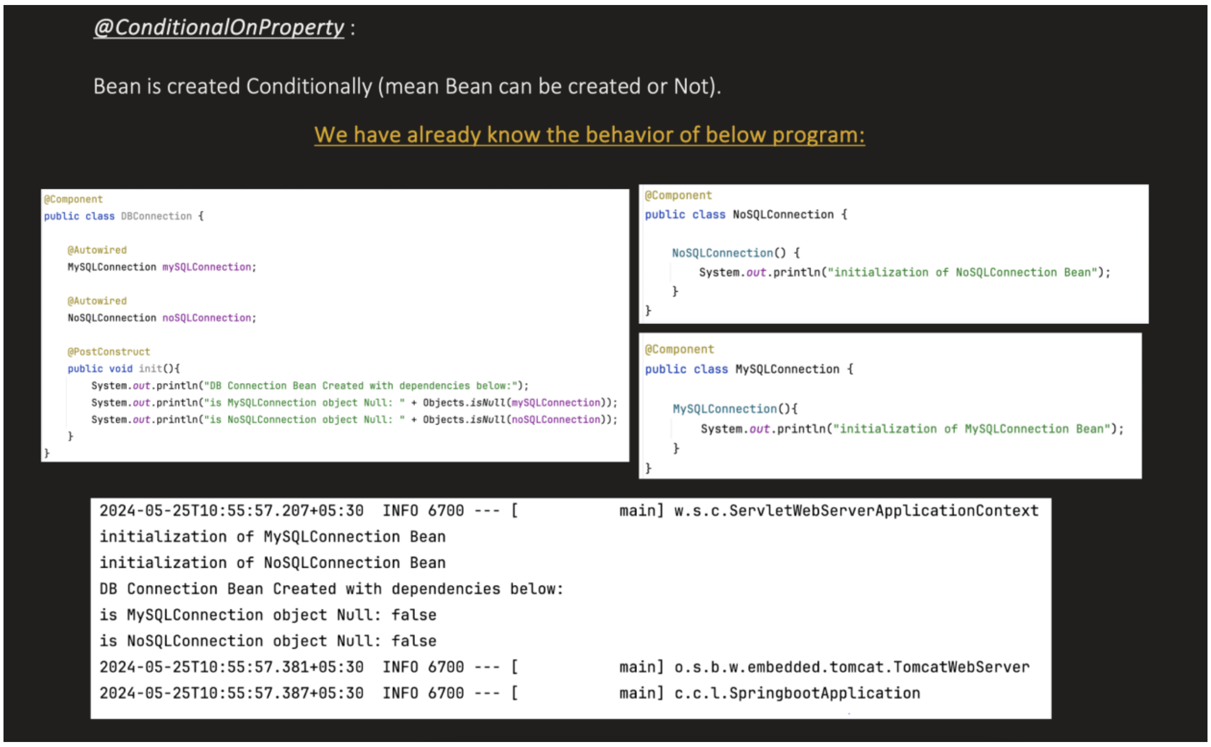
Spring boot: @ConditionalOnProperty



@ConditionalOnProperty :

Bean is created Conditionally (mean Bean can be created or Not).

We have already know the behavior of below program:

```java
@Component
public class DBConnection {

    @Autowired
    MySQLConnection mySQLConnection;

    @Autowired
    NoSQLConnection noSQLConnection;

    @PostConstruct
    public void init(){
        System.out.println("DB Connection Bean Created with dependencies below:");
        System.out.println("is MySQLConnection object Null: " + Objects.isNull(mySQLConnection));
        System.out.println("is NoSQLConnection object Null: " + Objects.isNull(noSQLConnection));
    }
}
```

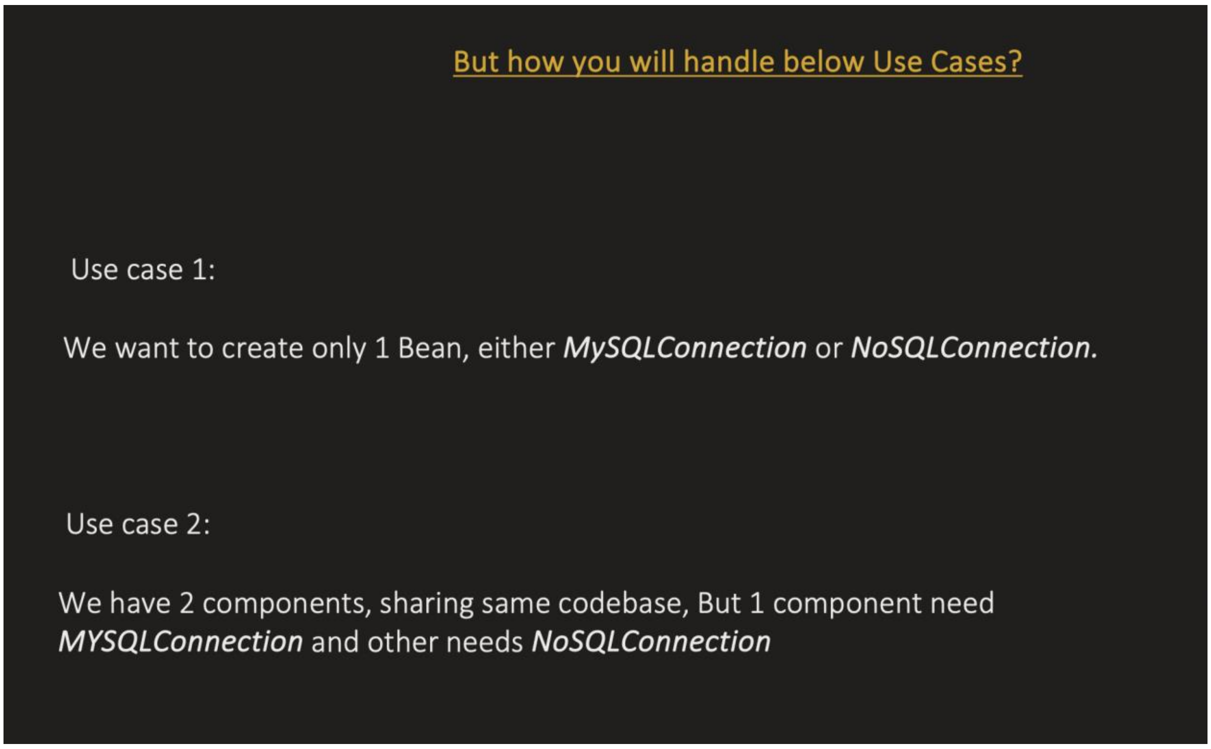
```java
@Component
public class NoSQLConnection {

    NoSQLConnection() {
        System.out.println("initialization of NoSQLConnection Bean");
    }
}
```

```java
@Component
public class MySQLConnection {

    MySQLConnection(){
        System.out.println("initialization of MySQLConnection Bean");
    }
}
```

```
2024-05-25T10:55:57.207+05:30  INFO 6700 --- [           main] w.s.c.ServletWebServerApplicationContext
initialization of MySQLConnection Bean
initialization of NoSQLConnection Bean
DB Connection Bean Created with dependencies below:
is MySQLConnection object Null: false
is NoSQLConnection object Null: false
2024-05-25T10:55:57.381+05:30  INFO 6700 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer
2024-05-25T10:55:57.387+05:30  INFO 6700 --- [           main] c.c.l.SpringbootApplication
```



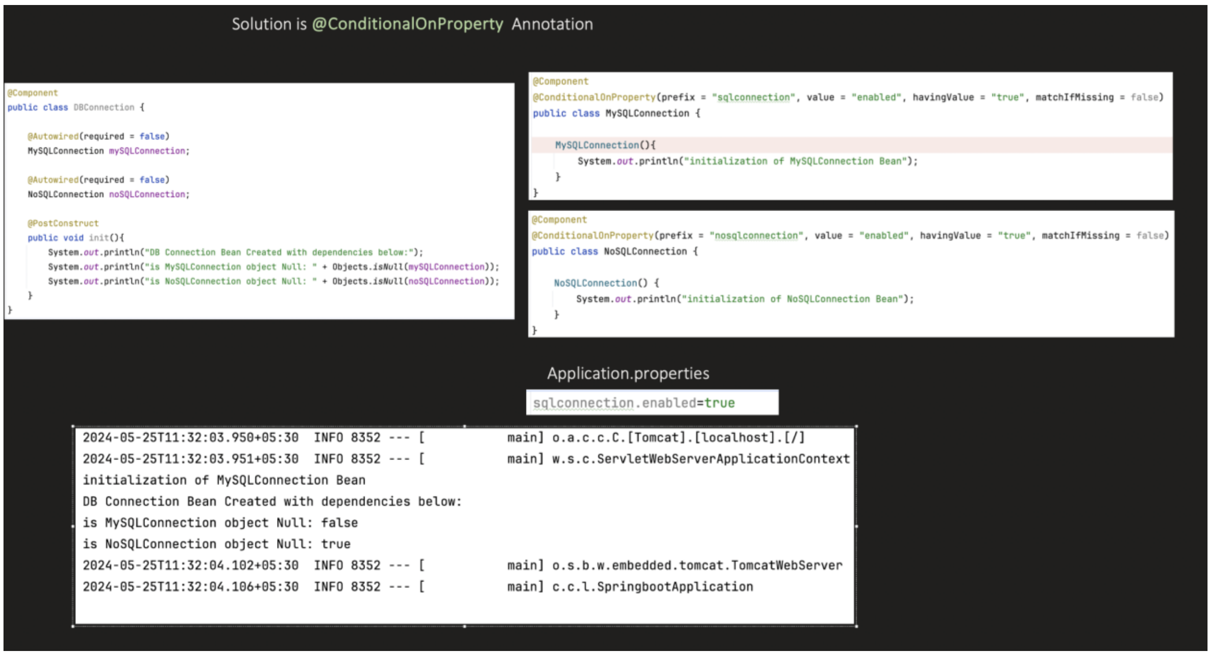
But how you will handle below Use Cases?

Use case 1:

We want to create only 1 Bean, either ***MySQLConnection*** or ***NoSQLConnection***.

Use case 2:

We have 2 components, sharing same codebase, But 1 component need *MYSQLConnection* and other needs *NoSQLConnection*



Solution is @ConditionalOnProperty Annotation

```java
@Component
public class DBConnection {

    @Autowired(required = false)
    MySQLConnection mySQLConnection;

    @Autowired(required = false)
    NoSQLConnection noSQLConnection;

    @PostConstruct
    public void init(){
        System.out.println("DB Connection Bean Created with dependencies below:");
        System.out.println("is MySQLConnection object Null: " + Objects.isNull(mySQLConnection));
        System.out.println("is NoSQLConnection object Null: " + Objects.isNull(noSQLConnection));
    }
}
```

```java
@Component
@ConditionalOnProperty(prefix = "sqlconnection", value = "enabled", havingValue = "true", matchIfMissing = false)
public class MySQLConnection {

    MySQLConnection(){
        System.out.println("initialization of MySQLConnection Bean");
    }
}
```

```java
@Component
@ConditionalOnProperty(prefix = "nosqlconnection", value = "enabled", havingValue = "true", matchIfMissing = false)
public class NoSQLConnection {

    NoSQLConnection() {
        System.out.println("initialization of NoSQLConnection Bean");
    }
}
```

Application.properties

```
sqlconnection.enabled=true
```

```
2024-05-25T11:32:03.950+05:30  INFO 8352 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]
2024-05-25T11:32:03.951+05:30  INFO 8352 --- [           main] w.s.c.ServletWebServerApplicationContext
initialization of MySQLConnection Bean
DB Connection Bean Created with dependencies below:
is MySQLConnection object Null: false
is NoSQLConnection object Null: true
2024-05-25T11:32:04.102+05:30  INFO 8352 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer
2024-05-25T11:32:04.106+05:30  INFO 8352 --- [           main] c.c.l.SpringbootApplication
```

Advantages:

1. Toggling of Feature
2. Avoid cluttering Application context with un-necessary beans.
3. Save Memory
4. Reduce application Startup time

Disadvantages:

1. Misconfiguration can happen.
2. Code Complexity when over used.
3. Multiple bean creation with same Configuration, brings confusion.
4. Complexity in managing.