@scope Annotations

Previously we need to mention the scope inside the xml file as shown below

```
<bean id="bmw" class="com.tapacad.spring.BMW"
scope="singleton">
</bean>
```

As you can see from the above xml file, for the bean we have mentioned the scope as singleton

Now to represent the bean using annotations we had made use of **@Component annotation**
There only we need to mention **@Scope("singleton")** as shown below

BMW.java

```
@Scope("singleton")
@Component("bmw")
public class BMW implements Car
{
  .
  .
}
```

Now let's try to create two bmw bean and see weather one bean object is creating or multiple

```
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
          Car car1 = context.getBean("bmw", Car.class);
          Car car2 = context.getBean("bmw", Car.class);

//        print car1 and car2 object reference
          System.out.println(car1);
          System.out.println(car2);

//        close context
          context.close();

     }

}
```

Output:

```
com.tapacad.spring.BMW@31fa1761
com.tapacad.spring.BMW@31fa1761
```

As you can see from the above both beans are pointing the same object as the **scope is singleton**

Now let’s change the scope to prototype and see weather it works

BMW.java

```
@Scope("prototype")
@Component("bmw")
public class BMW implements Car
{
	.
	.
}
```

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
		Car car1 = context.getBean("bmw", Car.class);
		Car car2 = context.getBean("bmw", Car.class);

//		print car1 and car2 object reference
		System.out.println(car1);
		System.out.println(car2);

//		close context
		context.close();
	}
}
```

Output:

```
com.tapacad.spring.BMW@32502377
com.tapacad.spring.BMW@2c1b194a
```

As you can clearly see from the above output both beans are pointing to different objects as the **scope is prototype**