**@Qualifier**

• **org.springframework.beans.factory.annotation.Qualifier**

offers a better solution for this problem.

• It is used on a field or parameter as a qualifier for candidate beans when

autowiring.

• **Qualifier** annotation takes only one **String** argument as **value** which

serves as a qualifier to differentiate among the candidate beans.

• **value** should typically take the name of the one of the candidate

components for the injection.

Autowired bir implement durumunda calısıyor ama diyelim ki birden fazla implement durumu var bu sefer Autowired bunu belirleyemez hangisini implement edeceğini o yüzden qualifier kullanarak bunu belirtmis oluyoruz.

**@Qualifier : Birden fazla implement olan durumlarda kullanılır.**

• Qualifier should be used where @Autowired exists.

• If Qualifier uses the name of the target bean intended for injection

than the injection happens.



• It is possible to use @Qualifier with @Component too.

• In this case the values of both @Qualifier annotations must match.



• Qualifiers can also be used in XML configuration file as a nested element

</qualifier> of </bean>.

• </qualifier> has similarly one String attribute which is used when

injecting with @Autowired.

**How to Select Among Beans?**

• Think about the model used in **greeting12**:

• **GreetingRenderer** implementations are injected

**GreetingProvider** using **@Autowired**.

• What would happen if there were more than one implementation of

**GreetingProvider**?



• Observe that **GreetingProvider** has two implementations both of

which are annotated by @**Component**.



**How to Select Among Beans?**

• Spring throws **UnsatisfiedDependencyException** when it gets

confused regarding which bean to inject.

• **NoUniqueBeanDefinitionException** is the nested exception with

the message like No qualifying bean of type … vailable: expected single

matching bean but found 2:

• There are several solutions for the problem:

• Naming convention: Using matching names for both bean and variable or

parameter name at the injection point.

• Using **Primary** and **Qualifier** annotations.

• Qualification through generics.

**Using Matching Names**

• If the names of the component and the variable or parameter at

injection point are the same then Spring uses the bean with matching name.

• If the names of the candidate beans for the injection clash then the same problem occurs.



• This solution requires using component names in their clients.

• And it relies on naming convention.

**GreetingProvider** has two implementations both of which are

annotated by @**Component**.

**• Observe how the injecton is resolved through naming convention.**

