Spring Framework :

Prerequisites:

1)Core java
2)jdbc
3)Hibernate
4)SQL
5)Html and css

Spring framework:

1)Spring core(IOC)
2)Spring DataJPA
3)SpringMVC
4)Restful WebServices
5)Microservices

How to create Spring Boot Application

-> We can create Spring Boot application in 2 ways

1) Spring Initiazr (start.spring.io) website
2) IDE

Spring Initializer:

-> Spring Intializr is used to create spring boot applications

start.spring.io

-> When we create boot application using initializr it will give the application in the form of zip file.

-> We need to extract that zip file and we should import that project into our IDE.

-> We can create spring boot application directley from IDE but internally IDE will interact with start.spring.io website to create the project.

Note: We should have internet connection to create spring boot applications.

Spring Boot application folder structure

--Maven is build tool which will provide readymade project structure .
-- Apache Maven is a project management software.

What is Maven tool?
Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation.

```
- 01-SpringBoot-App
          - src/main/java
                - com.bikkadit
                    - Application.java
          - src/main/resources
                - application.properties
          - src/test/java
                - com.bikkadit
                    - ApplicationTest.java
          - Maven Dependencies
                - jar files
          - pom.xml
```

-> Spring Boot application we have created using Maven build tool

```
     src/main/java -------------> application source code here

     src/main/resources --------> .xml, .yml, .props here

           src/test/java -------------> unit test classes here

     Maven Dependencies --------> Dependencies will display here

     target -------------------> .class files & artifacts

     pom.xml ------------------> Maven config file
```

-> When we create boot application we are getting "Application.java" class by default. This class is called as start class of Spring Boot.

-> Spring Boot start class is also called as Main class in boot application.

-> Spring Boot application execution will begin from main class only.

```
-----------------------------Application.java--------------------------
@SpringBootApplication
public class Application {

       public static void main(String... args){
     SpringApplication.run(Application.class, args);
       }
}
-----------------------------------------------------------------------
```

-> application.properties file will be created under src/main/resource folder.

-> In this properties we will configure application config props like

1) data source
2) smtp
3) actuators
4) kafka
5) redis etc..

Note: Instead of properties file we can use .yml file also.

-> YAML/YML stands for Yet Another Markup Language

-> In boot application by default test class also created

@SpringBootTest ----> To represent class as SB test class

@Test ------> To represent one method as test case

```
------------------------------ApplicationTest.java---------------------
@SpringBootTest
class ApplicationTests {

     @Test
     void contextLoads() {
     }
}
```

-> In Boot application by default we will get below 2 dependencies

```
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

Internals of Boot start class

-> When we create boot application we will get start class by default.

-> Start class is the entry point for boot application execution.

-> start class is also called as main class in spring boot.

```
@SpringBootApplication
public class Application{

    public static void main(String[] args){
     SpringApplication.run(Application.class, args);
    }
}
```

Bootstrapping process in Spring Boot

-> When we create Spring boot starter project by default we will get Boot start class. It is also called as main class in boot application.

-> Spring Boot application execution will begin from that start class only.

-> That boot start class contains main method which calls SpringApplication.run(..) method

```
-----------------------------------------------------------------------
@SpringBootApplication
public class Application{

       public static void main(String... args){
    SpringApplication.run(Application.class, args);
}
```

-> run ( ) method is performing bootstrapping process for spring boot application.

-> After completion of bootstrapping run ( ) method will return ConfigurableApplicationContext.

-> When our application started, SpringApplication class will be loaded and its constructor will be executed.

-> In SpringApplication class constructor they are calling deduceFromClassPath( ) method. This method is responsible to identify what type of boot application we have created.

-> In Spring Boot we can create below 3 types of applications

1) Reactive Application
2) Web Servlet Application
3) Default Application(Standalone Application)

-> Based on starters we configure in pom.xml it will identify what type of project it is.

```
    spring-boot-starter-webflux =====> Reactive App
    spring-boot-starter-web =========> Web Servlet App
```

spring-boot-starter ==============> Standalone Application

-> Based on type of the application boot will identify the configuration which is required and it will provide that configuration.

-> In Spring framework we will create IOC container using below approaches

1) BeanFactory factory = new XmlBeanFactory (Resource res);

2) ApplicationContext ctxt = new ClassPathXmlApplicationContext(String loc);

-> Programmer no need to create IOC container in Spring Boot.

-> In Spring Boot IOC creation will be taken care by run ( )

-> In Spring Boot, based on our application type IOC will be created.

Standlone : AnnotationConfigApplicationContext

WebServlet: AnnotationConfigServletWebServerApplicationContext

Reactive: AnnotationConfigReactiveWebServerApplicationContext
Runners in Spring Boot

-> Runners will be called from run ( ) method.

-> Runners are used to execute the logic only one time once the application got started.

Ex-1 : Loading data from DB to cache

Ex-2 : Sending email once application got started etc...

-> Spring Boot supports for 2 types Runners

1) ApplicationRunner

2) CommandLineRunner

What is Spring Framework & Why Spring Boot Came?

-> Spring is an application development framework

-> By using Spring framework we can develop end to end application

-> By using Spring framework we can develop below types of applications

a) Standalone applications
b) Web applications (Customer 2 Business Communaction)
c) Distributed applications (Business 2 Business Communication)

-> Spring Framework is developed in Modular Fashion

a) Spring Core
b) Spring Context
c) Spring DAO
d) Spring ORM
e) Spring AOP
f) Spring Web MVC

Note: Spring Core Module is base module for other modules in Spring Framework.

-> Spring Core Module providing fundamental concepts of Spring Framework. They are

a) IOC container
b) Dependency Injection

-> Spring supports loosely coupling among the classes in the application.

-> Spring is an versatile framework (Spring can integrate with other frameworks easily)

-> Spring is a Non-Invasive Framework (Pojo model programming)

When Spring framework is already available why Spring Boot Came into picture?

-> Spring framework providing lot of things for us but We have to integrate all those things by writing lot of configuration.

-> Programmer is responsible to take care of configurations manually in spring framework.

-> Spring Boot is an approach to develop Spring Based applications with less configurations.

-> Spring Boot came into market to provide Auto-Configurations for applications development.

-> Spring Boot internally uses Spring framework only.

Note: Spring Boot is not replacement for Spring framwork. Spring Boot promoting Spring framework.SpringBoot is an enhancement for spring framework.

What is Spring framework?
What type of applications can be developed using Spring?
What are the modules in Spring?
What is the drawback with Spring framework?
What is Spring Boot?

What are the advantages of Spring Boot?

->Spring Boot is an approach to develop Spring Based applications with minimal configurations.

-> Spring Boot is not replacement for Spring framwork because Spring Boot Internally uses Spring framework only.

Spring Boot Advantages

-> Starter Poms

-> Version Management

-> Auto Configuration

-> Embedded Servers

-> Actuators etc..
----

1) Starter Poms

-> Spring Boot provided starter poms to simplify maven/gradle configurations.

a)spring-boot-starter-web
b)spring-boot-starter-data-jpa
c)spring-boot-starter-mail
d)spring-boot-starter-security
e)spring-boot-starter-actuator
f)spring-boot-starter-webflux

2) Version Management

-> When we are adding dependencies in pom.xml no need to configure version for dependency because Spring Boot will take care of versions.
Spring Boot will understand which version is compatiable with Spring Boot version and it will consider that.

3) Auto Configuration

->Auto-Configuration in one of the main advantage of the Spring Boot. Autoconfiguration will take care of configuration required to run our application.

4)-> Embedded Servers

-> Spring Boot providing Embedded Servers to run our web applications. Tomcat is the default embedded server in spring boot.
We can configure external servers like Jetty and undertow.

5)Actuators
-> Actuators are used to provide production ready features from the

application. Actuators will help us in monitoring and maintaing the application.

a) classes loaded
b) threads running (thread dump)
c) objects created (heap dumps)
d) url mapping
e) last 100 http req info
f) config Props
g) env
h) health
i) info etc....

@SpringBootApplication annotation

-> @SpringBootApplication annotation is used to represent start class.

```
@SpringBootApplication
public class Application{

    public static void main(String[] args){
            SpringApplication.run(Application.class, args);
    }
}
```

-> @SpringBootApplication annotation is equal to below 3 annotations

@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan

@ComponentScan:

Component scanning is process of identifying java classes which are represented as spring bean.
In SpringBoot component scan is enabled by default .When we run our springboot application
component scan will happen bydefault.

Component scan will follow package naming convention to scan for the classes which are
represented as spring bean.

Componenet scan will always start from base package.

The package which contains springboot start class is called as base package.

After base package scanning got completed ,it will start scanning sub packages of base package.

The package names which are starting from base package name is called as sub package.

com.bikkadit =(base package)
com.bikkadit.dao = scan
com.bikkadit.controller=scan
com.bikkadit.service=scan
com.bikkadit.repository=scan
in.bikkadit =not scan

@SpringbootConfiguration Annotation :It is equal to @Configuration annotation.
It represents our class as configuration class.It will represent our java class as Spring bean.
 example :Swagger Congiguration.

@EnableAutoConfiguration Annotation :it identifies the configuration which is required to run our
application and will provide that configurations automatically.

Frequently Used Annotations In Spring Boot apps

@Component
@Service
@Repository

@Configuration
@Bean

@Controller
@RestController

@Autowired

@Component : To represent java class as Spring Bean.

@Service: To represent java class as Spring Bean. This is used for service layer classes. Service layer classes contains business logic.

@Repository : To represent java class as Spring Bean. This is used for dao layer classes/interfaces. Dao layer classes contains persistence logic.

@Configuration : To represent java class as Configuration class. Beans customization we can do here.

@Bean : The method which is responsible to create object for spring bean class. This is method level annotation.

@Controller: To represent java class as request handler in web mvc applications. This is used in C 2 B applications.

@RestController: To represent java class as Distributed Component. This is used in B 2 B applications.

@Autowired: To perform dependency injection.

IOC & DI

-> IOC & DI are fundemental concepts of Spring Framework.

-> IOC stands for Inversion of control

-> DI stands for Dependency Injection

-> In spring framework, IOC container is responsible to perform dependency injection among the classes.

-> If we use IOC to perform Dependency Injection then we can achieve loosely coupling among the classes.

-> Application will contain several classes

-> As part of application execution one class should talk to another class.

-> In java, we can use below 2 ways to call one class method in another class method.

1) Inheritence
2) Composition

-> If we use either Inheritence or Composition then our classes will become tighlty coupled.

-> To avoid tighlty coupling among the classes Spring provided IOC & DI.

Q) What is Dependency Injection

-> IOC is a principle which going to manage and colloborate dependencies among the objects available in the application.

-> IOC will manage the lifecycle of beans those which are available in application.

-> IOC will perform Dependency Injection.

-> The process of Injecting one class object into another class is called as Dependency Injection.

-> We can perform Dependency Injection in 3 ways

1) Setter Injection
2) Constructor Injection
3) Field Injection

-> If dependent bean object injected into target bean object through target bean setter method then it is called as 'Setter Injection'.

-> If dependent bean object injected into target bean object through target bean constructor then it is called as 'Constructor Injection'.

-> If dependent bean object injected into target bean object through target bean variable then it is called as 'Field Injection'.

-> To perform dependency injections we are using @Autowired.

Constructor Injection

-> The process of injecting dependent bean object into target bean object using target bean constructor is called as Constructor Injection.

-> By default IOC will call 0-param constructor to create bean object.

-> If we want IOC to call parameterized constructor to create the object we will specify @Autowired annotation.

Note: If we have only one constructor in the target class then @Autowired annotation is optional.

Setter Injection

-> It is the process of injecting dependent bean object into target bean object using target bean setterr method is called as Setter Injection.

-> To enable Setter Injection we will write @Autowired annotation at target class at Setter level.

Field Injection

-> It is the process of injecting dependent bean object into target bean object using target bean variable is called as Field Injection.

-> To enable Field Injection we will write @Autowired annotation at variable level.

```
@Service
public class UserService{

        @Autowired
        private UserDao userdao;

}
```

-> When we specify @Autowired annotation at variable level then IOC will use Reflection API to inject dependent object into target object.

Note: Field Injection not recommended to use

1) It is breaking oops principle

2) It is not supporting for Immutable dependencies

3) It is breaking Single Responsibility Principle

4) It is hiding dependencies

Q) Can we do both setter & Constructor injection for same variable?

Ans) Yes we can do but setter injection will override constructor injection.

-> Constructor injection will happen at the time of object creation.

-> Setter injection will happen after the object creation.

-> In order to perform Dependency Injection we are using @Autowired

-> If we are using @Autowired annotation that means we are giving instruction to IOC to identify dependent bean object and inject into target bean object.

-> @Autowired annotation we can specify at below 3 places

-> Constructor Level

-> Setter Method Level

-> Variable Level

Autowiring is process of injecting dependent bean object into target bean object automatically is called as Autowiring.

-> Autowiring will work based on modes

-> byType
-> byName
-> no

-> "byType" means auto-wiring will happen based on data type of the variable

-> "byName" means auto-wiring will happen based on variable name

-> "no" means auto-wiring will not happen

Note : ByDefault Autowiring uses ByType mode.

Bean Scopes

-> Bean Scope will decide how many objects should be created for bean class by IOC container.

-> In Spring we have below scopes for beans

1) singleton (default)
2) prototype
3) request
4) session
5) global (It removed from Spring 3.0)

-> By default all spring beans are singleton scope that means IOC container will create only one object for the class.

-> Prototype scope is used to create new object everytime.

-> request & session scopes we will use in Spring Web MVC based applications.

```
@Component
@Scope(value = "prototype")
public class Car {

    public Car() {
        System.out.println("****** Car :: Constructor *******");
    }
}
```

```
    @Bean
    @Scope(value = "prototype")
    public Engine getEngine() {
        Engine eng = new Engine();
        return eng;
    }
```

-> To specify scope for a bean we will use @Scope annotation like above.

-> @Scope annotation we can use at class level and method level (@Bean)

@Primary And @Qualifier Annotation.
====================================

If we use both annotations @Primary And @Qualifier at a time then Preferance goes to @Qualifier Annatation.

1) What is Spring Framework?
2) What is Spring Boot?
3) Spring vs Spring Boot
4) Advantages of Spring Boot
5) Spring Boot Starters
6) Spring Initializr website (start.spring.io)
7) Spring Boot First application creation
8) Boot App creation using IDE

9) Spring Boot Application Folder Structure
10) Start class in Spring Boot
11) SpringApplication.run ( ) method
12) @SpringBootApplication
13) Auto Configuration
14) Component Scanning
15) Package Naming Conventions
16) IOC Container
17) Dependency Injection (CI, SI, FI)
18) Why Field Injection Not Recommended
19) Autowiring
20) @Primary & @Qualifier
21) Bean Scopes
22) Stereotype Annotations
23) @Configuration & @Bean annotations
24) Runners in Spring Boot
25) Banner in Spring Boot

Spring Web MVC Module

-> Spring Web MVC is one of the module in Spring Framework

-> By Using Spring Web MVC Module we can build 2 types of applications

1) Web applications
2) Distributed Applications

-> Web applications are meant for Customer to Business Communication (C2B)

Ex: gmail, facebook, irctc, makemytrip etc..

-> Distributed applications are meant for Business to Business Communication (B2B)

Ex :   Passport app <-----------> AADHAR app

flipkart <-------------> Internet Bank apps

-> Spring Web MVC Module is developed based on 2 design patterns

1) MVC Design Pattern
2) Front Controller Design Pattern

-> MVC desgin pattern is used to achieve loosely coupling among components

M ---> Model   (It represents data)
V ---> View   (It represents presentation logic)
C ---> Controller (It contains request handling logic)

-> FrontController Design Pattern is used to perform Pre-Processing and Post-Processing of every request.

-> In Spring Web MVC, DispatcherServlet will act as Front Controller. It is responsible to perform pre-processing and post-processing of a request.

Spring Web MVC Architecture
----
1) DispatcherServlet
2) HandlerMapper
3) Controller
4) ModelAndView
5) ViewResolver
6) View

-> DispatcherServlet is a predefined servlet class available in Spring MVC module. It acts as Front Controller and it is responsible for pre-processing and post-processing of a request.

-> HandlerMapper is responsible to identify which request should be processed by which controller.

-> Controller is called as Request Handler and it is responsible to handle the request.

-> After request processing got completed, Controller method will return ModelAndView object.

Model -> Represents data in key-value format(It contains data)

View -> Logical File Name (jsp/html etc)

Note: In Spring Web MVC, Controllers don't know which presentation technology is used in the project.

-> ViewResolver is used to identify view files available in our application

prefix + viewname + suffix

-> View Component is used to render model data on the view file

----
Building First Web Application Using Spring Boot
----
1) Create Spring Starter Project with below dependencies

i) spring-boot-starter-web
ii) tomcat-embed-jasper   (take this from mvn repo)

2) Configure below properties in application.properties file

i) server port
ii) View Resolver

3) Create Controller class using @Controller annotation

4) Write the methods in controller and bind them to HTTP Request

5) Create View Files (Presentation files)

6) Start the application and test it

Note: When we add spring-boot-starter-web dependency then it will give Apache Tomcat as default Embedded server.

Annotatioons in SpringMVC:

@Controller :
===========
To represent java class as request handler in web mvc applications. This is used in C 2 B applications.

@Controller will return respnce in the form of model and view.

                    Model -> Represents data in key-value format
                    View -> Logical File Name (jsp/html etc)

To represent our class as Controller we are using @Controller annotation

@Controller Annotation will represent our java class as spring bean.

In web application Controller is mandatory to handle the request

@RestController:
===============
@RestController Annotations in spring is nothing but combination of @Controller and @ResponseBody Annotation.It was added in spring 4.0 to
make the development of RestFul Services in spring framework easier.

@RestController will return response in the form of xml,json,text,Html.

@RestController Annotation is used to represent java class as distributed component.This is used in B 2 B applications.

@RestController Annotation will represent our java class as spring bean.

Inside restController class we will write method to handle requests.

-> To bind controller methods to Http Request methods we will use below annotations

GET Request ------> @GetMapping

POST Request -----> @PostMapping

PUT Request ------> @PutMapping

DELETE Request ----> @DeleteMapping

@GetMapping:When our rest controller method is responsible to send information to client then that method will be binded to
HTTP get Request using @GetMapping annotation.

@PostMapping:If Rest api method is responsible to create new record at server side then we will bind that method to http
post request method.To bind Rest Controller method to http Post request we will use @Post mapping Annotation.

@PutMapping:If our Rest Api Method is responsible to update existing record then we will bind our RestController
method to http put request method .To bind our RestController method to http put request method we will use @PutMapping Annotation.

@DeleteMapping: If our Rest Api Method is responsible to delete record then we will bind our RestController
method to http Delete request method .To bind our RestController method to http delete request method we will use @DeleteMapping Annotation.

@RequestBody: @RequestBody annotation is used to indicating a method parameter should be bind to the body of the HTTP request.
Internally, this annotation uses HTTP Message converters to convert

the body of HTTP requests to java objects.
-> In web applications we will perform below 2 operations

1) Sending data from controller to UI

2) Taking data from UI to Controller

Sending Data From Controller to UI

-> To send data from Controller to UI we will use Model/ModelAndView object

-> Model is a Map which represents data in the form of Key-value pair

    model.addAttribute(String key, Object value)

-> We can get the value using Key

2) Taking data from UI to Controller:
==

> To recieve data from UI to controller we have below options

1) Query Parameters

2) Path Parameters

3) Request Body

Query Parameters
====================
-> Query Parameter will re-present data in URL in the form of Key-value pair

    Ex : localhost:9090/course?name=SBMS

-> Query parameters will start with ? symbol

-> Query Parameters will be seperated by & symbol

Ex : localhost:9090/course?name=SBMS&trainerName=Santosh

-> Query Paramers will present only at end of the URL

-> Query Parameters are used to send data from UI to server in URL

-> To read Query Parameter value we will use @RequestParam annotation

Path Parameters
================
-> Path Parameters will represent data in URL directley without key

    Ex: localhost:9090/course/{name}/details

-> Path Parameters can present any where in the URL

-> Path Parameter will start with / symbol and will be seperated by / symbol only

Ex: localhost:9090/course/{name}/details/{trainerName}

-> To read Path Parameter values we will use @PathVariable annotation in controller

----

Query Parameters & Path Parameters

-> Query Parameters & Path Parameters are used to send data to server in URL

-> Query Parameters will represent data in Key-Value format

-> Path Parameters will represent data directley

-> Query Parameters should present only at the end of the URL

-> Path Parameters can present anywhere in the URL

-> Query Parameters will be seperated by '&' symbol

-> Path Parameters will be seperated by '/'

-> To read Query Parameters from the URL we will use @RequestParam annotation

-> To read Path Parameters from the URL we will use @PathVariable annotation

> To work with Query parameters we will use @RequestParam annotation

-> To work with Path Parameters we will use @PathVariable annotation
----
-> Query Parameters & Path Parameters will display data in URL.

-> Query Parameters & Path Parameters are only recommended to send Small and non-sensitive text data.

-> If we want to send huge & sensitive data then we should go for Request body.

-> HTTP POST, PUT and DELETE requests contains request body where as GET request doesn't have request body.

Spring Data JPA :

Spring Data JPA Introduction

-> Spring Data JPA came into market to simplify CRUD operations in applications.

C - create
R - Retrieve
U - Update
D - Delete

-> Performing CRUD operation is very common requirement in applications.

-> Spring Data JPA provided Repositories to simplify CRUD operations

1) CrudRepository
2) JpaRepository

-> CrudRepository provided methods to perform only Crud Operations

-> JpaRepository provided methods to perform Crud Operations + Pagination + Sorting.

----
1) What is Persistence Layer?
Ans) Persistence layer contains components which are responsible to communicate with database.

Best Practises To Follow In Persistence Layer
----
1) For Every table we shuld have a seperate DAO

USER_MASTER -----------> UserMasterDao.java

ROLE_MASTER -----------> RoleMasterDao.java

User_ROLES ------------> UserRolesDao.java

DC_CASES --------------> DcCasesDao.java

2) For every table we shold maintain atleast one Primary key.

Primary key = Not Null + unique.

3) In every table we shoud maintain below 4 columns for data analysis.

CREATED_DATE
UPDATED_DATE
CREATED_BY
UPDATED_BY

4) We should not ask end user to enter value for Primary Key Column because they may enter duplicate value.

5) We will use Generators to generate the value for Primary key.

6) We should not use physical connections to communicate with database.

7) We should maintain connection pooling for db connections.

8) We should manage transactions in our applications.

9) We should maintain cache for static tables data.

What is ORM?

-> ORM stands for Object Relational Mapping

-> This ORM is very famous technique which is used to develop persistence layer.

-> Below are the main components in ORM

1) Entity
2) Mapping
3) Configuration

-> The java class which is mapped with data base table is called as Entity.

-> To map java class with db table we will use annotations

```
@Entity:To represent java class as Entity class.
@Table : Mapping class name with table name.
@Id    : To reprensent variable as primary key column.
@Column :To map variable name with table column name.
```

-> DataSource details we will configure in application.properties or application.yml file.

-> In java applications we can develop persistence layer in below ways

1) JDBC
2) Spring JDBC
3) Hibernate
4) Spring ORM
5) Spring Data JPA

Developing First Application Using Data JPA
----
1) Create Spring Starter Project with below dependencies

1) spring-boot-starter
2) spring-boot-starter-data-jpa
3) Mysql driver

2) Create Entity Class (Mapping with DB table)

3) Create Repository Interface and extend properties from JPA Repository

4) Configure Data Source & ORM Properties in application.properties file

5) Call Repository methods and test the application.

Spring Data JPA Internals

-> Spring Data JPA came into market to simplify CRUD operations in the application.

-> Before Data JPA, we used to write lot of boiler plate code to perform Crud Operations.

-> Data JPA Provided Repository interfaces to perform CRUD operations.

1) CrudRepository
2) JpaRepository

-> If our interface extend properties from any one of above repository then our interface will get methods to perform Crud operations.

```
public interface UserRepository extends CrudRepository<T, ID>{

}
```

-> First Argument Represents Entity Class
-> Second Argument Represents DataType of Primary Key

Note: @Repository is optional.

CrudRepository interface methods

-> CrudRepository is a predefined interface available in Spring Data JPA.

-> CrudRepository interface providing methods to perform Crud Operations in our application

1) save(Object T)
2) saveAll(Iterable<Object> entities)

3) findById(Serializable ID)
4) findAllById(Iterable<Serializable> ids)
5) findAll( )

6) count()
7) isExistsById(Serializable id)

8) deleteById(Serializable id)
9) deleteAllById(Iterable<Serializable> ids)
10) delete(Object T)
11) deleteAll(Object Entities)
12) deleteAll ( )
----

save(Object E) -> Insert / Update The Record.

saveAll (Iterable<E> entities) -> Insert/Update Collection of records

Note: With given primary if record already present in the table then it will execute update query else it will execute insert query

findById(Serilizable ID) : To retrieve record using Primary key

findAllById(Iterable<Serializable> ids) : To records using multiple Primary Keys

findAll ( ) : To retrieve all records from the table

existsById(Serilizable Id) : To check presence of record using Primary Key

count ( ) : To get total records count of the table.

deleteById(Serializable id) : To delete a record using primary key

deleteAllById(Iterable ids) : To delete multiple records using primary keys

delete(Object entity): To delete a record based on entity object

deleteAll ( ) : To delete all records from the table

Working with findByXXX methods in Spring Data JPA

-> If we want to retrieve records using Non Primary Key columns then we can go for findBy methods.

-> When we write findBy methods, based on method name Data JPA will construct the query

Note: Method Name will play important role in this process.

```
public interface UserRepository extends CrudRepository<UserEntity, Serializable> {

    public UserEntity findByEmail(String email);

    public UserEntity findByUname(String name);

    public List<UserEntity> findByAge(Integer age);
    public UserEntity findByEmailAndUname(String email, String name);

    public UserEntity findByAgeGreaterThan(Integer age);

}
```

JpaRepository

-> JpaRepository is a predefined interface available in Spring Data JPA

-> By using JpaRepository also we can perform Crud Operations

-> JpaRepository is superior than CrudRepository

-> JpaRepository supporting below 3 additional functionalities

1) Pagination
2) Sorting
3) Query By Example

JpaRepository & Methods available in JpaRepository

-> Using JpaRepository we can perform below operations

1) Crud Operations
2) Sorting
3) Pagination
4) Query By Example

-> Sorting technique we will use to sort the data while retriving

```
findAll(Sort.by("uname").ascending());
```

-> Query By Example is used to construct the query dynamically based on values available in Entity class obj.

```
UserEntity entity = new UserEntity();
entity.setUname("Santosh");
entity.setAge(29);

Example of = Example.of(entity);

findAll(of);
```

Pagination

-> Pagination is the process of dividing total records into multiple pages.

```
PageRequest pageReq = PageRequest.of(pageNo, pageSize);
findAll(pageReq);
```

Working with Custom Queries in Data JPA

-> In this approach we will write the query and we will execute it.

-> Custom Queries we can execute in 2 ways

1) Native SQL Queries
2) HQL Queries

-> Native SQL queries are database dependent queries

-> Native SQL queries contains table name and column names

-> If we change our application communication from one database to another database then few native sql queries may not support then we have to change them in our application.

-> If we want to change from one DB to another DB entire application re-testing is required.

-> To avoid the problems of Native SQL queries, we will use HQL queries

-> HQL stands for Hibernate Query Language

-> HQL queries are database independent queries

-> In HQL queries we will use Entity class name and entity class variable names.

-> HQL queries can't be executed in database directley. HQL queries will be converted into SQL queries before exeuction.

-> HQL queries will be converted into SQL queries by Dialect classes.

Diffrence between SQL and HQL Queries:
===
-> Native SQL queries are database dependent queries

-> HQL queries are database independent queries.

-> HQL queries will be converted to SQL queries before execution.

-> Dialect class will convert HQL query to SQL query

-> To execute custom queries we will use @Query annotation

-> From maintenence standpoint HQL queries are recommended

-> From performance standpoint SQL queries are recommended

SQL : select * from user_details;
HQL : from UserEntity;

SQL : select * from user_details where user_id=101;
HQL : from UserEntity where userId=101;

SQL : select * from user_details where user_id=101 and user_name='Santosh';
HQL : from UserEntity where userId=101 and username ='Santosh';

SQL : select user_id, user_name where user_age =30;
HQL : select userId, username where userAge=30;

```
public interface UserRepository extends CrudRepository<UserEntity, Serializable> {

          //Hql Query
    @Query("from UserEntity where email=:email")
    public UserEntity findUserByEmail(String email);
        //Hql Query
    @Query("select userId, uname from UserEntity where age=:age")
    public Object[] findUserByAge(Integer age);
    //Sql Query
    @Query(value="select * from user_details where user_id=:userId", nativeQuery=true)
    public UserEntity findUserById(Integer userId);

}
```

-> In Order to execute custom queries for insert, update and delete we will use below annotations in data jpa

```
        @Modifying
        @Transactional
        @Query
```

Connection Pooling

-> If our application wants to communicate with database then we should have connection with database.

```
    Connection con = DriverManager.getConnection(url,uname,pwd);
```

-> If we get connection from database using above line then that connection will be called as Physical Connection.

-> If we open physical connection with database then we are responsible to close that connection.

```
        con.close( );
```

-> If we don't close that connection other resources can't access that connection.

-> If connections are completed in DB then we will get Connections Exhausted Problem.

Note: Every db will have limited no.of connections.

-> To avoid DB connections problems we will use Connection Pooling.

-> Connection Pooling is the process of keeping connections ready which are required for our application.

-> Connection pooling is the process of creating and storing connections in a pool once application got started.

-> Connection Pool will be managed by Pool Manager

-> When the application got started then Connection Pool will be created and it will keep connections ready

-> The programs which requires DB connection will get that connection from Connection Pool instead of getting connection from Database.

-> Hikari Connection Pool is the default connection pool using in Spring data jpa applications
-Bydefault it will provide 10 connections.

-> In some applications we will use Server Managed Connection Pools

Generators

-> Generators are used to generate the value for primary key.

-> It is not at all recommended to ask end user to enter the value for primary key because they may enter duplicate value.

-> Primary key is a constraint which is combination of below 2 constraints

1) NOT NULL
2) UNIQUE

-> When we use generators application is responsible to generate the value for primary key column
----
-> To generate value for primary key we used @GeneratedValue annotation

-> If our underlying database is ORACLE then this annotation will use HIBERNATE_SEQUENCE to generate value for primary key.

-> HIBERNTATE_SEQUENCE value will start from 1 and will be incremented by 1.

-> If our underlying database in MYSQL then this annotation will use AUTO_INCREMENT to generate value for primary key.

NOTE: SEQUENCES are DB dependent. Oracle will support where as MYSQL will not support.

```
@Entity
@Table(name = "STUDENTS_TBL")
public class Student {

    @Id
    @Column(name = "STUDENT_ID")

    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer studentId;
    @Column(name = "STUDENT_NAME")
    private String studentName;

    @Column(name = "STUDENT_EMAIL")
    private String studentEmail;

    @Column(name = "STUDENT_RANKS")
    private Integer studentRank;

}
```

-> Timestamping is the process of generating values for CREATED_DATE AND UPDATED_DATE columns.

```
        @CreationTimestamp

        @UpdateTimestamp
```

-> CREATED_DATE & UPDATED_DATE columns will help us in analyzing the table data.

Composite Primary Key:

-> Primary Key is a constraint which is combination of below 2 constraints

1) UNIQUE
2) NOT NULL

-> Primary key constraint will make sure our table contains unique data

-> Primary Key is mandatory for the table to use ORM frameworks.

-> To map Entity class variable to primary key column we will use @Id annotation

-> To generate the value for Primary Key column we will use Generators.

-> We can have more than one column as primary key in database table.

-> If we have more than one column as primary key then we will call it as Composite Primary key.

-> We can't use generators for composite primary key

```
@Data
@Entity
@Table(name = "BANK_ACCOUNTS")
@IdClass(AccountPK.class)
public class Account {

    @Column(name = "BRANCH_NAME")
    private String branchName;

    @Column(name = "MIN_BAL")
    private Double minBal;

    @Id
    private Integer accId;

    @Id
    private String accType;

    @Id
    private String holderName;
}
```

```
@Data
public class AccountPK implements Serializable {

    private Integer accId;
    private String accType;
    private String holderName;
}
------------------------------------------------------------------------------
public interface AccountRepository extends JpaRepository<Account,
AccountPK> {

}
------------------------------------------------------------------------------
```

How to rollback transaction

-> The unit amount of work we are doing is called as one transaction. If any operation is failed we are going to rollback all other operations also in the transaction. It means Either or all or None.

```
          @Transactional(rollbackFor=Exception.class)
```

Example:

```
package com.bikkadIt.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Employee {

     @Id
     private int empId;

     private String empName;

     private Double empSal;

     public int getEmpId() {
          return empId;
     }

     public void setEmpId(int empId) {
```

```
		this.empId = empId;
	}

	public String getEmpName() {
		return empName;
	}

	public void setEmpName(String empName) {
		this.empName = empName;
	}

	public Double getEmpSal() {
		return empSal;
	}

	public void setEmpSal(Double empSal) {
		this.empSal = empSal;
	}

	@Override
	public String toString() {
		return "Employee [empId=" + empId + ", empName=" +
empName + ", empSal=" + empSal + "]";
	}


}



package com.bikkadIt.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Address {

	@Id
	private int addrId;

	private String city;

	private String state;
private String country;

private int empId;

public int getAddrId() {
     return addrId;
}

public void setAddrId(int addrId) {
     this.addrId = addrId;
}

public String getCity() {
     return city;
}

public void setCity(String city) {
     this.city = city;
}

public String getState() {
     return state;
}

public void setState(String state) {
     this.state = state;
}

public String getCountry() {
     return country;
}

public void setCountry(String country) {
     this.country = country;
}

public int getEmpId() {
     return empId;
}

public void setEmpId(int empId) {
     this.empId = empId;
}
```

```
	@Override
	public String toString() {
		return "Address [addrId=" + addrId + ", city=" + city + ", state=" + state + ", country=" + country + ", empId="
				+ empId + "]";
	}


}
```

```
package com.bikkadIt.repository;

import java.io.Serializable;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.bikkadIt.model.Address;
@Repository
public interface AddressRepository extends JpaRepository<Address, Serializable>{

}
```

```
package com.bikkadIt.repository;

import java.io.Serializable;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.bikkadIt.model.Employee;

@Repository
public interface EmployeeRepository extends JpaRepository<Employee, Serializable>{

}
```

```
package com.bikkadIt.sercice;

public interface UserServiceI {

    public void saveData() ;
}



package com.bikkadIt.sercice;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import com.bikkadIt.model.Address;
import com.bikkadIt.model.Employee;
import com.bikkadIt.repository.AddressRepository;
import com.bikkadIt.repository.EmployeeRepository;

@Service
public class UserServiceImpl implements UserServiceI{

    private EmployeeRepository employeeRepository;

    private AddressRepository addressRepository;



    @Autowired
    public UserServiceImpl(EmployeeRepository employeeRepository,
AddressRepository addressRepository) {
        super();
        this.employeeRepository = employeeRepository;
        this.addressRepository = addressRepository;
    }




    @Transactional(rollbackFor = Exception.class)
    public void saveData() {
          Employee emp =new Employee();
          emp.setEmpId(103);
           emp.setEmpName("Santosh");
           emp.setEmpSal(10000.00);
           employeeRepository.save(emp);

           int a=10/0;

           Address addr=new Address();
           addr.setAddrId(503);
           addr.setCity("pune");
           addr.setState("Maharashtra");
           addr.setCountry("India");
           addr.setEmpId(103);
           addressRepository.save(addr);
     }

}


package com.bikkadIt;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.autoconfigure.security.SecurityProperties.
User;
import
org.springframework.context.ConfigurableApplicationContext;

import com.bikkadIt.sercice.UserServiceI;

@SpringBootApplication
public class HowToRollbackTransApplication {

     public static void main(String[] args) {
     ConfigurableApplicationContext                          context=
     SpringApplication.run(HowToRollbackTransApplication.class,
args);

        UserServiceI                                        userServiceI=
     context.getBean(UserServiceI.class);
```

```
            userServiceI.saveData();

            context.close();
    }

}
```

HIBERNATE

Sources :

https://www.interviewbit.com/hibernate-interview-questions/

https://www.javatpoint.com/hibernate-interview-questions

Assignments:(Prepare by your own)

1) it converts all checked exceptions into unchecked exceptions.
2)it provides criteria builder API.
3)it suppotrs validation features.
4)it provides cache mechanism.
   1)first level cache
   2)second level cache.

-it provides inheritance relationship .
1)default inheritance
2)single table
3)joint for table per class.

- What is automatic dirty checking in hibernate?

- How to connect two DataBases in one application.

Hibernate:

Hibernate is java framework that simplifies the development of java appliacation to interact with database.

Hibernate is ORM(Object relational mapping) tool.

It reduces the developer efforts ,time ,cost.

hibernate is an opensource,lightweight framework.

It is invented by gavin king in 2001.

Any type of appliacation can build with Hibernate framework.
ex.standalone application,web application,Distributed application.

Advantage of Hibernate over Jdbc :
1)Hibernate generate sql query at run time according database.
2)it provides automatic table creation feature.
3)It provides primary key auto generation feature.
4)it converts all checked exceptions into unchecked exceptions.
5)Hibernate provides their own query language i.e HQL(Hibernate query language).
6)it provides criteria builder API.
7)it suppotrs validation features.
8)it provides cache mechanism.
   1)first level cache
   2)second level cache.
-It provide relationship mapping
1)one to one mapping
2)one to many mapping.
3)many to one mapping.

4)many to many mapping.

-it provides inheritance relationship .
1)default inheritance
2)single table
3)joint for table per class.

1) List some of the databases supported by Hibernate?

1)MySQL
2)Oracle
3)HSQL
4)PostgreSQL

2)What is SessionFactory ?
SessionFactory provides the instance of Session.It holds the data of second level cache that is not enabled by default.
The org.hibernate.SessionFactory interface provides factory method to get the object of Session.

3) Is SessionFactory a thread-safe object?
Yes, SessionFactory is a thread-safe object, many threads cannot access it simultaneously.

4)What is Session?
It holds a first-level cache of data. The org.hibernate.Session interface provides methods to insert, update and delete the object.

5)Is Session a thread-safe object?
No, Session is not a thread-safe object, many threads can access it simultaneously. In other words, you can share it between threads.

6) What is save method?

Hibernate provides the save method on the session interface to save a record into the database.
So the session.save inserts a record into the database.
It returns the id, that is the primary key that will be assigned to the database record.

7) What is saveOrUpdate method?
In addition to the save method, Hibernate also provides a method called saveOrUpdate on the session
interface. Just as the name suggests, this method either performs a save operation or an update
operation. If there is no record in the database corresponding to the object passed in,
it inserts a record, but if there is a record corresponding to the object passed in,
it just updates the existing record.

7.2) What is persist Method()?

persist method is used INSERT records into the database,
but return type of persist is void.

8) What is the difference between get and load method?

get()	load()
1) Returns null if an object is not found.	1Throws ObjectNotFound Exception if object is not found.
2) get() method always hit the database.	2)load() method doesn't hit the database always .
3) It returns the real object, not the proxy.	3) It returns proxy object.
4) It should be used if you are not sure about the existence of instance.	4)It should be used if you are sure that instance exists.

9) What is the difference between update and merge method?
The differences between update() and merge() methods are given below.

No.	The update() method	merge() method
1)	Update means to update something.	Merge means to combine something.
2)	update() should be used if the session doesn't contain an already persistent state with the same id. It means an update should be used inside the session only. After closing the session, it will throw the error.	merge() should be used ifsession you don't know the state of the session, means you want to make the modification at any time.

10) What are the states of the object in hibernate?(Life cycle of Object in Hibernate)
There are 3 states of the object (instance) in hibernate.

Transient: The object is in a transient state if it is just created and not associated with a session.

Persistent: The object is in a persistent state if a session is open, and you just saved the
instance in the database or retrieved the instance from the database.

Detached: The object is in a detached state if a session is closed.
After detached state, the object comes to persistent state if you call lock() or update() method.

11)How to make an immutable class in hibernate?
Immutable class in hibernate creation could be in the following way.

If we are using the XML form of configuration, then a class can be made immutable by marking mutable=false.

The default value is true there which indicating that the class was not created by default.

In the case of using annotations, immutable classes in hibernate can also be created by
using @Immutable annotation.

12) What is automatic dirty checking in hibernate?

The automatic dirty checking feature of Hibernate, calls update statement automatically on the objects
that are modified in a transaction.

13)How many types of association mapping are possible in hibernate?
There can be 4 types of association mapping in hibernate.
One to One
One to Many
Many to One
Many to Many

15) What is lazy loading in hibernate?
Lazy loading in hibernate improves the performance. It loads the child objects on demand.

Since Hibernate 3, lazy loading is enabled by default, and you don't need to do lazy="true". It means not to load the child objects when the parent is loaded.

16) What is Eager loading in hibernate?
Eager loading in hibernate it will decrease the performance. It loads the child objects automatically when application started.

It means it will load the child objects when the parent is loaded.

16)What is HQL (Hibernate Query Language)?
Hibernate Query Language is known as an object-oriented query

language.
It is like a structured query language (SQL).

The main advantage of HQL over SQL is:

Database independent.
Simple to write a query.

17)What is the difference between first level cache and second level cache?

No. First Level Cache   Second Level Cache

1) First Level Cache is associated with Session.   1) Second Level Cache is associated with SessionFactory.

2) It is enabled by default.   2) It is not enabled by default.We have to enable it.
For Enabling we have to add EH Cache jar.

3) The first level cache is available only until the session is open, 3) The second-level cache is available through the
the first level cache is destroyed. once the session is closed, application's life cycle, It is only destroyed when application is stopped.

18)What can you tell about Hibernate Configuration File?
Hibernate Configuration File or hibernate.cfg.xml is one of the most required configuration files in Hibernate.
By default, this file is placed under the src/main/resource folder.
The file contains database related configurations and session-related configurations.
Hibernate facilitates providing the configuration either in an XML file (like hibernate.cfg.xml) or a properties file (like hibernate.properties).

This file is used to define the below information:

Database connection details: Driver class, URL, username, and password.

hibernate.cfg.xml

There must be one configuration file for each database used in the application,
 suppose if we want to connect with 2 databases, then we must create 2 configuration files with different names.

Hibernate properties: Dialect, show_sql, second_level_cache, and mapping file names.

Hibernate.properties

What is SDLC ?

-> SDLC stands for software development life cycle.

-> SDLC explains application development process from starting to ending.

-> There are 7 stages involved in SDLC

1) Requirements Gathering
2) Requirements Analysis
3) Requirements Design
4) Development
5) Testing
6) Deployment
7) Maintenence

-Methodologies
----
1) Waterfall Model
2) V-Model
3) Iterative Model
4) Spiral Model
5) Agile Model (Currently Trending In Market)

-> In Waterfall model we will follow sequential process.
All Phases will be completed one after other.

-> Requirements & Budget are fixed in waterfall model

-> Accomdating New changes in waterfall model is difficult

-> Agile Methodology is a practice that promotes continuous iteration of development
and testing throughout the software development lifecycle of the project.

-> Both development and testing activities will happen parallely in Agile Methodology.

Agile Terminology

Product Owner
Scrum Master
Tech Lead
Team Members

-> Product Owner is the person who is responsible for devlivering project/product to client.

-> Scrum Master is the person who will manage, monitor and colloborate agile team members work

-> Tech Lead is the person who is responsible to solve technical problems in project (Kind of Team Lead)

-> Team Members nothing but Developers & Testers

Story
Story Points
Backlog Grooming
Backlog
Sprint Planning
Sprint
Scrum
Mid Iteration Review
Retrospective

-> Story nothing but a task that we have to work on.

-> For Every Story we will have story points. Story points will decide duration to complete that story.

3 points - 1 day task

5 points - 2 days task

8 points - 3 days task

-> Backlog Grooming is the meeting that will be conducted by Scrum Master with
Team members to discuss whatever task we have to do in sprint in the project.

-> Once Backlog Grooming meeting got completed we will create backlog stories in JIRA

Note: In project any body can create backlog stories.

-> Sprint Planning is the meeting that will be conducted by Scrum Master to prioritize
 the stories which are in the backlog.

-> Sprint means the stories that are prioritized in Sprint Planning for fixed durtion to complete.
 Sprint duration will be 2 weeks.

-> Mid-Iteration Review is the meeting that will be conducted by Scrum Master once
 first week completed in Sprint to review stories which are completed and which are in pending.

-> Retrospective is the meeting that will be conducted by Scrum Master once Sprint got completed.
 In Retrospective meeting we will discuss about last 2 weeks work, lessons learnt,
 achievements, improvements etc.

SOLID OOPS Principles

-> Java is an Object Oriented Programming Language.

-> While developing the project it is highly recommended to follow SOLID oops Princples also to have better design and easy maintenence.

-> SOLID principles

S --> Single Responsibility

O --> Open Closed

L --> Liskov Substitution

I --> Interface Seggregation

D --> Dependency Inversion

SOLID OOPS
===================================
S - Single Responsibility Principle (known as SRP)
O - Open/Closed Principle
L - Liskov's Substitution Principle
I - Interface Segregation Principle
D - Dependency Inversion Principle

Single Responsibility Principle

"One class should have one and only one responsibility"

Ex : Reading Excel, Filtering Data, Storing in DB

ReadExcel.java (It is responsible for reading data from excel)
FilterData.java (It is responsible for data filtering)
PatientsDao.java (Its responsible for db logic)

Note: Once class should have only one reason to modify

Open Closed Principle

"Software components should be open for extension, but closed for modification"

Ex : DispatcherServlet (Its functionality is closed but we can extend this)

Liskov's Substitution Principle

"Derived types must be completely substitutable for their base types"

Example:

```
public abstarct class SocialMedia{

public void chatWithFriend();
public void publishPost();
public void sendPhotoAndVideos();
public void groupvideocall();
}

public class FaceBook extends SocialMedia{

public void chatWithFriend(){
}
public void publishPost(){
}
public void sendPhotoAndVideos(){
}
public void groupvideocall(){
}
}

public class WhatsApp extends SocialMedia{

public void chatWithFriend(){
}
public void publishPost(){
// we can not publish post
}
public void sendPhotoAndVideos(){
}
public void groupvideocall(){
}


public class instagram extends SocialMedia{

public void chatWithFriend(){
}
public void publishPost(){

}
public void sendPhotoAndVideos(){
}
public void groupvideocall(){
//we can not do group vido call
}
```

Solution to this is

```
public interface SocialMediaInterf{

public void chatWithFriend();
public void sendPhotoAndVideos();

}

public interface publishPostInterf{

public void publishPost();

}

public interface groupVideoCallingInterf{
public void groupvideocall();

}


public class FaceBook implements
SocialMediaInterf,publishPostInterf,groupVideoCallingInterf{

public void chatWithFriend(){
}
public void publishPost(){
}
public void sendPhotoAndVideos(){
}
public void groupvideocall(){
}
}


public class WhatsApp implements SocialMediaInterf,groupVideoCallingInterf{

public void chatWithFriend(){
}

public void sendPhotoAndVideos(){
}
```

```
public void groupvideocall(){
}


public class instagram implements   SocialMediaInterf,publishPostInterf{

public void chatWithFriend(){
}
public void publishPost(){

}
public void sendPhotoAndVideos(){
}
```

Interface Segreggation Principle

“Clients should not be forced to implement unnecessary methods which they will not use”

Take an example. Developer Santosh created an interface Reportable and added two methods generateExcel() and generatedPdf(). Now client ‘A’ wants to use this interface but he wants to use reports only in PDF format and not in excel. Will he be able to use the functionality easily?

NO. He will have to implement both the methods, out of which one is extra burden put on him by designer of software. Either he will implement another method or leave it blank. This is not a good design.

So what is the solution? Solution is to create two interfaces by breaking the existing one. They should be like PdfReportable and ExcelReportable. This will give the flexibility to user to use only required functionality.

```
---------------------------------------------------
public interface ReportGenerator{
      public void generatePdfReport();
      public void generateExcelReport();
}
--------------------------------------------------------
```

```
public interface PdfReportGenerator{
        public void generatePdfReport();
}

public interface ExcelReportGenerator{
      public void generateExcelReport();
}
```

Note: Spring Bean Life Cycle interfaces.Spring provided life cycle methods init() and destory().

-> InitialingBean contains init() method

-> DisposableBean contains destory() method

Dependency Inversion Principle

“Depend on abstractions, not on concretions”

Example:

```
public class CreditCard {

     public void doTransaction(long amount){
          System.out.println("payment using Credit card");
     }
}
```

```
public class DebitCard {

     public void doTransaction(long amount){
          System.out.println("payment using Debit card");
     }
}
```

```
public class ShoppingMall {

    private DebitCard debitCard;

    public ShoppingMall(DebitCard debitCard) {
        this.debitcard= debitcard;
    }

    public void doPurchaseSomething(long amount){
        bankCard.doTransaction(amount);
    }

    public static void main(String[] args) {
        DebitCard debitCard=new DebitCard();



        ShoppingMall shoppingMall=new ShoppingMall(debitCard);
        shoppingMall.doPurchaseSomething(5000);
    }
}


public class ShoppingMall {

    private   CreditCard creditCard;

    public ShoppingMall( CreditCard creditCard) {
        this.creditcard= creditCard;
    }

    public void doPurchaseSomething(long amount){
        bankCard.doTransaction(amount);
    }

    public static void main(String[] args) {
         CreditCard creditCard=new CreditCard();


        ShoppingMall shoppingMall=new ShoppingMall(creditCard);
        shoppingMall.doPurchaseSomething(5000);
    }
}


public interface BankCard {

    public void doTransaction(long amount);
}




public class CreditCard implements BankCard{

    public void doTransaction(long amount){
        System.out.println("payment using Credit card");
    }
}





public class DebitCard implements BankCard{

    public void doTransaction(long amount){
        System.out.println("payment using Debit card");
    }
}




public class ShoppingMall {

    private BankCard bankCard;

    public ShoppingMall(BankCard bankCard) {
        this.bankCard = bankCard;
    }

    public void doPurchaseSomething(long amount){
        bankCard.doTransaction(amount);
    }

    public static void main(String[] args) {


        BankCard bankCard=new CreditCard();
        ShoppingMall shoppingMall=new ShoppingMall(bankCard);
        shoppingMall.doPurchaseSomething(5000);
    }
}
```

Introduction:

Hii, Thanks for giving me this opportunity to introduce about myself. This is Shrinivas patil having 2+ years of experiance as software engineer .I am currently working with Intelligic Software Private Limited, Pune.

I have 2+ years of experiance in Core Java,J2EE ,Hibernate,Spring,SpringBoot,RestFul WebServices,GIT,JIRA,MySql DataBase.

My Roles and Resopnsibilities are

1)I designed and developed using design patterns.
2)I involved in application level coding and databse design.
3)I developed Controller Layer,Service Layer and Persiatsance layer.
4) I am responsible for writting test cases,performing unit testing.
5)I am also resopnsible for handling exceptions in Application.
6)I have hands on experiance with MYSQL DataBase.
7)Designing, implementing and maintaining Java-based applications
8)Contributing in all phases of the development lifecycle
9)Writing testable, scalable and efficient code
10)Test and debug new applications and updates

I have worked on Insurance Domain.
My project name is Poverty Alleviation.

-> The main aim of Poverty Alleviation project is used to provide Govt Schemers for Durango state citizens.
As part of this project Durango govt is providing some health and insurance plans for Durango state citizens.These Plans are

1) NAP (Nutrition Assistance Program)
2) CAP (Child Assistance Program)
3) RIW (Durango Works)
4) Medicaid
5) Medicare

In Poverty Alleviation application, we have some modules.These modules are

1) Admin
2) CR (Citizen registration)
3) CDC (Citizen Data Collection)
4) CED (Citizen Eligibility Determination)
5) CO (Correspondence)
6) BI (Beneficary Issuance)
7) Batches
8) Reports

Version Control Software

-> Multiple developers will be involved in project development.

-> Developers will be working from different locations.

> We will use version control softwares in our project mainley for below 2 reaons

1) Source Code Integration
2) Source Code Changes Tracking

Problem-1 : How all the developers code will be integrated ?

Problem-2 : How to track which developer made code changes ?

-> To resolve above 2 problems we will use "Version Control Software" in the project.

-> Below are the version control softwares available in the market.

1) CVS ---------------outdated
2) Clear Case --------outdated
3) SVN ------------------almost outdated
4) Git Hub ------------Trending
5) BitBucket-----------Trending

Git Hub

----

1) What is Git Hub?
Ans) It is a cloud provider

2) What is Repository?
Ans) We will create Repositories in Git Hub. Repository is a place where we will store our code.

Note: For one project one repository will be created.

3) What is Git Client?
Ans) Git client is a software which is used to perform GIT operations in Git repository.

Note: Every developer will install git client software in thier System.

-> In GIT Hub account we can create 2 types of repositories

1) Public Repository
2) Private Repository

-> Public Repository means anybody can see but we   will choose who can commit

-> Private Repository means we will decide who can see and who can commit.

Git Bash

-> Git Bash is a command line software which is used to perform git operations.

-> We will execute GIT commands using GIT Bash

Git Commands
----
git help : It will display frequentley used git commands

git help <cmd-name> : To open command documentation

git init : To initialize our folder as git operations folder

git status: To display staged, un-staged and un-tracked files

Un-Tracked: New Files (We need to add these for commit)
These file names will be displayed in red colour.

Staged : Files are ready to commit (These are already added)
These file names will be displayed in green colour.

Un-Staged: Modified Files (We need to add these for commit)

git add : It is used to stage the files

```
git add <file-name> => Used for staging specific file

git add --a   ==> To stage all modified & new files
```

git commit : To commit staged files to Git local Repository.

```
git commit -m 'msg'
```

git remote add : To add Git Hub Repo URL to our path (Only firsttime we have to do)

git push : To push changes from Git Local repo to Git Hub Central Repository

```
git push -u origin main
```

git reset HEAD <file-name>:

git checkout <file-name> :

Log4J:

1. It is a Tracing or Logging Tool used in Specially in Production Environment. It is used to
find success messages, information, warnings, errors in application while using it.

2. By Default any Message/Error will be printed on Console, which is temporary location, it
can show only few lines like last 100 lines.

3. To see all problems from beginning to till date use Log4j concept.

4. Log4J can write problems/error to File(.log), Database, Email, Network etc..

5. Log4J provides Error Details like Exception type, class and method with line number it
occured, Date and time also other information .

6. Log4J also supports writing errors to Console also.

7. Using System.out.println prints only on console, but not to file or any other memory.

Log4J has 3 components:

1. Logger (LOG) Object: This object must be created inside the class as a instance variable. It is
used to enable Log4J service to current class. If this object is not created in the class then Log4J
concpet will not applicable(will not work)for that class.

It has 5 priority methods with names

(along with order)

srno	Name	Method
1	DEBUG	debug(msg)
2	INFO	info(msg)
3	WARN	warn(msg)
4	ERROR	error(msg)
5	FATAL	fatal(msg)
-	OFF	-

a. debug(msg) : It prints a message with data. It is used to print a final result of process.
Like EmpId after saved is : 2362.

b. info(msg) : It is used to print a simple message. Like process state-I done, if block end.
Email sent etc..

c. warn(msg): It is used to print warning messages. Like Collection is not with Generic, local
variable not used, resource not closed etc...

d. error(msg): It is used to print Exceptions like NullPointer, ArrayIndex, SQLException etc..

e. fatal(mgs) : It indicates very high level problem. Like Server/DB Down, Connection
timeout, Network broken, Class Not Found etc...

OFF is used to disable Log4J concept in application. Log4J code need to be deleted.

---

2. Appender : It will provide details for "Where to print Message?". Means in what type of
memories, messages must be stored.

i. File use FileAppender   *****

ii. Database use JdbcAppender

iii. Email use SmtpAppender

iv. Console use ConsoleAppender

v. Network use TelnetAppender

In one Application we can use more than one appender also.

---

3. Layout : It provide the format of message to be printed.
  Possible layouts are:

i. Simple Layout : Print message as it is

ii. HTML Layout : Print message in HTML format(<html><body>....)

iii. XML Layout: Print message in XML format (<Errors><Type>..<Message>..)

iv. Pattern Layout : This class provides output pattern for a message that contains date, time,
class, method, line number, thread name, message etc..

In development mostly used Appender is FileAppender and Layout is Pattern Layout

a)Date&Time pattern
    %d = date and time
examples:
a) %d
b) %d {dd-MMM-yy hh:mm:ss SSS}
c) %d {dd-MM-yy hh:mm:ss SSS}
d) %d {HH:mm:ss}
e) Here meaning of every word used
f) in date pattern is,
g) dd = date
h) MMM = Month Name

i) MM = Month number
j) yy = Year last two digitis
k) yyy = Year in 4 digits
l) hh = Hours in 12 format
m) HH = Hours in 24 format
n) mm = Minutes
o) ss = Secs
p) SSS = MillSecs
    %C = Class Name
    %M = Method Name
    %m = Message
    %p = Priority method name(DEBUG,INFO..)
    %L = Line Number
    %l = Line number with Link
    %n = New Line(next line)
    %r = time in milli sec.
    %% = To print one '%' symbol.
    we can also use symbols like - [] , /
One Example Pattern with all matching "%d-%C[%M] : {%p}=%m<%L> %n "

Example :

```
package com.bikkadIT;

import java.io.IOException;

import org.apache.log4j.Appender;
import org.apache.log4j.ConsoleAppender;
import org.apache.log4j.FileAppender;
import org.apache.log4j.HTMLLayout;
import org.apache.log4j.Layout;
import org.apache.log4j.Logger;
import org.apache.log4j.PatternLayout;
import org.apache.log4j.SimpleLayout;
import org.apache.log4j.xml.XMLLayout;

public class TestApp {

        private static Logger log = Logger.getLogger(TestApp.class);

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

```
//		Layout layout = new SimpleLayout();
		Layout layout = new HTMLLayout();
//		Layout layout = new XMLLayout();
//		Layout layout=new PatternLayout("%p %d %C %M %m %n");
//		Layout layout=new
PatternLayout("[%p]--- %d{HH:mm:ss} %C %M %m %L %l   %n");

		//Appender app = new
FileAppender(layout,"C:\\Users\\HP\\Desktop\\Messages.log");
		Appender app=new ConsoleAppender(layout);
		log.addAppender(app);
		log.debug("This is debug method");
		log.info("This is info method");
		log.warn("Ths is warn method");
		log.error("This is error method");
		log.fatal("This is fatal error");
	}

}
```

1) What is Logging?

-> It is the process of storing application execution details

2) Why we need logging?

-> In Realtime, our application will be deployed into multiple Environments like DEV, QA, UAT, PILOT and PROD.

-> When application running in those environments, if any functionality not working then it is very difficult to understand the problem.

-> To trace the problems in application we will use Logging.

3) What are the Components available in Logging?

-> To work with logging we will use below 3 components

1) Logger

2) Layout
3) Appender

-> Logger is the class which provided methods to generate log messages based on log levels

-> Layout will re-present format of the log message (what information should be there in log msg)

-> Appender will represent destination of log message (Where log msg should be printed)

-> We are having below appenders

i) ConsoleAppender
ii) FileAppender
iii) JdbcAppender
iv) SMTPAppender etc...

4) What are the log levels available?

TRACE, DEBUG, INFO, WARN, ERROR, FATAL

-> If we set one log level, then log messages will be printed from that level to all higher levels.

-> If we set log level as INFO, then it will print INFO msgs, WARN msgs, ERROR msgs, FATAL msgs

----

-> Generally we have to write some configuration for logging. Earlier people used to write that configuration using log4j.properties

-> If we are developing our application using Spring Boot, it will provide default configuration for logging.

Layout -> PatternLayout
Appender -> ConsoleAppender

Note: Spring Boot will perform logging using logback (Default behaviour)

-> If required we can customize logging configuration in Spring Boot application.

Example :

```
package com.bikkadIt.controller;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class WelcomeRestController {

	private static Logger
logger=LoggerFactory.getLogger(WelcomeRestController.class);

	@GetMapping("/welcome")
	public String welcomeMsg() {
		logger.debug("welocme() method execution started");
		String msg="Welcome to BikkadIt";
		logger.debug("welocme() method execution ended");
		logger.info("welcome() method executed successfully");
		return msg;
	}
}
```

```
server.port=9090

logging.level.root=INFO

logging.file.name=MyApp1.log
```

log4j.properties

log4j.properties file: This file is used to specify all the configuration details of Log4J. Especially
like Appender Details and Layout Details with Patterns and also root Logger details. This File
contains details in key=value format (.properties).

Data will be shown in below order like
1. rootLogger
2. appenders
3. layouts

a) In this file (log4j.properties) use symbol '#' to indicates comments.

b) We can specify multiple appenders in log4j.properties file Like 2 File Appenders, one
JdbcAppender, One SmtpAppender etc..

c) Every Appender must be connected with layout

d) Appender name must be define before use at rootLogger level.

e) Appender name can be anything ex: abc,hello,sysout,file,db,email etc..

f) log4j.properties file must be created under src folder (normal project) or src/main/resource folder (maven project).

g) Make rootLogger = OFF to disable Log4J completely without deleting code in any class or
properties file

h) log4j.properties file will be auto detected by Log4J tool. No special coding is required.

Example : log4j.properties

```
log4j.rootLogger=INFO,stdout,,file
```

```
# Redirect log messages to console
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd} %p %C:%L
 %n

# Redirect log messages to a log file
log4j.appender.file=org.apache.log4j.FileAppender
log4j.appender.file.File=C:/Users/BR011TX/Desktop/Log/logfiles.log
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd
HH:mm:ss} %p %c:%L - %m%n
```

```
package com.bikkadIt.pro;
import org.apache.log4j.Logger;
public class Log4jPro {

		private static Logger log=Logger.getLogger(Log4jPro.class);
		public static void main(String[] args) {

				log.debug("This is debug message");
				log.info("This is info message");
				log.warn("This is warn message");
				log.error("THis is Error message");
				log.fatal("This is fatal message");
		}
}
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.bikkadIT</groupId>
   <artifactId>Log4JBasic</artifactId>
```

```
  <version>0.0.1-SNAPSHOT</version>


 <dependencies>
<dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.8</version>
</dependency>
</dependencies>

<properties>
      <maven.compiler.source>1.8</maven.compiler.source>
      <maven.compiler.target>1.8</maven.compiler.target>
   </properties>

</project>
```

How to implement Logging in Realtime Project

-> When the project deployed into production server, multiple users will access our application at a time.

-> When multiple users are accessing our application then there is a chance of getting some issues in our application.

-> To identify root cause of the issues we will use log messages.

-> As many users are accessing our application so many log messages will be printed.

-> In realtime we will not use only one log file to store application log messages because so many logs will be generated on daily basis.

Ex: facebook will be used by crores of ppl daily
Ex: gmail will be used by crores of ppl daily
Ex: flipkart & amazon will be used by crores of ppl daily

-> To resolve this issue we will use below technique

RollingFileAppender

-> RollingFileAppender we can implement in 2 ways

1) TimeBasedRolling
2) SizeBasedRolling

-> In order to implement TimeBasedRolling or SizeBasedRolling we will use logback.xml file in SpringBoot application.

-> logback is the default logging technique used by Spring Boot.

-> Keep logback.xml file inside src/main/resources folder

```
-------------------------logback.xml-----------------------------------------------------------------
----
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
	<appender name="ConsoleAppender"
		class="ch.qos.logback.core.ConsoleAppender">
		<encoder>
			<pattern>
				%d{MM:dd HH:mm:ss.SSS} [%t] [%level]
[%logger{36}] - %msg%n
			</pattern>
		</encoder>
	</appender>

	<appender name="RollingAppender"
		class="ch.qos.logback.core.rolling.RollingFileAppender">
		<file>app.log</file>
		<encoder

	class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
			<Pattern>%d{MM:dd HH:mm:ss.SSS} [%t] [%level]
[%logger{36}] - %msg%n
			</Pattern>
		</encoder>
		<rollingPolicy

	class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
	<file-MandiePattern>app_%d{dd-MM-yyyy}.log</file-MandiePattern
>
				<maxHistory>10</maxHistory>
				<totalSizeCap>10KB</totalSizeCap>
			</rollingPolicy>
	</appender>

	<!-- Logging custom package -->
	<logger name="com.bikkadIt" level="info" additivity="false">
		<appender-ref ref="ConsoleAppender"></appender-ref>
		<appender-ref ref="RollingAppender"></appender-ref>
	</logger>

	<!-- Logging spring boot package -->
	<logger name="org.springframework.boot" level="info"
		additivity="false">
		<appender-ref ref="ConsoleAppender"></appender-ref>
		<appender-ref ref="RollingAppender"></appender-ref>
	</logger>


	<root level="info">
		<appender-ref ref="ConsoleAppender"></appender-ref>
	</root>
</configuration>
```

e-Mandi:

e-Mandi is Central government project.

e-Mandi is electronics National agriculture market.

->In India   Every state contains state govt.

-> Both state govt and Central gov are very supportive for Indian Farmers.

-> Indian govt providing very good facilities for Indian Farmers in terms of Agriculture.

-> e-Mandi is a Central govt project.

-> e-Mandi   project is implemented for India country Farmers.

-> The main aim of e-Mandi project is used to provide 'One Nation One Market' for indian Farmers.
which will give below benefits:

1)Transparent Online Trading

2)Real Time Price for their produce.

3) Quality Certification

4) Logistics facility.

5) Warehousing(Commodity storage facility).

6) Online eAuction.

7)Online ePayment on Farmer Bank Account.

8) No corruption.

-> The Farmers who wants to sell their produce they have visit nearest Mandi(APMC) office.
In Mandi Office Mandi employee are there .When farmer will reach at mandi office entrance gate with
their produce then mandi employee will collect some basic information like aadhar card and bank details
from farmer   and they will register that farmer on e-Mandi portal.
After successful registration they will collect some additional information like commodity ,approx quantity
of commodity and mandi employee will do gate entry for particular farmer.
After successful gate entry Mandi Employee will give gate entry pass to farmer.
Then farmer will go to agent shop and farmer will put their produce on agent shop.
Then mandi employee will collect sample from produce and they will do samling and quality checking

of produce and upload quality certificate on e-Mandi portal.
Then all traders will come to agent shop and they will do eauction on e-Mandi android application.
Out of all trader who will give higest bid to farmer produce that trader will get that produce.
Then weighment porcess will be completed. and after that Trader will do online epayment to farmer.

In e-Mandi application, we have below modules

1) Administration

2) Gate Entry

3) Lot Management

4) Sampling/Assaying

5) e-Auction

6)Weighment

7)ePayment

8) Batches

9) Reports

10) Exit

National Agriculture Market (e-Mandi) is a pan-India electronic trading portal which
networks the existing APMC mandis to create a unified national market for agricultural commodities.

National Agriculture Market or e-Mandi is an online trading platform for agricultural commodities in India.

The market facilitates farmers, traders and buyers with online trading in commodities.
The market helps in better price discovery and providing facilities for smooth marketing of produce.

What is the main objective of e-NAM?
The main objectives of e-NAM are to promote uniformity in agriculture marketing,
remove information asymmetry in the market and promote real-time price discovery.

Benefits of Trading on e-Mandi

Transparent Online Trading. Real-Time Price Discovery. Better Price Realization For Producers.
Reduced Transaction Cost For Buyers.

How does an e-Mandi work?

e-Mandi basically increases the choice of the farmer when he brings his produce to the mandi for sale.
Local traders can bid for the produce, as also traders on the electronic platform sitting in other State/ Mandi.
The farmer may choose to accept either the local offer or the online offer.

Details Description of e-Mandi:

1) Administration

-> In Administration module we have below functionalities

1) Trader Account Management
    - Create Account
    - View Accounts
    - Edit Account & Update Account
    - Activate/De-Activate Account

1) Agent Account Management
    - Create Account
    - View Accounts
    - Edit Account & Update Account

- Activate/De-Activate Account

1) Mandi Employee Account Management
- Create Account
- View Accounts
- Edit Account & Update Account
- Activate/De-Activate Account

-> Mandi Secretory will provide Mandi Employee & Trader , Agent details to e-Mandi Admin.

-> e-Mandi Admin will create accounts for Mandi Employee & Trader and Agent then e-Mandi Application will send mail to Mandi Employee & Trader and Agent
with e-Mandi login details.

-> Mandi Employee will login e-Mandi application for managemnt of trading.

-> In Login page, Forgot Password hyperlink is available to recover the password.

Login Functionality

-> Mandi Employee & Trader , Agent enters Email and Password will click on 'Login' button

-> For login functionality application is going to check below conditions

a) Credentials are valid or not

b) Account is Active or not

c) Role of the user

2) Gate Entry :

->Gate Entry module is used to determine weather a Farmer can sell their produce on e-Mandi or not.

Note: Only Maharashtra state Farmers should sell their produce on e-Mandi.

The Farmers who wants to sell their produce they have visit nearest Mandi(APMC) office.
In Mandi Office Mandi employee are there .When farmer will reach at mandi office entrance gate with
their produce then mandi employee will collect some basic information like aadhar card and bank details
from farmer and they will register that farmer on e-Mandi portal.

-> As part of Farmer registration process, e-Mandi will communicate with AAdhar App to retrieve Farmer state name
based on citizen aadhar no.

-> If Farmer belongs to Maharashtra state then Farmer registration will be completed successfully and Farmer Reg No
will be generated for Farmer.

-> If Farmer not belongs to Maharashtra then Farmer registration will be stopped and Farmer can't sell their produce on e-Mandi.

After successful registration they will collect some additional information like commodity ,approx quantity
of commodity and mandi employee will do gate entry for particular farmer.

After successful gate entry Mandi Employee will give gate entry pass to farmer.

Requirement : Develop AAdhar-WEB apllication with below functionalities
----

1) AAdhar Enrollment : It will take Farmer data as input and it will generate and provides Aadhar NO.

input data
----
firstname
lastname
gender
dob
state-Mandie

output

AAdhar no

2) AAdhar Verification : It takes AAdhar No as input and it provides Farmer data.

Input

Aadhar No.

Output
----
firstname
lastname
gender
dob
state-Mandie

3) Lot Management :

In Lot Management we have to functionalites

1)Split
2)Merge

4) Sampling/Assaying:

In This modeule we have two Functionalities

1) Sample Creation
2) Produce quality Checking

5) e-Auction:
In this modeule we have below functionalities

1)bid creation
2)bid declaration
3)sale agreement

4)sale Bill

6)Weighment.

In this module farmer produce weighment will be done.

7)ePayment.

-> This module is responsible to send produce amount to the Farmer Bank Account who sold their produce on e-Mandi.

8) Batches:

-> Batch is a program which performs business operation on bulk of records.

-> Below are some of the usecases for batches

1) Sending sale bill to Farmers.
2) Sending sale Agreemnet   farmers.
3) Sending Bill Amount to all faremrs.

9) Reports:

-> This module is used to generate reports in e-Mandi.

Daily Report

Weekly Report

Monthly Report

Gate Entry Report

sale agreement Report

sale bill report Report

eAuction Report

Farmer Registraction Report.

Trader Registraction Report.

Agent Registraction Report.

10) Exit :

In this module Exit receipt pass will be given to farmer.