

## Spring - 4

### 1. Base project:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure: `Project Ex`, `Servers [springProject]`, `spring-autowiring-qua` (selected), `src/main/java` (containing `com.seleniumexpress`, `App.java`, `Body.java`, `Heart.java`, `Human.java`, `beans.xml`), `src/test/java`, `JRE System Library`, `Maven Dependencies`, `src`, `target`, and `pom.xml`.
- Code Editor:** Displays the `Human.java` file content:1 package com.seleniumexpress.di.spring\_autowiring\_qualifier;
2
3 public class Human
4 {
5 private Heart heart;
6
7 public void setHeart(Heart heart) {
8 this.heart = heart;
9 }
10
11 public void startPumping()
12 {
13 if(heart != null)
14 heart.pump();
15 else
16 System.out.println("Not pumping");
17 }
18 }
19
- Bottom Bar:** Shows tabs for `Markers`, `Properties`, `Servers`, `Data Source Explorer`, `Snippets`, and `Console`. The `Console` tab is active, displaying the message: `<terminated> Body [Java Application] /Users/aniketkanade/p2/pool/plugins/org.eclipse.jdt.core/pumping`.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Same as the first screenshot.
- Code Editor:** Displays the `Heart.java` file content:1 package com.seleniumexpress.di.spring\_autowiring\_qualifier;
2
3 public class Heart
4 {
5 public void pump()
6 {
7 System.out.println("pumping");
8 }
9 }
10
- Bottom Bar:** Same as the first screenshot.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Same as the first screenshot.
- Code Editor:** Displays the `beans.xml` file content:1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <beans xmlns="http://www.springframework.org/schema/beans"
4 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5 xsi:schemaLocation="http://www.springframework.org/schema/beans
6 http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
7
8 <bean id="heartObject" class="com.seleniumexpress.di.spring\_autowiring\_qualifier.Heart"/>
9
10 <bean id="human" class="com.seleniumexpress.di.spring\_autowiring\_qualifier.Human">
11 <property name="heart" ref="heartObject"/>
12 </bean>
13
14 </beans>
- Bottom Bar:** Shows tabs for `Design` and `Source`. The `Source` tab is active, displaying the message: `<terminated> Body [Java Application] /Users/aniketkanade/p2/pool/plugins/org.eclipse.jdt.core/pumping`.

```

Human.java   Heart.java   beans.xml   Body.java
1 package com.seleniumexpress.di.spring_autowiring_qualifier;
2 import org.springframework.context.ApplicationContext;
3 import org.springframework.context.support.ClassPathXmlApplicationContext;
4
5 public class Body
6 {
7     public static void main(String[] args)
8     {
9         ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
10        Human human = context.getBean("human",Human.class);
11        human.startPumping();
12    }
13
14 }
15
16
17

```

Markers Properties Servers Data Source Explorer Snippets Console X

<terminated> Body [Java Application] /Users/aniketkanade/p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86\_64 pumping

## 2. Iteration 1: (byName)

`<bean id = "human" class="com.seleniumexpress.autowired.Human" autowire="byName">`

`</bean>`

**Dear Spring !!**  
**While creating **Human** class object any dependency present in Human class meeting **autowire = "byName"** criteria, Inject those beans to their respective dependency.**

`3 public class Human {`  
`4`  
`5 private Heart heart;`

**Heart class reference variable('heart') name and bean id name ('heart') matched !!**

`<bean id = "heart" class="com.seleniumexpress.autowired.Heart"></bean>`

SUBSCRIBE

```

Human.java   Heart.java   *beans.xml*   Body.java
1 <?xml version="1.0" encoding="UTF-8"?>
2
3<beans xmlns = "http://www.springframework.org/schema/beans"
4   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation = "http://www.springframework.org/schema/beans
6   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
7
8<bean id="heart" class="com.seleniumexpress.di.spring_autowiring_qualifier.Heart"/>
9
10<bean id="human" class="com.seleniumexpress.di.spring_autowiring_qualifier.Human"
11   autowire="byName">
12</bean>
13
14<!-- Explanation: As in Human.java class, variable name is "heart"
15 it will automatically autowire to bean named "heart" -->
16
17</beans>

```

Design Source

Markers Properties Servers Data Source Explorer Snippets Console X

<terminated> Body [Java Application] /Users/aniketkanade/p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86\_64 pumping

### 3. Iteration 2: (byType)

The screenshot shows the Eclipse IDE interface with the beans.xml file open in the editor. The XML code defines two beans: 'heartObject' of type Heart and 'human' of type Human. The 'human' bean is annotated with @Autowire(byType=true). A tooltip explains that since the variable name is 'heart', it will automatically autowire to the 'Heart' bean.

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
    <bean id="heartObject" class="com.seleniumexpress.di.spring_autowiring_qualifier.Heart">
    <bean id="human" class="com.seleniumexpress.di.spring_autowiring_qualifier.Human"
        autowire="byType">
    </bean>
    <!-- Explanation: As in Human.java class, variable name is "heart"
        it will automatically autowire to bean of type "heart" -->
</beans>

```

### 4. Iteration 3: (byConstructor)

The screenshot shows the Eclipse IDE interface with the Human.java file open in the editor. It contains a constructor that takes a Heart object as a parameter. Below, the beans.xml file is shown with the 'human' bean annotated with @Autowire(constructor=true). A tooltip explains that whenever a constructor is available, it will be used for dependency injection.

```

package com.seleniumexpress.di.spring_autowiring_qualifier;
public class Human
{
    private Heart heart;
    public Human(Heart heart) {
        super();
        this.heart = heart;
    }
    public void setHeart(Heart heart) {
        this.heart = heart;
    }
    public void startPumping()
    {
        if(heart != null)
            heart.pump();
        else
            System.out.println("Not pumping");
    }
}

```

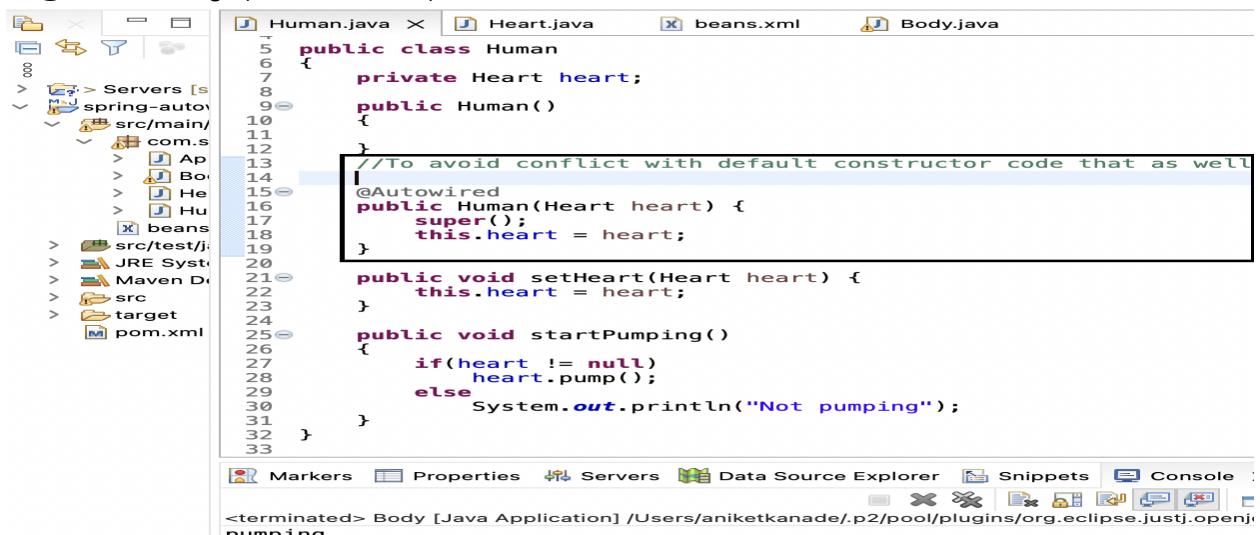
  

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
    <bean id="heartObject" class="com.seleniumexpress.di.spring_autowiring_qualifier.Heart">
    <bean id="human" class="com.seleniumexpress.di.spring_autowiring_qualifier.Human"
        autowire="constructor">
    </bean>
    <!-- Explanation: Whenever there is constructor available
        for dependency it will autowire -->
</beans>

```

## 5. @autowire tag: (at constructor)



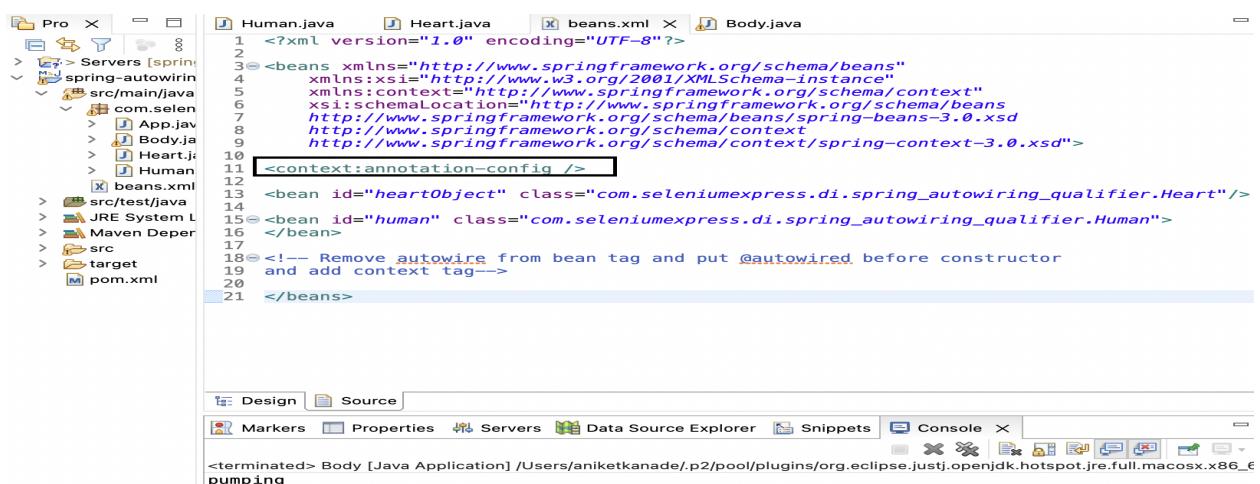
```

Human.java X Heart.java beans.xml Body.java
1 public class Human
2 {
3     private Heart heart;
4     public Human()
5     {
6     }
7     //To avoid conflict with default constructor code that as well
8     @Autowired
9     public Human(Heart heart) {
10         super();
11         this.heart = heart;
12     }
13     public void setHeart(Heart heart) {
14         this.heart = heart;
15     }
16     public void startPumping()
17     {
18         if(heart != null)
19             heart.pump();
20         else
21             System.out.println("Not pumping");
22     }
23 }
24
25
26
27
28
29
30
31
32
33

```

Markers Properties Servers Data Source Explorer Snippets Console

<terminated> Body [Java Application] /Users/aniketkanade/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86\_64 pumping

```

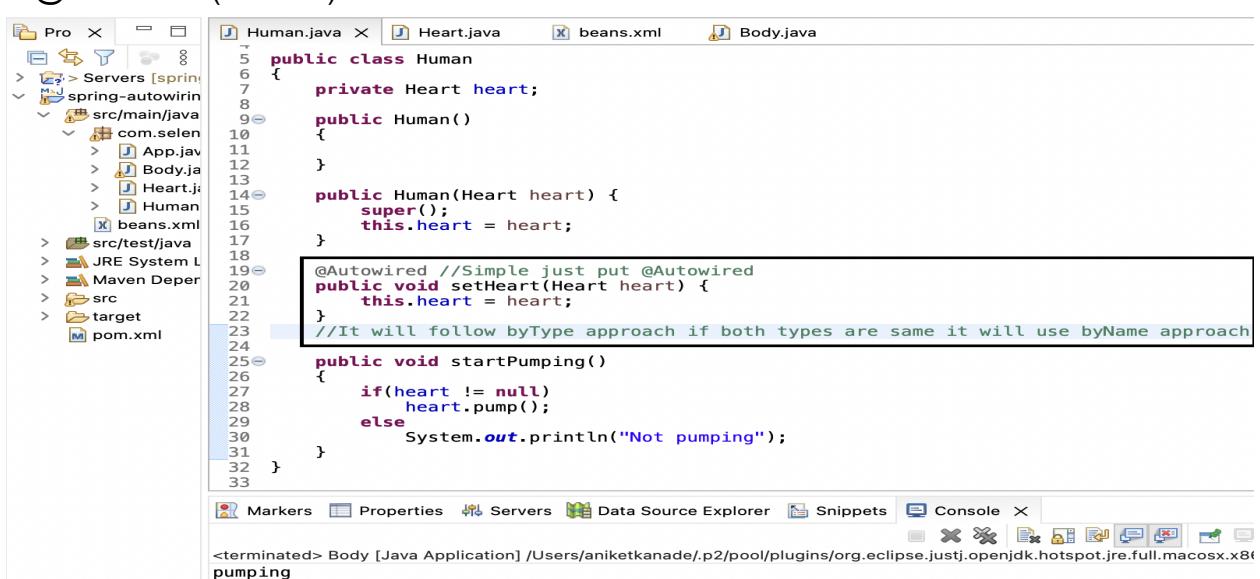
Human.java Heart.java beans.xml Body.java
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <beans xmlns="http://www.springframework.org/schema/beans"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xmlns:context="http://www.springframework.org/schema/context"
6   xsi:schemaLocation="http://www.springframework.org/schema/beans
7   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
8   http://www.springframework.org/schema/context
9   http://www.springframework.org/schema/context/spring-context-3.0.xsd">
10 <context:annotation-config />
11
12 <bean id="heartObject" class="com.seleniumexpress.di.spring_autowiring_qualifier.Heart"/>
13 <bean id="human" class="com.seleniumexpress.di.spring_autowiring_qualifier.Human">
14 </bean>
15
16 <!-- Remove autowire from bean tag and put @Autowired
17 and add context tag-->
18
19
20
21 </beans>

```

Design Source

<terminated> Body [Java Application] /Users/aniketkanade/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86\_64 pumping

## 6. @Autowired: (at setter)



```

Human.java X Heart.java beans.xml Body.java
1 public class Human
2 {
3     private Heart heart;
4     public Human()
5     {
6     }
7     public Human(Heart heart) {
8         super();
9         this.heart = heart;
10    }
11
12    @Autowired //Simple just put @Autowired
13    public void setHeart(Heart heart) {
14        this.heart = heart;
15    }
16    //It will follow byType approach if both types are same it will use byName approach
17
18    public void startPumping()
19    {
20        if(heart != null)
21            heart.pump();
22        else
23            System.out.println("Not pumping");
24    }
25
26
27
28
29
30
31
32
33

```

Markers Properties Servers Data Source Explorer Snippets Console

<terminated> Body [Java Application] /Users/aniketkanade/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86\_64 pumping

- Autowiring is not possible for primitive types. Its only for reference types.

## 7. Qualifier:

The screenshot shows two Eclipse IDE windows demonstrating the use of the `@Qualifier` annotation for autowiring.

**Top Window (Screenshot 1):**

- Project Explorer:** Shows a project named "spring-autowiring" with packages like `src/main/java/com.selenium` containing files `App.java`, `Body.java`, `Heart.java`, and `Human.java`.
- Code Editor:** Displays the `Heart.java` file with the following code:
 

```

3 public class Heart
4 {
5     private String nameOfAnimal;
6     private int noOfHeart;
7
8
9
10    public String getNameOfAnimal() {
11        return nameOfAnimal;
12    }
13
14    public void setNameOfAnimal(String nameOfAnimal) {
15        this.nameOfAnimal = nameOfAnimal;
16    }
17
18    public int getNoOfHeart() {
19        return noOfHeart;
20    }
21
22
23    public void setNoOfHeart(int noOfHeart) {
24        this.noOfHeart = noOfHeart;
25    }
26
27
28    public void pump()
29    {
30        System.out.println("pumping");
31    }
      
```
- Console:** Shows the output: `pumping`.

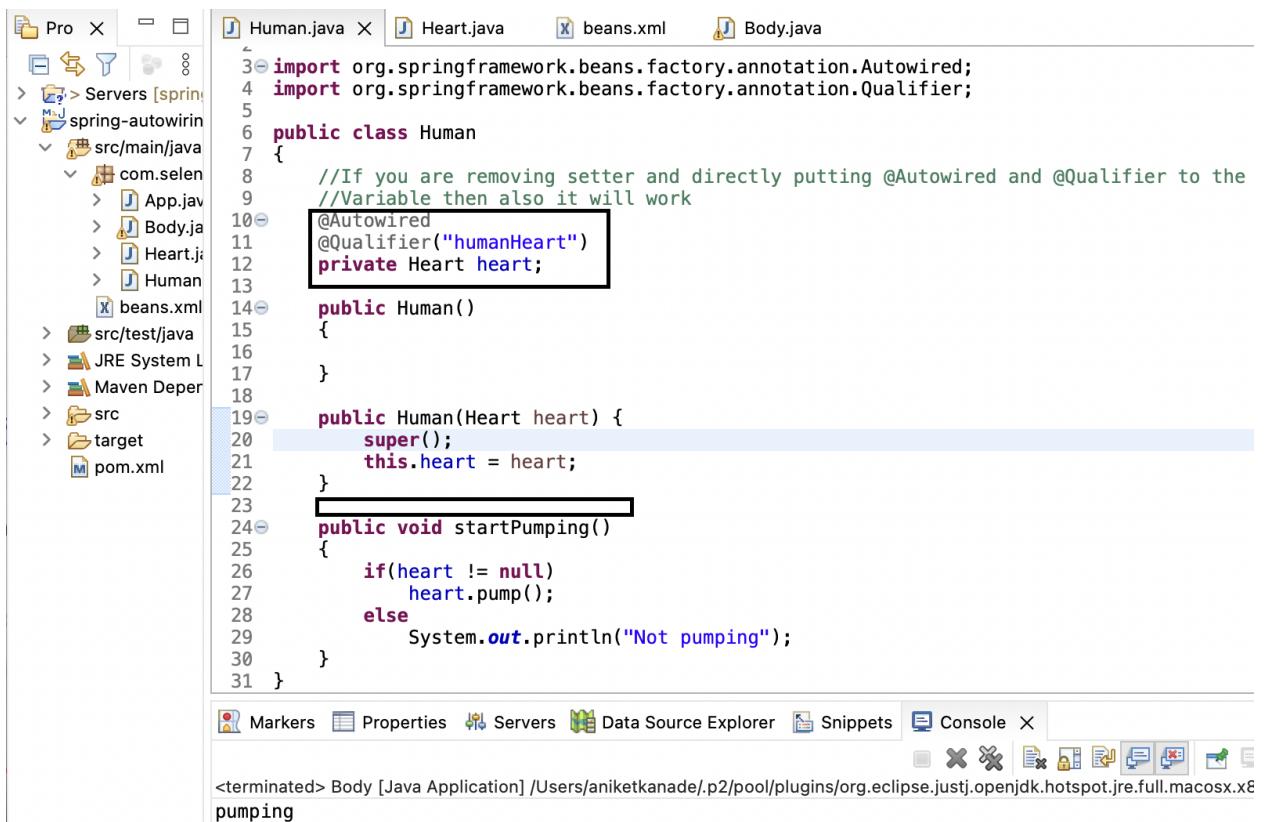
**Bottom Window (Screenshot 2):**

- Code Editor:** Displays the `Human.java` file with the following code:
 

```

8     private Heart heart;
9
10    public Human()
11    {
12
13    }
14
15    public Human(Heart heart) {
16        super();
17        this.heart = heart;
18    }
19
20    //as in beans.xml both humanHeart and octopusHeart byType and byName
21    //are not possible so better option is to use @Qualifier
22    @Autowired
23    @Qualifier("humanHeart")
24    public void setHeart(Heart heart) {
25        this.heart = heart;
26    }
27
28    public void startPumping()
29    {
30        if(heart != null)
31            heart.pump();
32        else
33            System.out.println("Not pumping");
34    }
      
```
- Console:** Shows the output: `pumping`.

If we remove setter methods then also it will work:



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer (left):** Shows the project structure with files like Human.java, Heart.java, beans.xml, and Body.java.
- Code Editor (center):** Displays the Human.java code. The code includes imports for org.springframework.beans.factory.annotation.Autowired and org.springframework.beans.factory.annotation.Qualifier. It defines a Human class with a constructor taking a Heart object and a startPumping method. The code is annotated with `@Autowired` and `@Qualifier("humanHeart")` on the private field and the constructor parameter.
- Console (bottom):** Shows the output of the application: `pumping`.

```
Human.java X Heart.java beans.xml Body.java
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.beans.factory.annotation.Qualifier;
5
6 public class Human {
7 {
8     //If you are removing setter and directly putting @Autowired and @Qualifier to the
9     //Variable then also it will work
10    @Autowired
11    @Qualifier("humanHeart")
12    private Heart heart;
13
14    public Human()
15    {
16    }
17
18    public Human(Heart heart) {
19        super();
20        this.heart = heart;
21    }
22
23
24    public void startPumping()
25    {
26        if(heart != null)
27            heart.pump();
28        else
29            System.out.println("Not pumping");
30    }
31 }
```

Markers Properties Servers Data Source Explorer Snippets Console

<terminated> Body [Java Application] /Users/aniketkanade/p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86 pumping