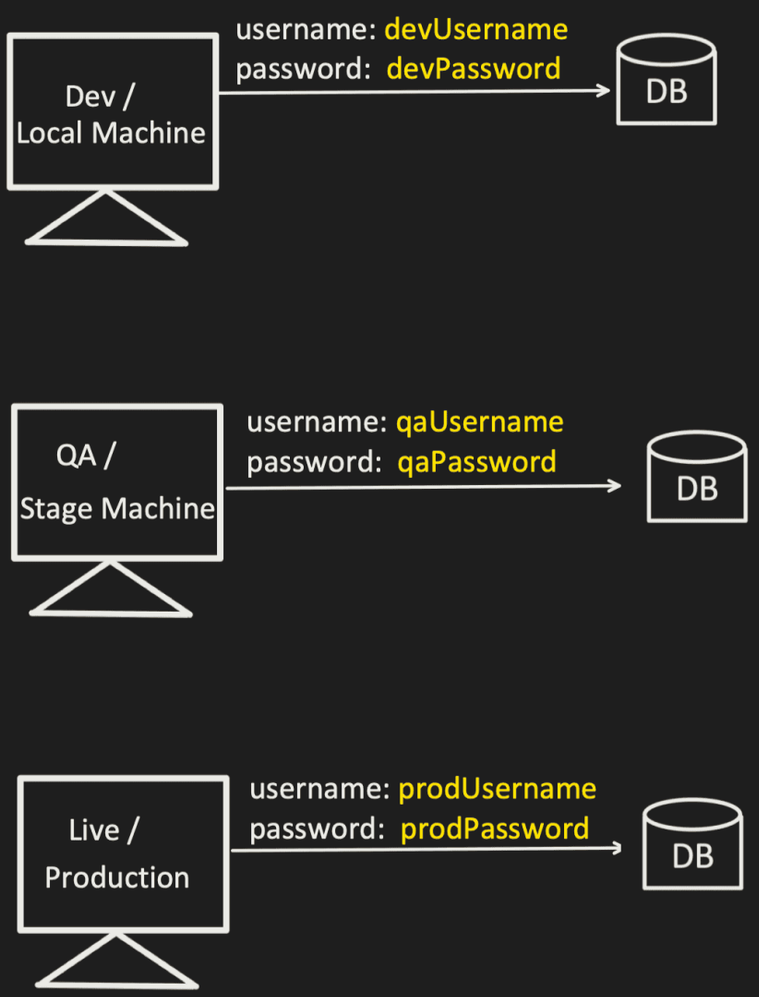
Let's deep dive into **@Profile** annotation to understand it better



This "username", "password" is just one example.
There are so many other configuration, which are different for different environments, like:

- URL and Port number
- Connection timeout values
- Request timeout values
- Throttle values
- Retry values etc.

How to do it?
We put the configurations in "application.properties" file.
But how to handle, different environment configurations?

That's where Profiling comes into the picture


```java
@Component
public class MySQLConnection {

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

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

    @PostConstruct
    public void init(){
        System.out.println("username: " + username + " passowrd: " + password);
    }
}
```

application.properties

```
username=defaultUsername
password=defaultPassword
```

application-dev.properties

```
username=devUsername
password=devPassword
```

application-qa.properties

```
username=qaUsername
password=qaPassword
```

application-prod.properties

```
username=prodUsername
password=prodPassword
```

```
2024-06-01T22:55:24.548+05:30  INFO 48119 --- [           main] c.c.l.SpringbootApplication
2024-06-01T22:55:24.549+05:30  INFO 48119 --- [           main] c.c.l.SpringbootApplication
2024-06-01T22:55:24.982+05:30  INFO 48119 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer
2024-06-01T22:55:24.990+05:30  INFO 48119 --- [           main] o.apache.catalina.core.StandardService
2024-06-01T22:55:24.990+05:30  INFO 48119 --- [           main] o.apache.catalina.core.StandardEngine
2024-06-01T22:55:25.016+05:30  INFO 48119 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]
2024-06-01T22:55:25.016+05:30  INFO 48119 --- [           main] w.s.c.ServletWebServerApplicationContext
username: defaultUsername passowrd: defaultPassword
2024-06-01T22:55:25.175+05:30  INFO 48119 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer
2024-06-01T22:55:25.179+05:30  INFO 48119 --- [           main] c.c.l.SpringbootApplication
```

And during application startup, we can tell spring boot to pick specific "application properties" file,
Using "spring.profiles.active" configuration.

```java
@Component
public class MySQLConnection {

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

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

    @PostConstruct
    public void init(){
        System.out.println("username: " + username + " passowrd: " + password);
    }
}
```

application.properties

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

application-dev.properties

```
username=devUsername
password=devPassword
```

application-qa.properties

```
username=qaUsername
password=qaPassword
```

application-prod.properties

```
username=prodUsername
password=prodPassword
```

```
2024-06-02T00:04:23.599+05:30  INFO 52175 --- [           main] c.c.l.SpringbootApplication              : The following 1 profile is active: "qa"
2024-06-02T00:04:24.002+05:30  INFO 52175 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2024-06-02T00:04:24.008+05:30  INFO 52175 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-06-02T00:04:24.008+05:30  INFO 52175 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.19]
2024-06-02T00:04:24.033+05:30  INFO 52175 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-06-02T00:04:24.033+05:30  INFO 52175 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 413 ms
username: qaUsername passowrd: qaPassword
2024-06-02T00:04:24.192+05:30  INFO 52175 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path ''
2024-06-02T00:04:24.198+05:30  INFO 52175 --- [           main] c.c.l.SpringbootApplication              : Started SpringbootApplication in 0.78 seconds (process running for 0.959)
```

We can pass the value to this configuration "spring.profiles.active" during application startup itself.

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

or

Add this in **Pom.xml**

```xml
<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>
    <profile>
        <id>stage</id>
        <properties>
            <spring-boot.run.profiles>qa</spring-boot.run.profiles>
        </properties>
    </profile>
</profiles>
```

```
mvn spring-boot:run -Pproduction
```

```
2024-06-02T00:21:57.336+05:30  INFO 53096 --- [           main] c.c.l.SpringbootApplication              : The following 1 profile is active: "prod"
2024-06-02T00:21:57.647+05:30  INFO 53096 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2024-06-02T00:21:57.652+05:30  INFO 53096 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-06-02T00:21:57.652+05:30  INFO 53096 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.19]
2024-06-02T00:21:57.673+05:30  INFO 53096 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-06-02T00:21:57.673+05:30  INFO 53096 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 318 ms
username: prodUsername passowrd: prodPassword
2024-06-02T00:21:57.804+05:30  INFO 53096 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path ''
```

So, now we know, what's Profiling is. Lets see what is @Profile annotation.

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

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

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

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

    @PostConstruct
    public void init(){
        System.out.println("MySql username: " + username + " passowrd: " + password);
    }
}
```

```java
@Component
@Profile("dev")
public class NoSQLConnection {

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

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

    @PostConstruct
    public void init(){
        System.out.println("NoSQL username: " + username + " passowrd: " + password);
    }
}
```

application.properties

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

application-dev.properties

```
username=devUsername
password=devPassword
```

application-qa.properties

```
username=qaUsername
password=qaPassword
```

application-prod.properties

```
username=prodUsername
password=prodPassword
```

mvn spring-boot:run

```
2024-06-02T00:38:46.935+05:30  INFO 55048 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2024-06-02T00:38:46.940+05:30  INFO 55048 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-06-02T00:38:46.941+05:30  INFO 55048 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.19]
2024-06-02T00:38:46.964+05:30  INFO 55048 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-06-02T00:38:46.964+05:30  INFO 55048 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 410 ms
2024-06-02T00:38:47.109+05:30  INFO 55048 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path ''
```

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

```
2024-06-02T00:42:30.245+05:30  INFO 55216 --- [           main] c.c.l.SpringbootApplication              : The following 1 profile is active: "prod"
2024-06-02T00:42:30.563+05:30  INFO 55216 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2024-06-02T00:42:30.568+05:30  INFO 55216 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-06-02T00:42:30.568+05:30  INFO 55216 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.19]
2024-06-02T00:42:30.591+05:30  INFO 55216 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-06-02T00:42:30.591+05:30  INFO 55216 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 327 ms
MySql username: prodUsername passowrd: prodPassword
2024-06-02T00:42:30.731+05:30  INFO 55216 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path ''
```

We can also set multiple profiles at a time.

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

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

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

    @PostConstruct
    public void init(){
        System.out.println("MySql username: " + username + " passowrd: " + password);
    }
}
```

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

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

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

    @PostConstruct
    public void init(){
        System.out.println("NoSQL username: " + username + " passowrd: " + password);
    }
}
```

application.properties

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

application-dev.properties

```
username=devUsername
password=devPassword
```

application-qa.properties

```
username=qaUsername
password=qaPassword
```

application-prod.properties

```
username=prodUsername
password=prodPassword
```

```
2024-06-02T00:45:36.744+05:30  INFO 56836 --- [           main] c.c.l.SpringbootApplication              : The following 2 profiles are active: "prod", "qa"
2024-06-02T00:45:37.149+05:30  INFO 56836 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2024-06-02T00:45:37.156+05:30  INFO 56836 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-06-02T00:45:37.156+05:30  INFO 56836 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.19]
2024-06-02T00:45:37.179+05:30  INFO 56836 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-06-02T00:45:37.179+05:30  INFO 56836 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 413 ms
MySql username: qaUsername passowrd: qaPassword
NoSQL username: qaUsername passowrd: qaPassword
2024-06-02T00:45:37.327+05:30  INFO 56836 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path ''
```

Now lets come back to the previous question, which I asked:

You have 2 Application and 1 common code base, how you will make sure that, BEAN is only created for 1 Application, not for other?

Common Codebase

```java
@Component
@Profile("app1")
public class NoSQLConnection {

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

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

    @PostConstruct
    public void init(){
        System.out.println("NoSQL username: " + username + " passowrd: " + password);
    }
}
```

Application1
(application.properties)

```
spring.profiles.active=app1
```

Application2
(application.properties)

```
spring.profiles.active=app2
```