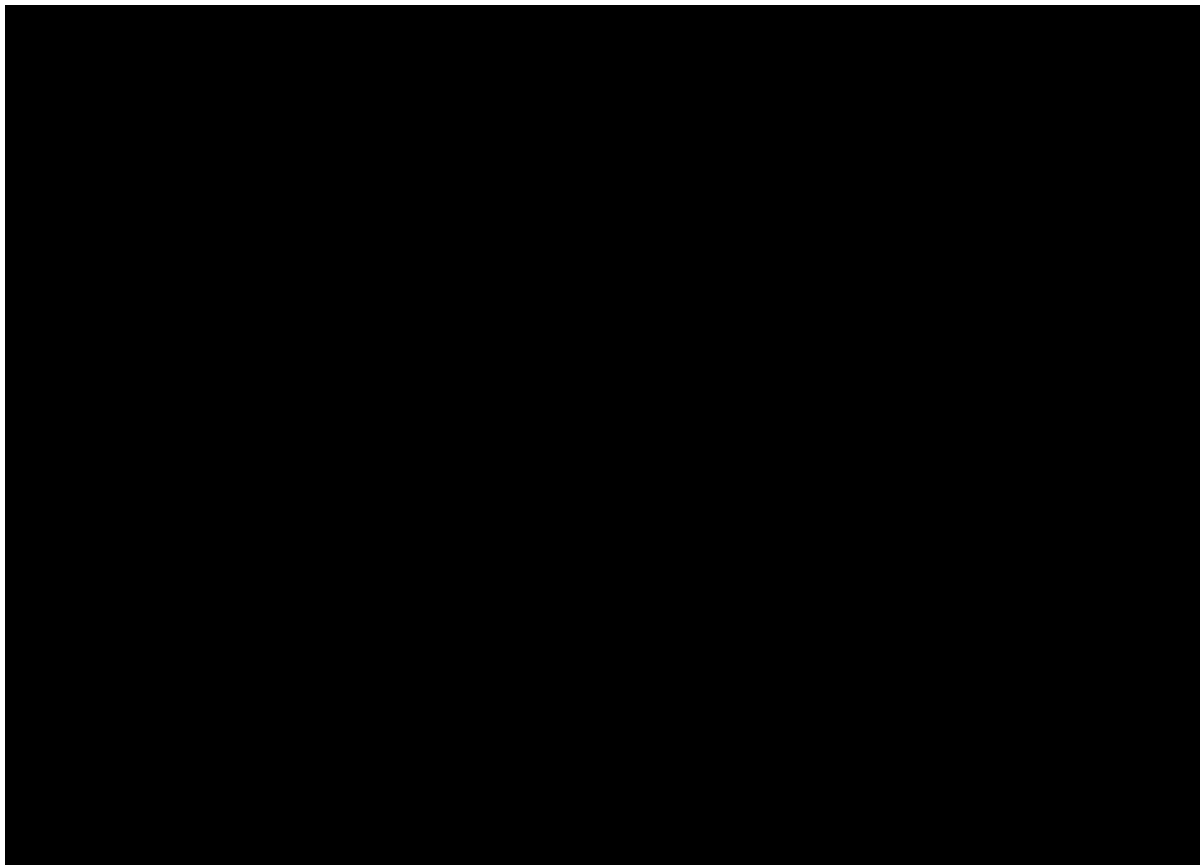


# Spring Framework 3.1 Tutorial





---

All the content and graphics on this tutorial are the property of tutorialspoint.com. Any content from tutorialspoint.com or this tutorial may not be redistributed or reproduced in any way, shape, or form without the written permission of tutorialspoint.com. .

# Table of Contents

Spring Framework Overview .....	1
<b>Benefits of Using Spring Framework.....</b>	<b>1</b>
<b>Dependency Injection (DI) .....</b>	<b>2</b>
<b>Aspect Oriented Programming (AOP) .....</b>	<b>2</b>
Spring Framework Architecture .....	3
Core Container.....	



Spring @Qualifier Annotation .....









# Spring Framework Architecture

*This section describes the basic architecture of the framework*









## Step 4 - Setup Spring Framework Libraries

















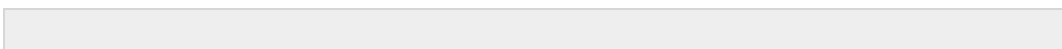
# Spring IoC Containers

# T





---



4 Create Beans configuration file *Beans.xml* under the **src** folder.

```
<?xml version=C2 RG encoding=UTF-8?>
```















































Second method of injecting dependency is through **Setter Methods**

















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

```
package com.tutorialspoint;

public class TextEditor {
    private SpellChecker spellChecker;

    // a setter method to inject the dependency.
    public void setSpellChecker(SpellChecker spellChecker) {
        System.out.println("Inside setSpellChecker." );
        this
```







package

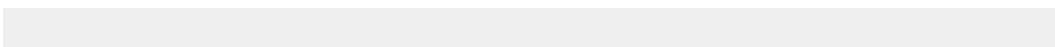








## Limitations with autowiring



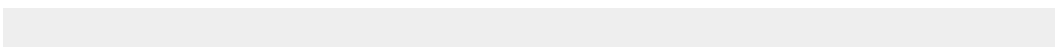








}















Following is the configuration file



```
public void spellCheck(){
```





```
        this.name = name;
    }
    public String getName() { }
```

















the foo bean receives a reference to bar via constructor injection. Now let us see one example:

## Example

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

Description
-------------

```
this.spellChecker W>> BDC BT
```



```
@Bean
@Scope("prototype")
public Foo foo() {
    return new Foo();
}
```



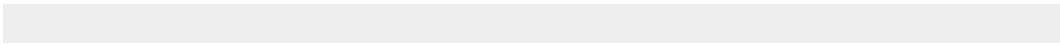


```
package com.tutorialspoint;
```



```
</beans>
```









Target object



```
</aop:config>
```







```
xsi:schemaLocation="http://www.springframework.org/schema/beans
```

Let me explain that above defined `<aop:pointcut>` selects all t















# Spring JDBC Framework

**W**hile working with database using plain old JDBC, it becomes cumbersome to write





















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

```
public void
```

```
(StudentJDBCTemplate)context.getBean("studentJDBCTemplate")
```







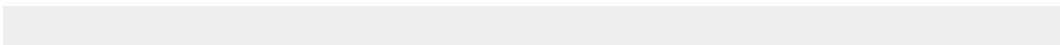


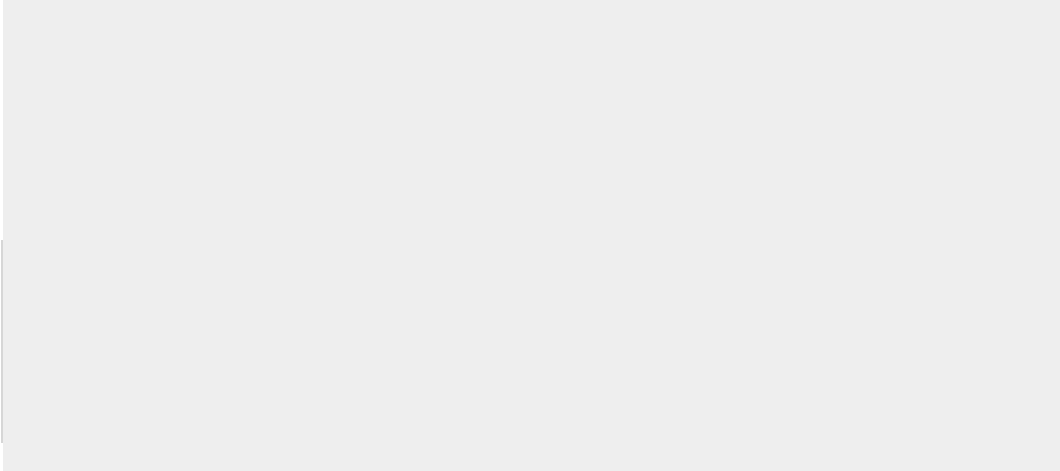


6	<b>TransactionDefinition.PROPGATION_REQUIRES_NEW</b> Create a new transaction, suspending the current transaction if one exists.



```
/**  
 * This is the method to be used to initialize  
 * database resources ie. connection.  
 */
```







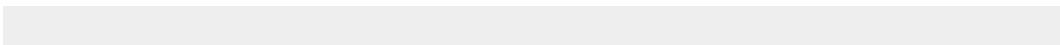






[package](#) com

```
public class StudentMarksMapper
```









2.





Here **`${message}`** is the attribute which we have setup inside the Controller. You can have



```
<html>
<head>
<title>Hello World</title>
</head>
<body>
  <h2>${message}</h2>
</body>
</html>
```



```
private String name;  
private Integer id;  
  
public void setAge(Integer age) {  
    this.age = age;  
}
```





```
<%@taglib
```

```
</html>
```

Finally, following is the list of Spring and other libraries to be included in your web application. You simply drag these files and drop them in **WebContent/WEB-INF/lib**

