SPRING @QUALIFIER ANNOTATION

There may be a situation when you create more than one bean of the same type and want to wire only one of them with a property, in such case you can use **@Qualifier** annotation along with **@Autowired** to remove the confusion by specifying which exact bean will be wired. Below is an example to show the use of @Qualifier annotation.

Example:

Let us have working Eclipse IDE in place and follow the following steps to create a Spring application:

Step	Description
1	Create a project with a name *SpringExample* and create a package *com.tutorialspoint* under the **src** folder in the created project.
2	Add required Spring libraries using *Add External JARs* option as explained in the *Spring Hello World Example* chapter.
3	Create Java classes *Student*, *Profile* and *MainApp* under the *com.tutorialspoint* package.
4	Create Beans configuration file *Beans.xml* under the **src** folder.
5	The final step is to create the content of all the Java files and Bean Configuration file and run the application as explained below.

Here is the content of **Student.java** file:

```
package com.tutorialspoint;

public class Student {
   private Integer age;
   private String name;

   public void setAge(Integer age) {
      this.age = age;
   }

   public Integer getAge() {
      return age;
   }

   public void setName(String name) {
      this.name = name;
   }

   public String getName() {
      return name;
   }
}
```

Here is the content of **Profile.java** file:

```
package com.tutorialspoint;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;

public class Profile {
   @Autowired
```

```
   @Qualifier("student1")
   private Student student;

   public Profile(){
      System.out.println("Inside Profile constructor." );
   }

   public void printAge() {
      System.out.println("Age : " + student.getAge() );
   }

   public void printName() {
      System.out.println("Name : " + student.getName() );
   }
}
```

Following is the content of the **MainApp.java** file:

```
package com.tutorialspoint;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
   public static void main(String[] args) {
      ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");

      Profile profile = (Profile) context.getBean("profile");

      profile.printAge();
      profile.printName();
   }
}
```

Consider the example of following configuration file **Beans.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/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.0.xsd">

   <context:annotation-config/>

   <!-- Definition for profile bean -->
   <bean  >
   </bean>

   <!-- Definition for student1 bean -->
   <bean  >
      <property name="name"  value="Zara" />
      <property name="age"  value="11"/>
   </bean>

   <!-- Definition for student2 bean -->
   <bean  >
      <property name="name"  value="Nuha" />
      <property name="age"  value="2"/>
   </bean>

</beans>
```

Once you are done with creating source and bean configuration files, let us run the application. If

everything is fine with your application, this will print the following message:

```
Inside Profile constructor.
Age : 11
Name : Zara
```