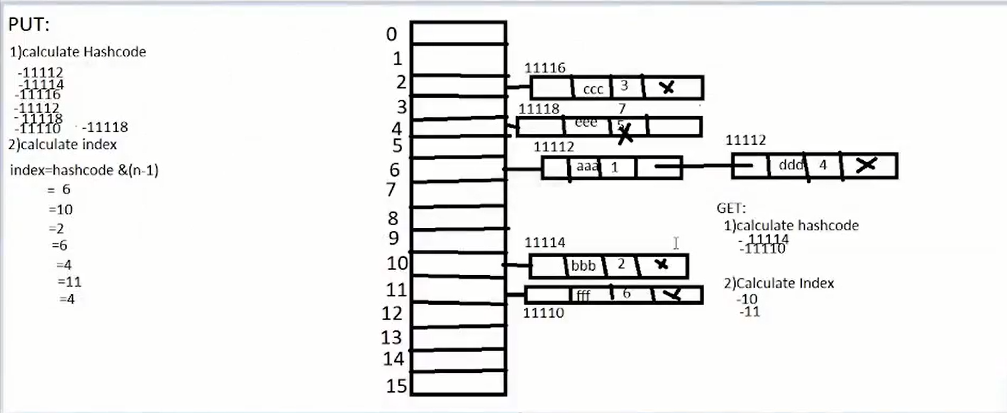


**MVC Architecture :**

****

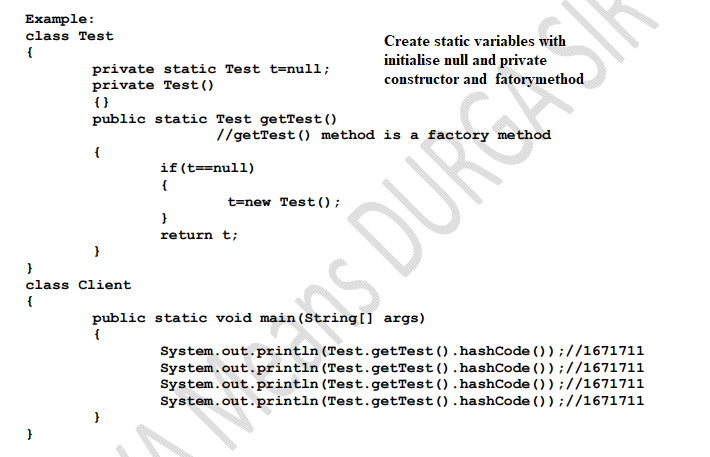
**Internal Hashmap Working**

* HashMap is a part of the Java collection framework. It uses a technique called Hashing. It implements the map interface. It stores the data in the pair of Key and Value. HashMap contains an array of the nodes, and the node is represented as a class. It uses an array and LinkedList data structure internally for storing Key and Value. There are four fields in HashMap.
* Before understanding the internal working of HashMap, you must be aware of hashCode() and equals() method.
* equals(): It chec ks the equality of two objects. It compares the Key, whether they are equal or not. It is a method of the Object class. It can be overridden. If you override the equals() method, then it is mandatory to override the hashCode() method.
* hashCode(): This is the method of the object class. It returns the memory reference of the object in integer form. The value received from the method is used as the bucket number. The bucket number is the address of the element inside the map. Hash code of null Key is 0.
* Buckets: Array of the node is called buckets. Each node has a data structure like a LinkedList. More than one node can share the same bucket. It may be different in capacity.

****

****

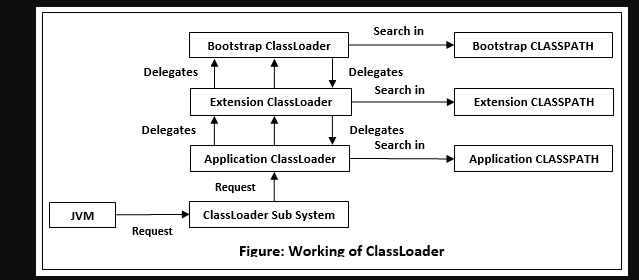
**How to create singleton class?**

****

**What is class loader in java?**

It follows delegations principles :

If bootstrap class loader, extension class loader and Applications or System class loader return null then jvm throws **classnotnotfoundexception**



**Json to java?**

## Mapping between JSON and Java entities

JSON.simple maps entities from the left side to the right side while decoding or parsing, and maps entities from the right to the left while encoding.

|  |  |
| --- | --- |
| **JSON** | **Java** |
| string | java.lang.String |
| number | java.lang.Number |
| true|false | java.lang.Boolean |
| null | Null |
| array | java.util.List |
| object | java.util.Map |

On decoding, the default concrete class of *java.util.List* is *org.json.simple.JSONArray* and the default concrete class of *java.util.Map* is *org.json.simple.JSONObject*.

## Encoding JSON in Java

Following is a simple example to encode a JSON object using Java JSONObject which is a subclass of java.util.HashMap. No ordering is provided. If you need the strict ordering of elements, use JSONValue.toJSONString ( map ) method with ordered map implementation such as java.util.LinkedHashMap.

**Advantages of Spring MVC and its features?**

* Lightweight. ...
* Inversion Of Control (IOC) ...
* Aspect Oriented Programming (AOP) ...
* Container. ...
* MVC Framework. ...
* Transaction Management. ...
* JDBC Exception Handling. ...
* Solving difficulties of Enterprise application development

**Diff bet @component and @componentScan annotation**

**@Component Annotation**

@Component is a class-level annotation. It is used to denote a class as a Component. We can use @Component across the application to mark the beans as Spring’s managed components. A component is responsible for some operations. Spring framework provides three other specific annotations to be used when marking a class as a Component.

1. @Service
2. @Repository
3. @Controller

**1: @Service:**We specify a class with @Service to indicate that they’re holding the business logic. Besides being used in the service layer, there isn’t any other special use for this annotation. The utility classes can be marked as Service classes.

**2: @Repository:**We specify a class with @Repository to indicate that they’re dealing with **CRUD operations**, usually, it’s used with DAO (Data Access Object) or Repository implementations that deal with database tables.

**3: @Controller:**We specify a class with @Controller to indicate that they’re front controllers and responsible to handle user requests and return the appropriate response. It is mostly used with REST Web Services

**@ComponentScan :which is used along with the @Configuration annotation to specify the packages that we want to be scanned.** @ComponentScan without arguments tells Spring to scan the current package and all of its sub-packages

**Spring actuator?**

**Spring Boot Actuator** is a sub-project of the Spring Boot Framework. It includes a number of additional features that help us to monitor and manage the Spring Boot application. It contains the actuator endpoints (the place where the resources live). We can use **HTTP** and **JMX** endpoints to manage and monitor the Spring Boot application. If we want to get production-ready features in an application, we should use the S**pring Boot actuator.**

Spring Boot Actuator Features

There are **three** main features of Spring Boot Actuator:

* **Endpoints**
* **Metrics**
* **Audit**

**Endpoint:** The actuator endpoints allows us to monitor and interact with the application. Spring Boot provides a number of built-in endpoints. We can also create our own endpoint. We can enable and disable each endpoint individually. Most of the application choose **HTTP**, where the Id of the endpoint, along with the prefix of **/actuator,**is mapped to a URL.

For example, the **/health** endpoint provides the basic health information of an application. The actuator, by default, mapped it to **/actuator/health**.

How to validate Spring MVC?

|  |  |
| --- | --- |
| **Annotation** | **Description** |
| @NotNull | It determines that the value can't be null. |
| @Min | It determines that the number must be equal or greater than the specified value. |
| @Max | It determines that the number must be equal or less than the specified value. |
| @Size | It determines that the size must be equal to the specified value. |
| @Pattern | It determines that the sequence follows the specified regular expression. |

Examples:

@Size(min=1,message="required")

**private** String pass;

**public** String submitForm( @Valid @ModelAttribute("emp") Employee e, BindingResult br)

  {

}

**Profiles in spring boot?**

## Spring Boot profiles

The development process of an application has different stages; the typical ones are development, testing, and production. Spring Boot profiles group parts of the application configuration and make it be available only in certain environments.

A profile is a set of configuration settings. Spring Boot allows to define profile specific property files in the form of application-{profile}.properties. It automatically loads the properties in an application.properties file for all profiles, and the ones in profile-specific property files only for the specified profile. The keys in the profile-specific property override the ones in the master property fil

The @Profile annotation indicates that a component is eligible for registration when the specified profile or profiles are active. The default profile is called default; all the beans that do not have a profile set belong to this profile

## Spring Boot profiles example

In the following application, we have three profiles (local, dev, prod) and two profile-specific property files. We use the spring.profiles.active to set active profiles and SpringApplicationBuilder's profiles method to add new active profiles.

pom.xml

src

├── main

│   ├── java

│   │   └── com

│   │   └── zetcode

│   │   └── Application.java

│   └── resources

│   ├── application-dev.properties

│   ├── application-prod.properties

│   └── application.properties

└── test

└── java

This is the project structure.

Executor framework in java

 Java provides its own multi-threading framework called the Java Executor Framework.

Java executor framework ([java.util.concurrent.Executor](https://www.geeksforgeeks.org/java-util-concurrent-executor-interface-with-examples/)), released with the JDK 5 is used to run the Runnable objects without creating new threads every time and mostly re-using the already created threads. We all know that there are two ways to create a thread in java. If you want to read more about their comparison, read [how to create threads in Java.](https://www.geeksforgeeks.org/multithreading-in-java/)

The[java.util.concurrent.Executors](https://www.geeksforgeeks.org/java-util-concurrent-executor-interface-with-examples/) provide factory methods that are being used to create [ThreadPools](https://www.geeksforgeeks.org/thread-pools-java/) of worker threads. Thread pools overcome this issue by keeping the threads alive and reusing the threads. Any excess tasks flowing in that the threads in the pool can handle are held in a Queue. Once any of the threads get free, they pick up the next task from this queue. This task queue is essentially unbounded for the out-of-box executors provided by the JDK.

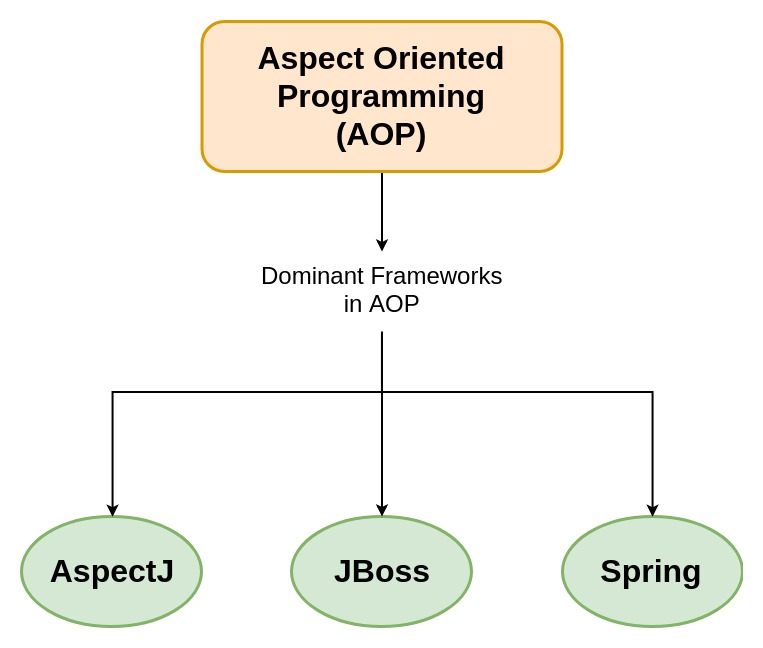
**Some types of Java Executors are listed below:**

1. SingleThreadExecutor
2. FixedThreadPool(n)+
3. CachedThreadPool
4. ScheduledExecutor

What is AOP

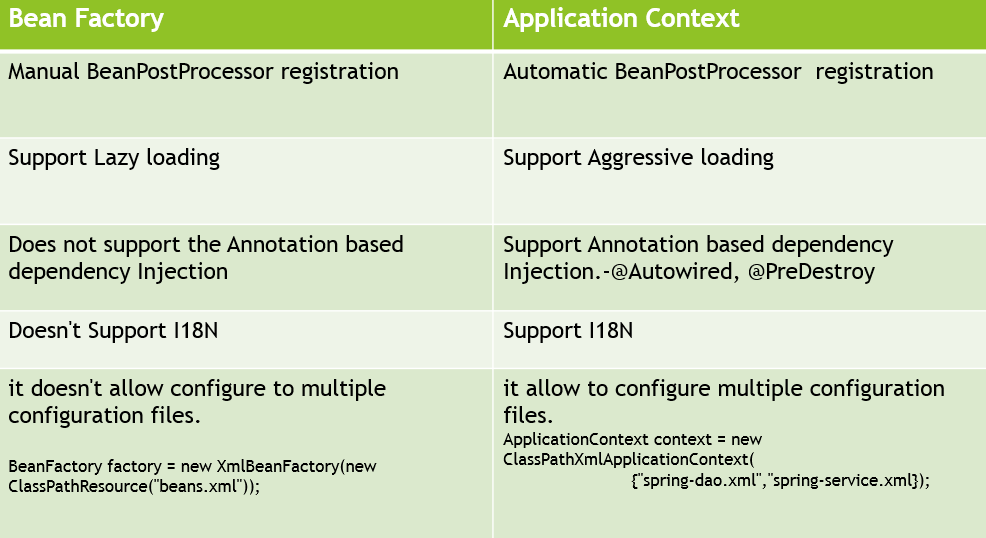
**Aspect oriented programming(AOP)** as the name suggests uses aspects in programming. It can be defined as the breaking of code into different modules, also known as [modularisation](https://www.geeksforgeeks.org/modular-approach-in-programming/), where the aspect is the key unit of modularity. Aspects enable the implementation of crosscutting concerns such as- transaction, logging not central to business logic without cluttering the code core to its functionality. It does so by adding additional behaviour that is the advice to the existing code. For example- Security is a crosscutting concern, in many methods in an application security rules can be applied, therefore repeating the code at every method, define the functionality in a common class and control were to apply that functionality in the whole application.

**Dominant Frameworks in AOP:**  
**AOP** includes programming methods and frameworks on which modularisation of code is

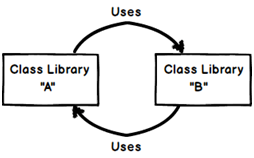
supported and implemented. Let’s have a look at the three **dominant frameworks in AOP**:  


* **AspectJ:** It is an extension for Java programming created at **PARC research centre**. It uses Java like syntax and included IDE integrations for displaying crosscutting structure. It has its own compiler and weaver, on using it enables the use of full AspectJ language.
* **JBoss:** It is an open source Java application server developed by JBoss, used for Java development.
* [Spring](https://www.geeksforgeeks.org/introduction-to-spring-framework/)**:** It uses XML based configuration for implementing AOP, also it uses annotations which are interpreted by using a library supplied by AspectJ for parsing and matching.

**Diff bet session factory and application context**



**Circular dependency**



How will configure multiple datasource in spring boot

Step1:

## Maven Setup(*spring-boot-starter-data-jpa*)

## DataSource Configurations

spring.jpa.generate-ddl=true

spring.datasource.url = jdbc:mysql://localhost:3306/db1

spring.datasource.username = [username]

spring.datasource.password = [password]

spring.datasource.driverClassName = com.mysql.jdbc.Driver

#second db2 ...

db2.datasource.url = jdbc:mysql://localhost:3306/db2

db2.datasource.username = [username]

db2.datasource.password = [password]

db2.datasource.driverClassName = com.mysql.jdbc.Driver

## 3. JPA Entities

Let’s define the following 2 JPA entities for our post.

1. Product
2. Custome

## 4. Package Structure

src/main/java

- com.javadevjournal

- product

- data

- repo

- config

- customer

- data

- repo

- config

## JPA Repositories

#### Product Repository

#### Customer Repository

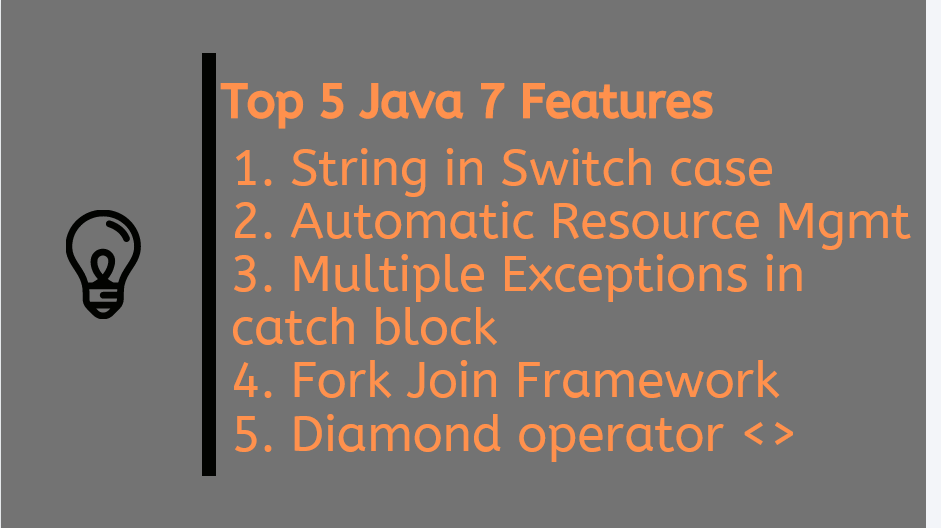
## 6. Spring Configuration Classes

1. *DataSource* details
2. *EntityManagerFactory*
3. *TransactionManage*

## 7. Testing

Let’s create a simple test case

**Java 7 features**



**Java 8 features:**

1) Lambda Expression

2) Functional Interfaces

3) Default methods

4) Predicates

5) Functions

6) Double colon operator(::)

7) Stream API

8) Date and Time AP

write a program for collection type dependency injection

how to import one xml file into another file

what are the diff ways of creating autowired class

how to input from user in IOC

in how many ways can handle exception in Spring

inner bean and outer bean scope

diff bet singleton and mvc design pattern

how to run spring boot application

by using which method in springboot we can sent request to the front end

Did you create Rest service explain

How to authenticate in spring boot

What are mapping used in jpa

If I want to update or delete in postmapping then what will happened it will get error or run

Why we used 400 error code

Why we used apache tomcat server in our application

What is application server for deployment

How will get the record in rest api

Return type of put method

How you will create requestbody in put method

How you will configure multiple datasources in springboot

If bean not found then what to do

How many type of container in java

**Core java**

Immutable Class

Singleton class

Constructor chaining

method chaining

copy constructor

Diff bet Heap and stack memory

diff bet constructor and method

break and continue statements

diff bet static and non static method

class loader

what is garbage collector

how does the constructor work

what is design pattern and explain

what is static and its uses

what is reflection API in java

how can we make singleton pattern synchronized

What is factory design pattern

What is final, finally and finalize

Difference Instance and static

**Collections**

Diff bet arraylist and vector

scenario on map explain its object type scenario

what is Map

scenario to pick linklist over arraylist

copy on arraylist

internal working of hashamap

hash collision

diff bet arraylist and linklist

diff bet comparable and comparator and suitable example for both

write code to iterate map

how to get map element given in the the map

diff bet hashmap and hashset

diff bet hashmap and vector

hierarchy of collection

diff bet map and collection

diff bet arraylist and set

diff scenario on hashmap

diff bet hashmap and hashtable

How get mehod will callinhashmap

What is the syntax for iterating hashamap

What are interfaces in collection

Diff bet set and arraylist

Algoritham used in hash collision

What is the concurrent hashmapexception

Hashset internal working

While iterating hashamap if we add another object what will be the output

Hashmap is thread safe or not

Locking mechanism in concurrent hashmap collection

How we can remove arraylistdupliates if wdon’t know any value or index of that element

Null insertion is possible for hashmap and cchm

Fail fast and fail safe iterator

Use of treeset

**OOPs and other**

OOPs concept

polymorphisam

abstraction

diff bet Abstract class and interace

diff bet Overloading and overriding

pillers of OOPs

Composition and aggreagation

can we access properties of parent and why

inheritance

what is the different scenario considering polymorphim

dynamic polymorphism

what is loose coupling

what is class and object

OOPs real time example

diff scenario of overriding

diff scenario of overloading

Why we use super keyword in constructor

Diamond problem In java

Customized singleton class and it is possible to access class instance in another class

When we make constructor as private the what will be the effect of private constructor

When constructor is autotype then what will happen

What will be the effect when constructor maked a protected

What happen when we make class as private

**Java 8 Featues**

functional interface

stream API

what are different featues of java 7 and java 8

Featues of java 8

Changes in java 8 in collection

How to sort employee data by using employee in stream api

Diff bet functional interface and normal interface

How to break the word using stream api

**java.lang.package**

contract bet equal and hashcode method

what is hashcode and equal method work

how does string memory allocated

what is SCP and explain its diff scenarios

diff bet equals method and == operator

string scenario

memory of string

diff bet stringbuffer and stringbuilder

how will string do mutable

write a program to unduplicate character

write program to reverse a string

write a program for string palindrome

Root class of all java class

How many ways to create object

How many methods in object class

**Hibernate**

primary and entity class

composite primary key

diff bet lazy loading and eager loading

update and merge method diff

diff bet load and get method

what is cache mechanism

how to handle session in project

advantages of hibernate over jdbc

code to configure database in hibernate

what is transaction

explain inheritance in hibernate

how to create a proxy object of hibernate

without hitting the database how to get data from database

why does the hibernate came

what the diff bet basic datasource and driver manager

What is session.begintransaction in hibernate

How you can map in hibernate explain mapping in hibernate. Have you used in your project

What is the use of cascade in mapping

One to many mapping

How to map many to many mapping in java

How to create more than one primary key in entity class

**Exception**

diff bet throw and throws

explain diff scenario considering exception

what is user defined exception and do write a code of user defined exception

explain diff scenario of try catch and finally and differences in flow using return type.

hierarchy of exception

Try with resources

How will you define customized exception

What are exception types

Example of checked and unchecked exception

**Serialization**

what s serialization

What is volatile keyword

Transient keyword

What is synchronization

How to handle when multiple user access only single instance

**MySQL**

diff bet inner join and left join

write query to findout second highest salary of an employee

do write the example of stored procedure and consumer

what is inner join and outer join

write a query to copy one table into another table

write a query to copy column from one table to another

Diff bet drop delete and truncate

Normalization in database

What is view in database

What is cursor in database

What is use of groupbyclause

How to find second max salary from employee table

What is function and prodedure in databse

Join in sql

What is inner ,left , right and full join

What is stored procedures in sql

What is ddl and dmllanguage

What are the ddl language opearations

**Multithreading**

What is thread pool

Types of thread pool

Callable interface

What is runnable and callable interfaces

What methodology you are following I project

build management tool

source control

which is version you have used

if in list there is 5,3,7,8,x... then what is the value of x and how to find it

what is thread and how we implement thread

when multiple static block id given then which block will be excecuted first if above is given method and below is the main method then which method will be executed first.

Logging

Explain your project in detials with programming logic

Explain your work flow in your project from front end to database

What happen when you log in your project after login what options you get

Did you come across any changes in your project with respect to your applications loading problem

Your work methodology model

How long generally your sprint time/point in your project

How do you estimate the time to give for your story in project on what parameters you calculate time

What version control you are using I n your workspace

Are aware of github command

What backend you used for database

Diff bet string and stringbuffer

Find sum of array element

How will check value b=20 present or not in array

How will you get requirement for project

How will you build and deployment.

What is double check

What is object logging

Which type of unit testing do you have perform in your project

Interceptiors

Doccer

Git flow