What is the difference b/w fallback profile and default profile and also child profile?

Ans) **Fallback Profile** :: The profile that executes when requested active profile is not available (If spring.profiles.active key related profile is not avaeible)
default profile :: The profile that executes when there no activate profile in the application. (If spring.profiles.active key is commented)
child profile/profile include :: The Profile that is included in application.properties by specifying the profile name to reuse keys and values specific profile in application.properties indirectly as fallback profile. (If spring.profiles.include is specified in application.properties file)

Can we keep multiple spring beans of same category/type(having common super class/implementing interface) in default profile?

Ans) No ,Not possible .. Conflicts /Ambiguity Problems may come

=>Spring /spring Boot gives org.springframework.beans.factory.NoUniqueBeanDefinitionException expected single matching bean but found 2: cds,tcp
will come more Dependent of same type found to inject to target bean

=> spring /spring boot gives "org.ef.beans.factory.NoSuchBeanDexception" if the given bean id spring bean is not available.

Q) what is difference b/w not adding @Profile for spring bean and adding @Profile("default") for spring bean?

Ans) If @Profile is not added for spring bean that will work all for profiles when profile are activated.. It will even work
If no profiles are activated
=> @Profile("default") based spring bean will work only when active profiles are not specified.

=>In real projects we keep DS classes, DAO classes in specific profile or in default profile becoz
persistence logic changes DB s/w to DB s/w or SQL queries are DB s/w dependent
=> In real projects , we do not keep controller, service classes in any profile becoz they are common
for all profiles and not dependent to any Db s/w.

==========
Runners in spring Boot
==========

=> Runners are java classes com spring beans of spring boot App implementing RxxRunner() directly or indirectly
=> Every runner class contains run()-) dealing with onetime executing logics and those logics will executes
when springApplication.run() is about to complete all its startup activies (right after creating ApplicationContext obj
and completing pre-installation and Injections)
=>The run()-) of every Runner class will be executed automatically by IOC container only for 1 time as part of
Application startup process that takes place in SpringApplication.run() method.
note: No Programmer prefers placing logics in the main() of main class.. they prefer placing same logics in the Runner classes..

What is the difference b/w static block and Runner with respect to Spring BootApp?

Ans)

static block of spring bean	Spring Bean writing Runner class
a) executes when IOC container loads this class for instatiation ..	a) run()-) this class will be called automatically by SpringApplication.run()-) as part of application startup activies
b) we can not pass inputs (args) while working with static block	b) we can get cmdline args as inputs in run()-) method as parameter values
c) static block allows only static variables/data to use	c) run()-) of Runner classes can deal with both static and non-static variables/ data
d) It is feature of java .. So can be used in spring and non-spring env..	d) feature of spring boot .. so can be used only in spring Boot application.. to place one time executing logics.
e) It is class level one time execution logic	e) It is spring boot application/project level one time executing logics

Two types of Runner in springboot
a) CommandLineRunner ()
b) ApplicationRunner ()
note:- Runners are co useful in in standalone, desktop Apps
Both these are two independent interfaces

What is the is the difference CommandLineRunner and ApplicationRunner?

Ans) Both the runners are giving run()-) as the callback method (becoz it will be called by
underlying IOC container automatically) .. Both runner's run()-) methods get
cmd line args(programs args) as the parmeter values .. but they the way they get them
is different.

org.springframework.boot.CommandLineRunner()
|--->public void run(String ...args)
var args that means we can pass our choice no.of args while calling the method
-var arg is internally array
gives cmd line args as raw values
(same as main()-) args[] or ...args)

org.springframework.boot.ApplicationRunner()
|--->public void run(ApplicationArguments args)
|-->allows to categorize and get cmd line arg values
as option args (start with --) and non-option args (only values)
from ApplicationArguments object.

methods of Application arguments..

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method and Description
boolean	containsOption(String name) Returns whether the set of option arguments parsed from the arguments contains an option with the given name.
List<String>	getNonOptionArgs() Returns the collection of non-option arguments parsed.
Set<String>	getOptionNames() Returns the names of all option arguments.
List<String>	getOptionValues(String name) Returns the collection of values associated with the arguments option having the given name.
String[]	getSourceArgs() Returns the raw unprocessed arguments that were passed to the application.

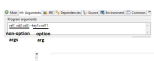
=>Both are Functional Interfaces (java8) becoz both of them contains only one method declaration
(run()-) method)
=>The Interface that contains only one abstract method declaration dircty or indirectly is called functional interface
To mark interface as functional interface we use @FunctionalInterface .

What is the use of Runners in spring Boot App? (Necessary)

Ans) =>we generally place intial logics or startup logics to execute in the run()-) method of Runner class
=>The logics are like
=>Executing DB script and keeping DB tables with rows ready
=>Performing some testing activities related to AutoConfiguration and its injections
=> Logging activies related to AutoConfiguration and startup activities
=>To send email or SMS alertMessages to admin when imp services started
during the startup process of spring Boot Application .. start of server , Intiialization of
SMS Gateway, PaymentGateway and etc... s
=>Though There are multiple modules in Project.. If a want to activate only certain modules thods
we take the support of Runners
=> To create dummy /admin email ids , dummy or admin accounts during the startup process
automatically we take support of runners
and etc...

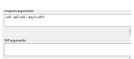
note: In realtime projects the main class (configuration class main class that contains @SpringBootApplication) main()-) method contain
only SpringApplication.run()-) method call.. The remaing all logics like getting spring Bean class object IOC container ,invoking b.me
and etc.. will be done from the run()-) methods of runner classes through various injections.

```
//SMSAlertServiceRunner.java
package com.st.runners;
import java.util.Arrays;
import org.springframework.boot.ApplicationArguments;
import org.springframework.boot.ApplicationRunner;
import org.springframework.stereotype.Component;

@Component
public class SMSAlertServiceRunner implements ApplicationRunner {

    public SMSAlertServiceRunner(){
        System.out.println("SMSAlertServiceRunner:: 0-param constructor");
    }

    @Override
    public void run(ApplicationArguments args) throws Exception {
        System.out.println("SMSAlertServiceRunner.run()");
        System.out.println("non-option args"+args.getNonOptionArgs());  //gives non-option args
        System.out.println("option-names"+args.getOptionNames());  //gives the keys of option args
        System.out.println("option-args--values"+args.getOptionValues("key1"));  //gives the value of given option arg name/key
        System.out.println("all args --values"+Arrays.toString(args.getSourceArgs()));  //gives both option and non-option args
    }
}
```



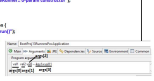
program args are commandline arg: like arg0,1,arg2,(1) and etc..
VM arg: gives to pass system properties/Application setting -d option
-Dspring.profiles.active=dev -Dusrname=raja -Dpassword=34mb

output
=====
SMSAlertServiceRunner.run()
non-option args:[val1, val2, val3]
option arg: key=[key1]
option arg--value:[val11]
all arg --values:[val1, val2, val3, --key1=val11]



note: CommandLine gives all cmdline args in sequence i.e
there is not provision to differentiate and get cmd args
as option arg (-key=val) and non -option arg(-val)

```
//AlterServiceRunner.java
package com.st.runners;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
@Component
public class AlterServiceRunner implements CommandLineRunner {

    public AlterServiceRunner(){
        System.out.println("AlterServiceRunner:: 0-param constructor");
    }

@Override
public void run(String... args) throws Exception {
  System.out.println("AlterServiceRunner.run()");
  System.out.println("cmd line args are : ");
  for(String arg:args) {
    System.out.println(arg);
  }
}//run()..
}//class

output
=====
AlterServiceRunner.run()
Cmd line args are :
val1
val2
val3
--key1=val11
```



note: if we container /server wants to generate InMemory Proxy classes dynamically
at run time in VM Memory of RAM .. then it internally uses "proxygen" memory of
JVM memory .. To increase that memory size use -DproxygenSize=24mb as System property
(Place in VM arguments)

=> If we place both ApplicationRunner and CommandLine Runner classes (This statement is give
in a spring boot project.. then ApplicationRunner class executes first w.r.t run()-) methods execution)
and CommandLineRunner class executes next..

=> Industry profers working with CommandLineRunner rather ApplicationRunners becoz
handling cmdline args is quite easy in CommandLineRunner classes..

=>*Runners really useful in projects of type web applications and Distributed App which runs in production env.. which executes for long time with out shutdown.. In those Apps if want to execute some startup logics or initialization logics then go for Runner Suppport.* Ex:- DB table creation with default records .. creating admin emaids and etc.)

What is the difference b/w starter and runner?

Ans) *starters are for files acting as libraries supporting spring boot's AutoconfigurationAutoconfiguration activities by collecting inputs from application.properties/yml* these are main jar files giving dependent, related jar files automatically through maven/gradle support

=>*Runners are spring beans having callback method (run()-) and contains application startup or initial logics as discussed in above use-cases.* These are one-time execution classes of spring app ,having startup logics

The methods that are called automatically by underlying IOC container or JVM are called callback methods..
note: main()-) is callback for JVM
note: servlet life cycle methods are callback methods

Adding Runners support in Layered Application
==========
step1) keep layered app ready

step2) place only SpringApplication.run()_) method call in main method of main class (@SpringBootApplication class)

```
public static void main(String[] args) {
    //get IOC container
    SpringApplication.run(BootProj01LayeredAppApplication.class, args);
}//main
```

step3) add Runner class injected controller spring bean obj through @Autowired and invoke b.methods in the run()_) method as shown below

```
//LayeredAppPostingRunner.java
package com.st.runner;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
import com.st.controller.PayrollOperationsController;
import com.st.model.Employee;

@Component
public class LayeredAppPostingRunner implements CommandLineRunner {
  @Autowired
  private PayrollOperationsController controller;

  public LayeredAppPostingRunner(){
    System.out.println("LayeredAppPostingRunner:: 0-param constructor");
  }

  @Override
  public void run(String... args) throws Exception {
    System.out.println("LayeredAppPostingRunner.run()");
    try {
      List<Employee> list=controller.showEmployeesByDesig("CLERK","MANAGER","SALESMAN");
      list.forEach(emp->{
        System.out.println(emp);
      });
    }//try
    catch(Exception e) {
      e.printStackTrace();
      System.err.println("Internal problem --try again !"+e.getMessage());
    }//catch

  }//run()

}//class

output
=====

PayrollOperationsController:: 0-param constructor
EmployeeAppServiceImpl:: 0-param constructor
EmployeeDAOImpl:: 0-param constructor
BootProj01LayeredAppApplication-started30[0]
2022-07-04 08:20:06.513  INFO 3724 --- [                  main] com.zxchange.v2.log.MLog                 : Mlog clients using java 1.4+ standard logging
2022-07-04 08:20:06.603  INFO 3724 --- [                  main] com.zxchange.v2.c3p0.C3P0Registry        : Initializing c3p0-0.9.1.2 [built 21-May-2007 15:04:56; debug? true; trace: 10]
LayeredAppSelectingRunner:: 0-param constructor
2022-07-04 08:20:06.799  INFO 3724 --- [                  main] com.st.BootProj01LayeredAppApplication   : Started BootProj01LayeredAppApplication in 1.951 seconds (JVM running for 2.429)
LayeredAppPostingRunner.run()
EmployeeAppServiceImpl:getEmployeesByDesig()
EmployeeAppServiceImpl:getEmployeesByDesig()
EmployeeDAOImpl:getEmployeesByDesig()--dc class name: class com.zxchange.v2.c3p0.ComboPooledDataSource
2022-07-04 08:20:07.003  INFO 3724 --- [                  main] c.m.v.c.AbstractPoolBackedDataSource     : Initializing c3p0 pool... com.zxchange.v2.c3p0.ComboPooledDataSource [ acquireIncrement -> 3, acquireRetryAttempts
-> 30, acquireRetryDelay -> 1000, autoCommitOnClose -> false, automaticTestTable -> null, breakAfterAcquireFailure -> false, checkoutTimeout -> 0, connectionCustomizerClassName -> null, connectionTesterClassName
-> com.zxchange.v2.c3p0.impl.DefaultConnectionTester, dataSourceName -> 3fgf1q1qqc8jx81rnxh1k|6c1k1k1c1, debugStackTraceForAllResourceCreation -> false, description -> null, driverClass ->
oracle.jdbc.driver.OracleDriver, factoryClassLocation -> null, forceIgnoreUnresolvedTransactions -> false, identityToken -> 3fgf1q1qqc8jx81rnxh1k|6c1k1k1c1, idleConnectionTestPeriod -> 0, initialPoolSize -> 10, jdbcUrl
-> jdbc:oracle:thin:@localhost:1521:xe, maxAdministrativeTaskTime -> 0, maxConnectionAge -> 0, maxIdleTime -> 0, maxIdleTimeExcessConnections -> 0, maxPoolSize -> 1000, maxStatements ->0,
maxStatementsPerConnection -> 0, minPoolSize -> 3, numHelperThreads -> 3, numThreadsAwaitingCheckoutDefaultUser -> 0, preferredTestQuery -> null, properties -> {user=******, password=******}, propertyCycle ->
0, testConnectionOnCheckin -> false, testConnectionOnCheckout -> false, unreturnedConnectionTimeout -> 0, usesTraditionalReflectiveProxies -> false ]
Employee(empno=7369, ename=SMITH, deptg=CLERK, salary=800.0, deptno=20)
Employee(empno=7521, ename=WARD, deptg=SALESMAN, salary=1250.0, deptno=30)
Employee(empno=7499, ename=ALLEN, deptg=SALESMAN, salary=1600.0, deptno=30)
```

What is the difference b/w @PostConstruct method in spring Bean and Runner class of Spring boot App?

Ans) **@PostConstruct method** is specific each spring bean ... It is one time execution block containing
some initialization logics, verification logics with respect each spring bean i.e this method
spring bean specific method..
=>Runner class spring boot app specific class .. Runners are application level one time execution
classes... These Runner classes contains application level one time executing logics.

@PostConstruct method , @PreDestroy method and run()_) method Runner classes
are callback methods.

method
Which method executes First? @PostConstruct and run()-) method of Runner class?

Ans) @PostConstruct method executes first becoz @PostConstruct executes right after spring bean instantiation and
injections ..where as runner will execute after completing all autoconfiguration, singlescope spring beans pre-instantiation
/life cycle operations, injections and etc.. other operations..

How to disable Certain runner class temporarily If multiple runner classes are placed in spring boot project?

Ans) Remove or comment the @Component annotation from runner class
that u do not want to execute.

Can I Run Runners in spring apps?
Ans) Yes .. becoz there SpringApplication.run()-) method in spring apps..
So runners are specific to Spring boot apps.

==Spring JDBC Videos ==
https://youtu.be/KcrmZnvTyw
https://youtu.be/ZLCIgIaTbg
https://youtu.be/A1Tw5L6YTA
https://youtu.be/kobgvh2vbEc
https://youtu.be/v4R3kRbqZM
https://youtu.be/NKBVk-7Ob0
https://youtu.be/ZbbbbbbbbWQ
https://youtu.be/mbnsHgnBc8
https://youtu.be/L3RFlnwxWVQ
https://youtu.be/VrHwRzfZoQ
https://youtu.be/FAfOGOoIRGc
https://youtu.be/9eKcuLaVKo
https://youtu.be/yHPRmdKw
https://youtu.be/5Y-CKLBUKvY