Video 10: Spring boot @Profile Annotation | How Profiling works in Spring Boot.

Suppose you have 2 applications and 1 common code base.

code base → mese-lite.

Application → Promise
↳ Sourcing

How will You make sure bean is created only for 1 Application, not for other?

1→ @ConditionalOnProperty [Already seen]

2→ @Profile → [we will discuss this]

Technically Yes @Profile can acheive this requirement.

But it is used more for $\boxed{\text{environment separation}}$ perspective rather than application separation.

Lets for example consider configurations. Like Database Connections.

Dev → (Dev Username, Dev Password)

Qa → (Qa UserName, Qa Password)

Q. We put configurations in "application.properties" file. But how to handle different env configurations. ?

That's where Profiling comes into the Picture

$$\begin{tikzcd} & \text{application.properties} \arrow[dl] \arrow[d] \arrow[dr] & \\ \text{application-\{profile1\}.properties} \arrow[d] & \text{application-\{profile2\}.properties} & \text{application-\{profile3\}.properties} \\ \text{Now this profile is nothing but env in Spring} & & \end{tikzcd}$$

```
{application-dev.properties}
username = devUsername
password = devPassword
```

```
{application-prod.properties}
username = prodUsername
password = prodPassword
```

```
{application.properties}
username = defaultUsername
password = defaultPassword.
```

→ No profile is set, so default config is picked up

```
@Component
public class MySQLConnection{

    @Value("${username}")
    String username;

    @Value ("${password}")
    String password;

    @PostConstruct
    public void init(){

        sout( username + password);
               ⇓            ⇓
        defaultUsername   defaultPassword
    }
}
```

→ these two properties are to be picked up from config file.

SpringBoot checks which file to pick. It checks for @Profile. No such thing found. Pick default {application.properties} ⇒ (No profile)

Q) How do we choose which application.properties file to choose..

Using "spring.profiles.active"

tells Spring which Profile You are currently working at

```
application.properties
username = defaultUsername
password = defaultPassword
spring.profiles.active = qa
```

this is how Spring gets to know which one to use

application-dev.properties	application-qa.properties	application-prod.properties
Username = devUsername	username = qaUsername	username = prodUsername
Password = devPassword	password = qaPassword	password = prodPassword.

** If a property is missing in "qa", it will pick that property from parent (application.properties)

But the above example is static only, but I want to be able to change things <u>dynamically</u>. How can I do that?

{2 Ways of doing that}

① Using VM arguments

```
mvn spring-boot:run
-Dspring-boot.run.profiles=prod.
```

this wins priority over {application.properties}

Spring.profiles.active = "qa"

(2nd way is using pom.xml.)

pom.xml.

(command)

```
mvn spring-boot:run -P production
```

profile

```
<profiles>
    <profile>
        <id> local </id>
        <properties>
            <spring-boot.run.profiles> dev </spring-boot.run.profiles>
        </properties>
    </profile>
    <profile>
        <id> production </id>
        <properties>
            <spring-boot.run.profiles> prod </spring-boot.run.profiles>
        </properties>
    </profile>
</profiles>
```

{choose this profile of application.properties}

@Profile → [To be used for different environment and no Application]

Using @Profile annotation, we can tell spring boot, to create a bean only when a particular profile is set.

```
@Component
@Profile("prod")
public class MySQLConnection {

    @Value("${username}")
    String username;

    @Value("${password})
    String password;

    @PostConstruct
    public void init(){
        Sout(username + password);
    }

}
```

Now in application.properties we have set [profile=qa]

Spring while trying to create a bean of this class will check @Profile("prod") which is not matching ("qa") hence it, will not get initialised. (prod ≠ qa)

application.properties

```
username = defaultUsername
password = defaultPassword
spring.profiles.active = qa
```

⟹ [Here profile is set to qa]

application-dev-properties

```
Username = devUsername
password = devPassword
```

application-qa.properties

```
Username= qaUsername
password = qaPassword
```

application-prod.properties

```
Username = prodUsername
password = prodPassword.
```

But if we run using the maven command,

```
mvn spring-boot:run
-Dspring-boot.run.profiles=prod,
```

Then during Application startup logs will be like.

```
The following 1 profile is active: "prod"
```

(**)

We can set multiple profiles at a time.

```
[The following 2 profiles are active: "prod", "qa"]
```

application.properties

```
username = defaultUsername
password = defaultPassword
spring.profiles.active = prod,qa
```

```
@Component
@Profile ("prod")
public class MySQLConnection{

}
```

```
@Component
@Profile ("qa")
public class NoSQLConnection{

}
```

Both Beans will get created since profiles has both (prod,qa)

But Imp (application.properties) will be picked of qa. The Last one is the comma separated List of profiles