S

Profiles in spring boot

=====

=>Both spring, spring boot support profiles

=>We work with profiles while developing spring apps

using a)xml driven cfgs b) xml+ annotation driven cfgs c) 100% code driven cfgs d)spring boot

=>The setup required to execute the application in called environment.. This environment includes compiled code + DB s/w + server + DataSource +JRE/JVM and etc..

note: This enviroment is also called as profile .. These are like netflix, amazon prime profiles which keeps track various activities of user like movie watching history,watch list, recomandations and etc..

of

=> As part project developent to project production we need different environments/profiles like a) profile/env for development

b) profile /env for testing

c) profile /env for uat (Testing at client organization)

d) profile /env for production

Proj1- Dev

=> Proj1 code

=> mysql db s/w

=> tomcat server

=> apache dbcp2

Proj1- Testing

=>Proj1 code =>mysql Db s/w

=> tomcat server

=> C3PO DS

At software company (WIPRO-HYD)

(It can be done using Local setup

or Cloud setup)

↓ AWS/azure/gcp/...

uat=user acceptence test

Proj1 - UAT

=>proj1 code

note:: no.of profile are not fixed, there can be multiple like dev, test, uat, pre- prod, prod and etc..

Proj1- prod

release

=>Proj1 code

=> oracle db

=>oracle db

=> tomcat server

=> wildfly server

➡> oracle ucp

....

=> hikari cp

At client organization [Uses Local setup or Cloud setup)

=> In Spring boot Project/App we can create different prorperties files/yml files for different profiles and profile we can activate specific each time.. using application.properties/yml or using System property.

profile specific info

application-dev.properties/yml application-test.properties/yml application-uat.properties/yml application-prod.properties/yml application.properties/yml

This is required to activate each profile

(best)

Through there are multiple properites fiels/yml files we need not to configure them as custom properties

(CitiBank- USA)

AWS/azure/gcp/...

Dev

Development

UAT :: User Acceptence Test PROD:: Production

In Real Pratice, we take separate machine for every profile/environment

=> The physical machine(Local setup) or Virtual machine(cloud setup) of "Dev" profile will be connected with developers

=> The physical machine(Local setup) or Virtual machine(cloud setup) of "Test" profile will be connected with Testers

=> The physical machine(Local setup) or Virtual machine(cloud setup) of "UAT" profile will be connected with UAT Testers

=> The physical machine(Local setup) or Virtual machine(cloud setup) of "PROD" profile will be connected with client org endusers, employees and project maintainance team

=> jdbc properties, data soruce type, DAO classes, servers, server ports and etc. will change profile profile where as service class, controller classes and etc.. are same

in all profiles in most of the cases

files or yml files using @PropertySource as long as we follow the naming convention

note:: in Yml, we can place multiple profiles information

in single yml file by seperating the profiles with "---"

we can can activate each profile by using one of the two ways

a) using application.properites/yml file (Best)

spring.profiles.active=dev (for properites file)

(for yml file)

(application-<profile_name>.properties/yml)

note: taking names like dev-application.properties/yml is invalid.. the only allowed is application-<profile>.properites/yml

(or) spring: profiles:

active: dev

b) Using System property

(Not recomanded)

-Dspring.profiles.active=dev

(user-defined System properties are alternate to command line args)

note: Should be given in eclipse IDE

as VM arguments using Run Configurations.

note: if the specified active profile related properties/yml file is not avaiable then it takes application.properties/yml file info as default fallback information.

for

=> To make stereo type annotation based spring bean java class working certain profile we need to place @Profile("-") on the top of spring bean,

@Repository("oraEmpDao")

@Profile({"uat", "prod" })

public class Oracle Emloyee DAOImpl implements IEmployeeDAO{

}

@Repository("ms EmpDao")

@Profile({"dev","test"})

public class MySqlEmloyee DAOImpl implements IEmployeeDAO{

The object for spring bean class

will be created only when that profile is activated.

}

=>To make @Bean method generated spring bean belonging to certain profile .. we can add @Profile on top of @Bean method in @Configuration class

@Bean(name="c3p0Ds")

@Profile("test")

public DataSource createC3PODs(){

}

of

note:: if no @Profile(-) is placed on top stereotype annotations based spring bean or @Bean methods

then that spring bean works for all profiles. generally we do this for service. controller classes

Conclusion

=> To keep autoconfiguration inputs using common properties specific to certain profile then

keep those common properties info of autoconfiguration in application-<profile-name>.properties/yml file (profile specific properites file)

(jdbc proeprties, data Source type,server port number and etc..)

=>To keep user-defined java class as spring bean in certain profile then place @Profile(-) with profile name on the top

of user-defined spring class definitation (DAO classes)

=> To keep pre-defined java class as spring bean in certain profile then place @Profile(-) with profile name on the top of @Bean method where that spring bean is defined. (eg: DataSource Spring Bean cfg)

=> To activate certain profile give instruction from the base application.properties/yml file

Proj1- Testing

Example App

===========

Proj1- Dev

=> Proj1 code

=> mysql db s/w

=> apache dbcp2

=> C3PO DS

=>Proj1 code =>mysql Db s/w

release

Proj1 - UAT =>proj1 code

to

=> oracle db

→> oracle ucp

Proj1-prod =>Proj1 code

=>oracle db

=> hikari cp

step1) keep the MiniProject ready to add the profiles

make sure that u have taken two different DAO classes

1 for Oracle DB logics

2 for MySQL DB logics

In spring boot app we can work with any standalone datasource

and its jdbc con pool by doing the following opeations

a) add that datasource jar file to classpath using pom.xml dependency

b) specify that data source related class name in application.properties/yml file using the key "spring.datasource.type"

datasource

jar file

Tomcat cp apache dbcp C3PO

step3) empty the application.properties file for the time being..

step4) add multiple application-<profile>.properties files for multiple profiles/environments

on 1 per profile/environment basis

#src/main/resources

application.properties

application-dev.properties

Base properties file

application-prod.properties Profiles specific properties file

application-test.properties

application-uat.properties

MySQLEmployeeDAOImpl.java

@Repository("mysqlEmpDAO")

tomcat-jdbc-<ver>.jar

commons-dbcp2-<ver>.jar c3p0-<ver>.jar

public class MySQLEmployee DAOImpl implements IEmployeeDAO {

//SQL Query

data source class name

org.apache.tomcat.jdbc.pool.DataSource org.apache.commons.dbcp2.BasicDataSource

com.mchange.v2.c3p0.Combo Pooled DataSource

private static final String GET_EMPS_BY_DESGS="SELECT EMPNO,ENAME,JOB,SAL, DEPTNO FROM

@Autowired

private DataSource ds;

EMPLOYEE WHERE JOB IN(?,?,?) ORDER BY JOB";

application-dev.properties

```
#jdbc properties
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql:///NTSPBMS616DB
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.dbcp2.max-total=100
spring.datasource.dbcp2.initial-size=10
spring.datasource.dbcp2.max-conn-lifetime-millis-100000
```

optional

```
#Specify the DataSource type
spring.datasource.type=org.apache.commons.dbcp2.BasicDataSource
```

(apache dbcp2)

application-test.properties

```
#jdbc properties
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql:///NTSPBMS616DB
spring.datasource.username=root
spring.datasource.password=root
c3P0.minSize=10
optional
c3P0.maxsize=100
#Specify the DataSource type
spring.datasource.type=com.mchange.v2.c3p0.ComboPooled DataSource
(c3P0)
application-uat.properties
#jdbc properties
spring.datasource.driver-class-name=oracle.jdbc.driver.Oracle Driver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
spring.datasource.username=system
spring.datasource.password=tiger
spring.datasource.oracleucp.max-pool-size=100
spring.datasource.oracleucp.min-pool-size=10
spring.datasource.oracleucp.time-to-live-connection-timeout=100000
#Specify the DataSource type
optional
spring.datasource.type=oracle.ucp.jdbc.PoolDataSourceImpl (oracle ucp)
application-prod.properties
#jdbc properties
}
Oracle EmployeeDAOImpl.java
@Repository("oraEmpDAO")
public class Oracle Employee DAOImpl implements IEmployeeDAO { //SQL Query
private static final String GET_EMPS_BY_DESGS="SELECT EMPNO,ENAME,JOB,SAL, DEPTNO FROM EMP
WHERE JOB IN(?,?,?) ORDER BY JOB";
@Autowired
private DataSource ds;
spring.datasource.driver-class-name-oracle.jdbc.driver.Oracle Driver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
spring.datasource.username=system
spring.datasource.password=tiger
```

spring.datasource.hikari.maximum-pool-size=100

spring.datasource.hikari.minimum-idle=10

Optional

spring.datasource.hikari.keepalive-time=100000

note: Since No DataSource class is configured here the HikariCP realated HikariDataSource class object will be created and activated

step4) makes sure that hikaricp, apache dbcp2, oracle ucp, c3p0, mysql jdbc driver and oracle thin jdbc driver

jar files are added to CLASSPATH

In pom.xml

step5)

<dependency>

<groupId>com.mysql</groupId>

mysql jdbc driver

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

tomcat cp

<!-- https://mvnrepository.com/artifact/org.apache.tomcat/tomcat-jdbc -->

<dependency>

<groupId>org.apache.tomcat</groupId>

<artifactId>tomcat-jdbc</artifactId>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2 --> <dependency>

<groupId>org.apache.commons</groupId>

apache dpcp2

<artifactId>commons-dbcp2</artifactId>

</dependency>

oracle

UCP

C3PO

<!-- https://mvnrepository.com/artifact/com.oracle.database.jdbc/ucp -->

<dependency>

<groupId>com.oracle.database.jdbc</groupId>

<artifactId>ucp</artifactId>

</dependency>

<!-- https://mvnrepository.com/artifact/com.mchange/c3p0-->

<groupId>com.mchange</groupId>

<dependency>

<artifactId>c3p0</artifactId>

<version>0.9.5.5</version>

</dependency>

<dependency>

oracle jdbc driver

<groupId>com.oracle.database.jdbc</groupId>

<artifactId>ojdbc8</artifactId>

<scope>runtime</scope>

</dependency>

makes sure that c3PO DataSource is created as the spring bean using @Bean method in @SBA class

In main class

note:: doing this is optional, if c3p0 related Combo PooledDataSource class is configured as as the data source type in application-test.properties file (refer application-test.properties file)

Not at all

required

@Autowired

private Environment env;

@Bean(name="c3PODs")

public ComboPooled DataSource createC3PODs() throws Exception { ComboPooledDataSource cds=new ComboPooledDataSource();

cds.setJdbcUrl(env.getProperty("spring.datasource.url"));

cds.setDriverClass(env.getProperty("spring.datasource.driver-class-name"));

cds.setUser(env.getProperty("spring.datasource.username"));

cds.setPassword(env.getProperty("spring.datasource.password"));

cds.setMinPoolSize(Integer.parseInt(env.getProperty("c3PO.minSize")));

cds.setMaxPoolSize(Integer.parseInt(env.getProperty("c3P0.maxSize")));

return cds;

}

step6) keep @Profile annotation in the following places

a) MySqlDAO class should execute in dev,test environment,So add

@Profile({"dev","test"}) on the top of MySqlDAOImpl class

@Repository("empDAO-mysql")

@Profile({"dev","test"})

public class MySQLEmployee DAOImpl implements IEmployeeDAO {

private static final String INSERT_EMPLOYEE_QUERY="INSERT INTO

}

@Autowired

private DataSource ds;

Optional here

EMPLOYEE_INFO(ENAME,DESG,SALARY,GROSS_SALARY,NET_SALARY) VALUES(?,?,?,?,?)";

b) Make OracleDAO class beloging "uat", "prod" profiles

So add @Profile("uat","prod") on the top of OracleDAOImpl class

@Repository("empDAO-oracle")

@Profile({"uat", "prod"})

public class Oracle EmployeeDAOImpl implements IEmployeeDAO {

private static final String INSERT_EMPLOYEE_QUERY="INSERT INTO EMPLOYEE_INFO VALUES(EID_SEQ1.NEXTVAL,?,?,?,?,?)";

@Autowired

private DataSource ds;

}

c) keep @Bean method based Combo Pooled DataSource object (c3PO) working only for "test" profile

using @Profile("test") on the top of its @Bean method

(not required in spring boot 3.x)

@Bean(name="c3PODs")

not required in spring boot 3.x

@Profile("test")

public ComboPooledDataSource createC3PODs() throws Exception {

System.out.println("BootProj03 LayeredAppRealtimeDiApplication.createC3PODs()");

ComboPooledDataSource cds=new ComboPooledDataSource();

cds.setDriverClass(env.getProperty("spring.datasource.driver-class-name"));

cds.setJdbcUrl(env.getProperty("spring.datasource.url"));

cds.setUser(env.getProperty("spring.datasource.username"));

cds.setPassword(env.getProperty("spring.datasource.password"));

cds.setMinPoolSize(Integer.parseInt(env.getProperty("c3P0.minSize")));

cds.setMaxPoolSize(Integer.parseInt(env.getProperty("c3P0.maxSize")));

return cds;

}

step7) activate specific profile from the application.properties file

application.properties

#Activate the profile spring.profiles.active=dev

step8) Run the application..

uat

test we can activate one profile at a time

prod

Optional

Q) if one target class is having multiple possible dependents to Inject then how

can we inject specific dependent of our choice with out distrubing the

source code of the Application/Project? (Loose coupling is required here)

Ans) It can be in two ways

bdon

a) Using bean aliasing and @Qualifier (This demands the spring bean cfg file(xml) utilization, So not recomanded) (old style)

b) using spring boot profiles (Best and recomanded)

How can we see that activated profile while running the spring boot Project?

Ans) In the log messages of the Project

n: Starting BootIocProj11LayeredAppMiniProjectA

n: The following 1 profile is active: "test"

(or)

Using the Environment object

// Get Environment object

Environment env=ctx.getEnvironment();

System.out.println("Activate profile ::" +Arrays.toString(env.getActiveProfiles()));

How to remove Block commenting in Eclipse IDE?

Ans)

window menu ----> preferences ----> search for formatting ----> new ---> name: n1

-->expand comments ----> deselect

Enable Javadoc comment formatting

Enable block comment formatting

--->apply ----> ok