通过注解的形式装载Bean，而不是用XML。xml的配置文件：

```
<?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/>
   <!-- bean definitions go here -->
</beans>
```

序号	注解 & 描述
1	@Required @Required 注解应用于 bean 属性的 setter 方法。
2	@Autowired @Autowired 注解可以应用到 bean 属性的 setter 方法，非 setter 方法，构造函数和属性。
3	@Qualifier 通过指定确切的将被连线的 bean，@Autowired 和 @Qualifier 注解可以用来删除混乱。
4	JSR-250 Annotations Spring 支持 JSR-250 的基础的注解，其中包括了 @Resource，@PostConstruct 和 @PreDestroy 注解。

@Required 注释

应用于 bean 属性的 setter 方法，它表明受影响的 bean 属性在配置时必须放在 XML 配置文件中，否则容器就会抛出一个 BeanInitializationException 异常。例子：

```
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;
   }
}
```

在Beans.xml文件中配置：

```
<?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/>

   <bean id="student" class="com.tutorialspoint.Student">
      <property name="name"  value="Zara" />
   </bean>

</beans>
```

@Autowired 注释

对在哪里和如何完成自动连接提供了更多的细微的控制。

Setter 方法中的 @Autowired：

可以在 setter 方法中使用 @Autowired 注释来除去元素。当Spring遇到一个在 setter 方法中使用的 @Autowired 注释，它会在方法中执行 byType 自动连接。例子：

```
public class TextEditor {
   private SpellChecker spellChecker;
   @Autowired
   public void setSpellChecker( SpellChecker spellChecker ){
      this.spellChecker = spellChecker;
   }
   public SpellChecker getSpellChecker( ) {
      return spellChecker;
   }
   public void spellCheck() {
      spellChecker.checkSpelling();
   }
}
```

在Beans.xml文件中配置：

```
   <context:annotation-config/>
   <bean id="textEditor" class="com.tutorialspoint.TextEditor">
   </bean>

   <bean id="spellChecker" class="com.tutorialspoint.SpellChecker">
   </bean>
```

属性中的 @Autowired

可以在属性中使用 @Autowired 注释来除去 setter 方法。当使用自动连接为属性传递的时候，Spring 会将这些传递过来的值或者引用自动分配给那些属性。例子：

```
public class TextEditor {
   @Autowired
   private SpellChecker spellChecker;
   public TextEditor() {
```

```
      System.out.println("Inside TextEditor constructor." );
   }
   public SpellChecker getSpellChecker( ){
      return spellChecker;
   }
   public void spellCheck(){
      spellChecker.checkSpelling();
   }
}
```

在Beans.xml文件中配置：

```
   <context:annotation-config/>
   <bean id="textEditor" class="com.tutorialspoint.TextEditor">
   </bean>

   <bean id="spellChecker" class="com.tutorialspoint.SpellChecker">
   </bean>
```

构造函数中的 @Autowired

说明当创建 bean 时，即使在 XML 文件中没有使用 元素配置 bean ，构造函数也会被自动连接。例子：

```
public class TextEditor {
   private SpellChecker spellChecker;
   @Autowired
   public TextEditor(SpellChecker spellChecker){
      this.spellChecker = spellChecker;
   }
   public void spellCheck(){
      spellChecker.checkSpelling();
   }
}
```

在Beans.xml文件中配置：

```
   <context:annotation-config/>
   <bean id="textEditor" class="com.tutorialspoint.TextEditor">
   </bean>
```

```
<bean id="spellChecker" class="com.tutorialspoint.SpellChecker">
</bean>
```

关闭@Autowired 的（required=false）选项

默认情况下，@Autowired 注释意味着依赖是必须的，它类似于 @Required 注释。但可以使用 @Autowired 的 （required=false） 选项关闭默认行为。

@Qualifier 注释

当创建多个具有相同类型的 bean 时，并且想要用一个属性只为其中的一个进行装配。在这种情况下，可以使用 @Qualifier注释和 @Autowired 注释通过指定哪一个真正的 bean 将会被装配来消除混乱。例子：

```
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() );
   }
}
```

在Beans.xml文件中配置：

的示例：

```
<context:annotation-config/>
<bean id="profile" class="com.tutorialspoint.Profile">
</bean>

<bean id="student1" class="com.tutorialspoint.Student">
   <property name="name"  value="Zara" />
   <property name="age"  value="11"/>
</bean>

<bean id="student2" class="com.tutorialspoint.Student">
   <property name="name"  value="Nuha" />
   <property name="age"  value="2"/>
</bean>
```

JSR-250 注释

@PostConstruct 和 @PreDestroy 注释：

为了定义一个 bean 的安装和卸载，可以使用 @PostConstruct 注释作为初始化回调函数的一个替代，@PreDestroy 注释作为销毁回调函数的一个替代。

```
@PostConstruct
public void init(){
   System.out.println("Bean is going through init.");
}
@PreDestroy
public void destroy(){
   System.out.println("Bean will destroy now.");
}
```

@Resource 注释：

可以在字段中或者 setter 方法中使用 @Resource 注释。

@Resource 注释使用一个 ‘name’ 属性，该属性以一个 bean 名称的形式被注入。例子：

```
public class TextEditor {
   private SpellChecker spellChecker;
   @Resource(name= "spellChecker")
   public void setSpellChecker( SpellChecker spellChecker ){
      this.spellChecker = spellChecker;
   }
   public SpellChecker getSpellChecker(){
      return spellChecker;
   }
   public void spellCheck(){
      spellChecker.checkSpelling();
   }
}
```

如果没有明确地指定一个 ‘name’，默认名称源于字段名或者 setter 方法。在字段的情况下，它使用的是字段名；在一个 setter 方法情况下，它使用的是 bean 属性名称。

1.@Component（“id”）

dao层注解：@Repository

service层注解：@Service

控制器层注解：@Controller

2.在XML配置扫描器

<context:component-scan base-package="包，包">

</context:component-scan>