


# When to use Qualifier and Primary in Spring

Asked 9 months ago   Active 4 months ago   Viewed 415 times

 I have ☐d that `@Qualifier` can be used in Injection phase whereas `@Primary` is used in Configuration phase. Am still unclear when to use which.

4

Also I have below doubts



1

- can `@Primary` and `@Qualifier` be used together? if yes does `@Qualifier` take precedence?
- can `@Primary` be used with `@Autowired`?
- How is the Injection phase different from Configuration phase, this in respect to Spring beans



[spring](#)   [annotations](#)

asked Jun 18 '19 at 5:38



[Just Another Developer](#)

51   7

## 2 Answers

Active

Oldest

Votes



`@Primary` indicates that a bean should be given preference when multiple candidates are qualified to autowire a single-valued dependency.

3

`@Qualifier` indicates specific bean should be autowired when there are multiple candidates.



For example, we have two beans both implement the same interface.



```
public interface BeanInterface {  
    String getName();  
}  
  
public class Bean1 implements BeanInterface {  
    @Override  
    public String getName() {  
        return "bean 1";  
    }  
}  
  
public class Bean2 implements BeanInterface {  
    @Override  
    public String getName() {  
        return "bean2";  
    }  
}
```

Here is our service.

```
@Service  
public class BeanService {  
    @Autowired  
    private BeanInterface bean;  
}
```

And our configuration.

```
@Configuration
public class Config {

    @Bean("bean1")
    public BeanInterface bean1() {
        return new Bean1();
    }

    @Bean("bean2")
    public BeanInterface bean2() {
        return new Bean2();
    }
}
```

When Spring starts, it will find there two beans("bean1" and "bean2") both can be autowired to `BeanService` since they implement the same interface `BeanInterface` . It reports error in my Ideal.

```
Could not autowire. There is more than one bean of 'BeanInterface' type.
Beans: bean1    (Config.java)
       bean2    (Config.java)
```

And without a hint, Spring does not know which to use.

So in our case, when we add `@Primary` to `Config.bean1()` .

```
@Bean("bean1")
@Primary
public BeanInterface bean1() {
    return new Bean1();
}
```

It tells Spring, "when you find more than one beans that both can be autowired, please use the **primary one** as your first choose." So, Spring will pick `bean1` to autowire to `BeanService` .

Here is another way to autowire `bean1` to `BeanService` by using `@Qualifier` in `BeanService.class` .

```
@Service
public class BeanService {

    @Autowired
    @Qualifier("bean1")
    private BeanInterface bean;
}
```

`@Qualifier` will tell Spring, "no matter how many beans you've found, **just use the one I tell you.**"

So you can find both `@Qualifier` and `@Primary` are telling Spring to use the specific bean when multiple candidates are qualified to autowire. But `@Qualifier` is more specific and has high priority. So when both `@Qualifier` and `@Primary` are found, `@Primary` will be ignored.

Here is the test.

```
@Configuration
public class Config {

    @Bean("bean1")
    @Primary
```

```
public BeanInterface bean1() {
    return new Bean1();
}

@Bean("bean2")
public BeanInterface bean2() {
    return new Bean2();
}

@Service
public class BeanService {

    @Autowired
    @Qualifier("bean2")
    private BeanInterface bean;

    @PostConstruct
    public void test() {
        String name = bean.getName();
        System.out.println(name);
    }
}
```

The output is "bean2".

edited Jun 27 '19 at 6:25

answered Jun 18 '19 at 8:27

 **weaver**

5891316

Wow, you have cleared all my doubts..Many thanks for the crystal clear example! How frequently do we use annotation based Configuration as opposed to XML, is xml based configuration better or scores over programmatic? – [Just Another Developer](#) Jun 20 '19 at 4:38

It's a trend to use java config since programmatic config is much more convenient and clearly. – [weaver](#) Jun 27 '19 at 3:49

Yes, @Qualifier takes precedence. Yes we can user @Primary and @Autowired together.

0 @Configuration Class related to Application Context which can be use to configure application level configuration.

edited Nov 22 '19 at 18:44

answered Jun 18 '19 at 7:10

 **Zigri2612**

1,6031325