Date : 18/09/2020

Spring Boot 7AM

Mr. RAGHU

--------------------------------------------

**Spring Boot Profiles- PART-2**

Profiles Fallback:-

=> Every properties file used in profiles concept must have same keys and all keys.

=> If any activated profile(not default) has no key named 'x', then Spring boot will

search for same in default properties file.

\*) If we move from one Env to another Env, then all keys may not be modified,

only few are going be chaged with new values. In that case this concept helps us to

avoid writing duplicates from default.

\*\* If key is not present everywhere then

@Value -- Exception

@ConfigurationProperties- default value

--Example---

#1. Create one Starter project

Name: SpringBoot2ProfilesFallback

Dep : Lombok

#2. application.properties

#app.db=DEV-ORCL

app.pwd=DEV-PWD

application-qa.properties

#app.db=QA-ORCL

app.pwd=QA-PWD

application-prod.properties

app.db=PROD-ORCL

app.pwd=PROD-PWD

#3. Runner

package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.stereotype.Component;

import lombok.Data;

@Component

@Data

@ConfigurationProperties(prefix = "app")

public class DataReaderRunner implements CommandLineRunner {

//@Value("${app.db}")

private String db;

//@Value("${app.pwd}")

private String pwd;

@Override

public void run(String... args) throws Exception {

System.out.println(this);

}

}

========================================================================

@Profile Annotation:-

One Specification (interface) may have multiple implementations(classes).

But in Spring/Spring boot, at a time only one impl class object only considered

for execution.

In above case, execute selected Impl class at runtime using @Profile annotation.

Even @Profile supports maintaing properties files even for all Envs.

--code--

1. create one starter project

Name : SpringBoot2ProfileAnno

Dep : Lombok

2. Properties files

--application.properties--

my.app.service=DEFAULT-SMS

my.app.fmt=DEFAULT-TXT

--application-email.properties-

my.app.service=EMAIL-GMAIL

my.app.fmt=EMAIL-MIME

3. Service Interface

package in.nareshit.raghu.service;

public interface AlertSystem {

public void sendMessage();

}

4. Impl classes

package in.nareshit.raghu.service.impl;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.context.annotation.Profile;

import org.springframework.stereotype.Component;

import in.nareshit.raghu.service.AlertSystem;

import lombok.Data;

@Component

@Profile("default")

@ConfigurationProperties(prefix = "my.app")

@Data

public class SmsAlertSystem implements AlertSystem {

private String service;

private String fmt;

@Override

public void sendMessage() {

System.out.println("FROM SMS SYSTEM-"+ service +"-"+ fmt);

}

}

------------

package in.nareshit.raghu.service.impl;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.context.annotation.Profile;

import org.springframework.stereotype.Component;

import in.nareshit.raghu.service.AlertSystem;

import lombok.Data;

@Component

@Profile("email")

@ConfigurationProperties(prefix = "my.app")

@Data

public class EmailAlertSystem implements AlertSystem {

private String service;

private String fmt;

@Override

public void sendMessage() {

System.out.println("FROM EMAIL SYSTEM-" + service +"-" +fmt);

}

}

5.Runner

package in.nareshit.raghu;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import in.nareshit.raghu.service.AlertSystem;

@Component

public class TestProfileAnno implements CommandLineRunner {

@Autowired

private AlertSystem als;

@Override

public void run(String... args) throws Exception {

als.sendMessage();

}

}

\*) Execution: --spring.profiles.active=email

======================

\*)Note:

@Component

class A{} -- A object always in every profile

@Component

@Profile("qa")

class B{} -- B object created only for qa profile

@Component

@Profile("default")

class C{} -- C object created only for default.

\*) If we did not provide @Profile then our code never comes under default.

Then our code executed in every profile/env.

\*) default concept is applied only for properties/yaml file, not for code.

Q1) --spring.profiles.active=default, which class objects are created?

A) A, C

Q2) --spring.profiles.active=qa, which class objects are created?

A) A, B

Q3) which class objects are created? [we did not give any profile name]

A) A, C

Q) If we write @Profile annotation, Do we need to write properties file as well ?

A) Not required, that depends on you code/requirements.

Q) What is the difference b/w @Qualifier and @Profile , @Primary ?

A)

https://www.youtube.com/watch?v=xusQhpQuODk&list=PLVlQHNRLflP9XSWeY4x4FLwnL3UOIxnTr&index=12

https://www.youtube.com/watch?v=-FlszP92JVM&list=PLVlQHNRLflP9XSWeY4x4FLwnL3UOIxnTr&index=13

---------------------------------------------------------------------------------------------

Child Profiles | Multi-profiles

Profiles using YAML