10. Spring boot @ConditionalOnProperty Annotation

- In large Scale Application where there are thousands bean.
- And Many of those bean are initialized at the time of application startup.
- It will cluttered your application context with Un-necessary bean which is not required.

@ConditionalOnProperty

- Bean is Created Conditionally [mean Bean can be created or Not].

```
package com.springProject.SpringCourseProject.component;

import org.springframework.stereotype.Component;

@Component
public class NoSqlConnection {

    public NoSqlConnection(){
        System.out.println("NoSqlConnection Is Initialized");
    }
}
```

```
package com.springProject.SpringCourseProject.component;

import org.springframework.stereotype.Component;

@Component
public class MySqlConnection {
    public MySqlConnection(){
        System.out.println("MySqlConnection Is Initialized");
    }
}
```

```
package com.springProject.SpringCourseProject.component;

import jakarta.annotation.PostConstruct;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
```

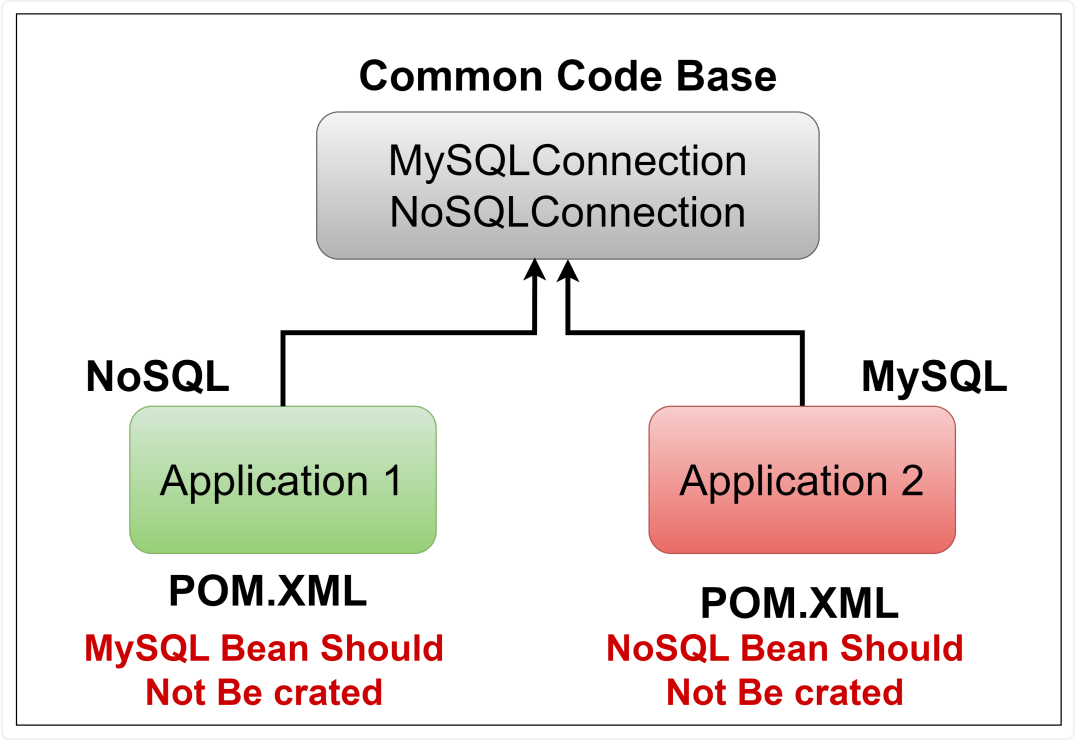
```
import java.util.Objects;

@Component
public class DBConnection {

    @Autowired
    MySqlConnection mySqlConnection;

    @Autowired
    NoSqlConnection noSqlConnection;

    public DBConnection(){
        System.out.println("DBConnection Initialized");
    }

    @PostConstruct
    public void init(){
        System.out.println("DBConnection BEAN create with below dependencies");
        System.out.println("Is MySqlConnection is NULL: "+
Objects.isNull(mySqlConnection));
        System.out.println("Is NoSqlConnection is NULL: "+
Objects.isNull(noSqlConnection));
    }
}
output
DBConnection Initialized
MySqlConnection Is Initialized
NoSqlConnection Is Initialized
DBConnection BEAN create with below dependencies
Is MySqlConnection is NULL: false
Is NoSqlConnection is NULL: false
```

Use Case: 1

- 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 NoSQLConnections.
- Migration from MySQL to NoSQL



Solution is @ConditionalOnProperty Annotation

- It will make the key prefix and value and try to find this value in application.properties and try match this key with havingValue like below example.

prefix.value=havingValue

MySQLConnection.enabled=true

```
@ConditionalOnProperty(prefix = "MySQLConnection",value = "enabled",havingValue =
"true",matchIfMissing = false)
```

```
#application.properties
#prefifix.value=havingValue->matcing string
MySQLConnection.enabled=true
```

matchIfMissing > if configuration is not present for **@ConditionalOnProperty()** for **MySQLConnection** in **application.properties**

if **matchIfMissing** value is **true** then **bean** will be created and vice versa

@Autowired(required=false)

- By default **@Autowired required** is **true** that means **dependency** should be resolved or **object** should be present at application startup.
- required=false if bean is present then resolve it other wise let it go and initialized with null into that field.

```
package com.springProject.SpringCourseProject.component;

import org.springframework.boot.autoconfigure.condition.ConditionalOnProperty;
import org.springframework.stereotype.Component;

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

    public NoSqlConnection(){
        System.out.println("NoSqlConnection Is Initialized");
    }
}
```

```
package com.springProject.SpringCourseProject.component;

import org.springframework.boot.autoconfigure.condition.ConditionalOnProperty;
import org.springframework.stereotype.Component;

@Component
@ConditionalOnProperty(prefix = "MySQLConnection",value = "enabled",havingValue =
"true",matchIfMissing = false)
public class MySqlConnection {
    public MySqlConnection(){
        System.out.println("MySqlConnection Is Initialized");
    }
}
```

```
spring.application.name=SpringCourseProject
isOnlineOrder=false
#prefifix.value=havingValue->matcing string
MySQLConnection.enabled=true
```

```
package com.springProject.SpringCourseProject.component;

import jakarta.annotation.PostConstruct;
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.stereotype.Component;

import java.util.Objects;

@Component
public class DBConnection {

    @Autowired(required = false)
    MySqlConnection mySqlConnection;

    @Autowired(required = false)
    NoSqlConnection noSqlConnection;

    public DBConnection(){
        System.out.println("DBConnection Initialized");
    }

    @PostConstruct
    public void init(){
        System.out.println("DBConnection BEAN create with below dependencies");
        System.out.println("Is MySqlConnection is NULL: "+
Objects.isNull(mySqlConnection));
        System.out.println("Is NoSqlConnection is NULL: "+
Objects.isNull(noSqlConnection));
    }
}

output
DBConnection Initialized
MySqlConnection Is Initialized
DBConnection BEAN create with below dependencies
Is MySqlConnection is NULL: false
Is NoSqlConnection is NULL: true
```

@Autowired field just reference to the corresponding BEAN

Advantage

1. Toggling of Feature
2. Avoid cluttering Application context with un-necessary beans.
3. Save Memory
4. Reduce Application Startup time [1000 bens unnecessary].

Dis-Advantage

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