@Value Annotation

This annotation can be used for injecting values into fields in Spring-managed beans, and it can be applied at the field or constructor/method parameter level.

Now if you observe the example below for an engine bean there are two more attributes we have defined colour and price. Let’s try to fetch the colour and price attribute in the main method

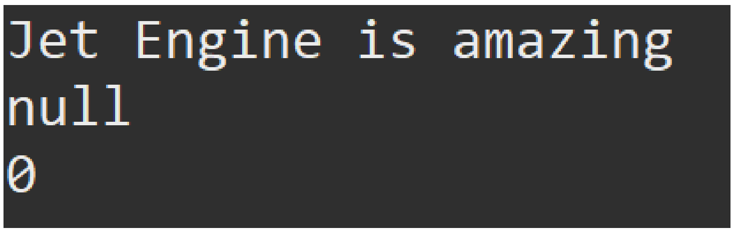
Audi.java

```
@Component("audi")
public class Audi implements Car
{
	@Autowired
	@Qualifier("jetEngine")
	private Engine engine;
	private String colour;
	private int price;

	public String getColour() {
		return colour;
	}

	public void setColour(String colour) {
		this.colour = colour;
	}

	public int getPrice() {
		return price;
	}

	public void setPrice(int price) {
		this.price = price;
	}
.
.
}
```

MyApp9.java

```
package com.tapacad.spring;
import
org.springframework.context.support.ClassPathXmlApplication
Context;

public class MyApp9 {

     public static void main(String[] args) {
//         Load Application context
           ClassPathXmlApplicationContext context =
                    new
ClassPathXmlApplicationContext("annotateApplicationContext.
xml");

//         Get bean
           Audi car = context.getBean("audi", Audi.class);

//         call getCarDetails()
           System.out.println(car.getEngineDetail());
           System.out.println(car.getColour());
           System.out.println(car.getPrice());

//         close context
           context.close();

     }
}
```

Output:



```
Jet Engine is amazing
null
0
```

Here as you can see from the above output the value for colour and price is not assigned.

Now to get the values and to assign the values you need to make use of **@value annotations** as shown below

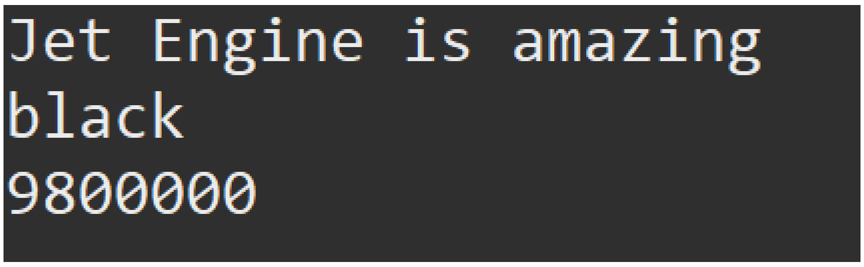
Audi.java

```
@Component("audi")
public class Audi implements Car
{
     @Autowired
     @Qualifier("jetEngine")
     private Engine engine;

     @Value("black") // assigning black to String colour
     private String colour;
     @Value("9800000") // assigning 9800000 to String price
     private int price;
.
.
}
```

MyApp9.java

```
package com.tapacad.spring;

import
org.springframework.context.support.ClassPathXmlApplication
Context;

public class MyApp9 {

     public static void main(String[] args) {
//         Load Application context
           ClassPathXmlApplicationContext context =
                     new
ClassPathXmlApplicationContext("annotateApplicationContext.
xml");

//         Get bean
           Audi car = context.getBean("audi", Audi.class);
//          call getCarDetails()
            System.out.println(car.getEngineDetail());
            System.out.println(car.getColour());
            System.out.println(car.getPrice());

//          close context
            context.close();

     }

}
```

Output:



```
Jet Engine is amazing
black
9800000
```

As you can see from the above output **@Value** have inserted the value of colour and price inside the attribute

Now let’s see how we can make use of property file to insert the value inside colour and price

Step 1: Now we need to mention the property-placeholder inside the xml file of **annotateApplicationContext.xml**

```
<context:property-placeholder
location="classpath:car.properties"/>
```

annotateApplicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/contex
t"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xs
d
        http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-contex
t.xsd">

       <context:component-scan
base-package="com.tapacad.spring"></context:component-scan>
        <context:property-placeholder
location="classpath:car.properties"/>
</beans>
```

car.properties

```
colour = blue
price =  6500000
```

Step 2: we get *Value got from the file* assigned to the field as shown below

```
@Value("${colour}")
private String colour;
@Value("${price}")
private int price;
```

Audi.java

```
package com.tapacad.spring;

import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Component("audi")
public class Audi implements Car
{
	@Autowired
	@Qualifier("jetEngine")
	private Engine engine;

	@Value("${colour}")
	private String colour;
	@Value("${price}")
	private int price;

	@Override
	public String getCarDetails() {
		return "Audi";
```

```java
    }

    @Override
    public String getEngineDetail() {
         return engine.getEngineDetail();
    }

    public Engine getEngine() {
         return engine;
    }

    public void setEngine(Engine engine) {
         this.engine = engine;
    }

    public Engine getRocketEngine() {
         return engine;
    }

    public void setRocketEngine(Engine engine) {
         this.engine = engine;
    }

    public String getColour() {
         return colour;
    }

    public void setColour(String colour) {
         this.colour = colour;
    }

    public int getPrice() {
         return price;
    }

    public void setPrice(int price) {
         this.price = price;
```

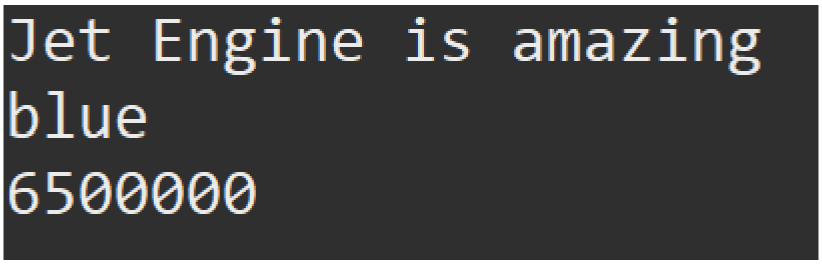
```
     }

     void myStartUp(){
          System.out.println("Bean Created");
     }

     void myClosing(){
          System.out.println("Bean Destroyed");
     }

     void fun1(){
          System.out.println("Default Bean Created");
     }

     void fun2(){
          System.out.println("Default Bean Destroyed");
     }



}
```

Step 3: Now let’s see weather the values mentioned in property file is been inserted or not

MyApp9.java

```
package com.tapacad.spring;
import
org.springframework.context.support.ClassPathXmlApplication
Context;

public class MyApp9 {

     public static void main(String[] args) {
//        Load Application context
          ClassPathXmlApplicationContext context =
                    new
ClassPathXmlApplicationContext("annotateApplicationContext.
xml");

//		Get bean
		Audi car = context.getBean("audi", Audi.class);

//		call getCarDetails()
		System.out.println(car.getEngineDetail());
		System.out.println(car.getColour());
		System.out.println(car.getPrice());

//		close context
		context.close();

	}
}
```

Output:



```
Jet Engine is amazing
blue
6500000
```

As you can see from the above output the values that we have mentioned in property file is assigned to the attribute colour and price