SPRING @REQUIRED ANNOTATION

The **@Required** annotation applies to bean property setter methods and it indicates that the affected bean property must be populated in XML configuration file at configuration time otherwise the container throws a BeanInitializationException exception. Below is an example to show the use of @Required 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* 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;

import org.springframework.beans.factory.annotation.Required;

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

   @Required
   public void setAge(Integer age) {
      this.age = age;
   }
   public Integer getAge() {
      return age;
   }

   @Required
   public void setName(String name) {
      this.name = name;
   }
   public String getName() {
      return name;
   }
}
```

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");

      Student student = (Student) context.getBean("student");

      System.out.println("Name : " + student.getName() );
      System.out.println("Age : " + student.getAge() );
   }
}
```

Following is the content of the 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 student bean -->
   <bean  >
      <property name="name"  value="Zara" />

      <!-- try without passing age and check the result -->
      <!-- property name="age"  value="11"-->
   </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 raise *BeanInitializationException* exception and print the following error along with other log messages:

```
Property 'age' is required for bean 'student'
```

Next, you can try above example after removing comment from 'age' property as follows:

```
<?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 student bean -->
   <bean  >
      <property name="name"  value="Zara" />
      <property name="age"  value="11"/>
   </bean>

</beans>
```

Now above example will produce following result:

```
Name : Zara
```

```
Age : 11
```