@Repository、@Service、@Controller 和 @Component

@Repository、@Service、@Controller 和 @Component 将类标识为Bean

spring 自 2.0 版本开始，陆续引入了一些注解用于简化 Spring 的开发。@Repository注解便属于最先引入的一批，它用于将数据访问层 (DAO 层) 的类标识为 Spring Bean。具体只需将该注解标注在 DAO类上即可。同时，为了让 Spring 能够扫描类路径中的类并识别出 @Repository 注解，需要在 XML 配置文件中启用Bean 的自动扫描功能，这可以通过 <context:component-scan/>实现。如下所示：

```
// 首先使用 @Repository 将 DAO 类声明为 Bean
package bookstore.dao;
@Repository
public class UserDaoImpl implements UserDao{ …… }
 // 其次，在 XML 配置文件中启动 Spring 的自动扫描功能
 <beans … > …… <context:component-scan base-package=”bookstore.dao” /> …… </beans>
```

如此，我们就不再需要在 XML 中显式使用 <bean/> 进行Bean 的配置。Spring 在容器初始化时将自动扫描 base-package 指定的包及其子包下的所有 class文件，所有标注了 @Repository 的类都将被注册为 Spring Bean。

为什么 @Repository 只能标注在 DAO 类上呢？这是因为该注解的作用不只是将类识别为Bean，同时它还能将所标注的类中抛出的数据访问异常封装为 Spring 的数据访问异常类型。 Spring本身提供了一个丰富的并且是与具体的数据访问技术无关的数据访问异常结构，用于封装不同的持久层框架抛出的异常，使得异常独立于底层的框架。

Spring 2.5 在 @Repository的基础上增加了功能类似的额外三个注解：@Component、@Service、@Constroller，它们分别用于软件系统的不同层次：

- @Component 是一个泛化的概念，仅仅表示一个组件 (Bean)，可以作用在任何层次。
- @Service 通常作用在业务层，但是目前该功能与 @Component 相同。
- @Constroller 通常作用在控制层，但是目前该功能与 @Component 相同。

通过在类上使用 @Repository、@Component、@Service 和 @Constroller 注解，Spring会自动创建相应的 BeanDefinition 对象，并注册到 ApplicationContext 中。这些类就成了 Spring受管组件。这三个注解除了作用于不同软件层次的类，其使用方式与 @Repository 是完全相同的。

另外，除了上面的四个注解外，用户可以创建自定义的注解，然后在注解上标注 @Component，那么，该自定义注解便具有了与所@Component 相同的功能。不过这个功能并不常用。

当一个 Bean 被自动检测到时，会根据那个扫描器的 BeanNameGenerator 策略生成它的 bean名称。默认情况下，对于包含 name 属性的 @Component、@Repository、 @Service 和@Controller，会把 name 取值作为 Bean 的名字。如果这个注解不包含 name值或是其他被自定义过滤器发现的组件，默认 Bean 名称会是小写开头的非限定类名。如果你不想使用默认 bean命名策略，可以提供一个自定义的命名策略。首先实现 BeanNameGenerator接口，确认包含了一个默认的无参数构造方法。然后在配置扫描器时提供一个全限定类名，如下所示：

```
<beans ...>
 <context:component-scan base-package="a.b" name-generator="a.SimpleNameGenerator"/> </beans>
```

与通过 XML 配置的 Spring Bean 一样，通过上述注解标识的Bean，其默认作用域是"singleton"，为了配合这四个注解，在标注 Bean 的同时能够指定 Bean 的作用域，Spring2.5 引入了 @Scope 注解。使用该注解时只需提供作用域的名称就行了，如下所示：

```
@Scope("prototype")
 @Repository
 public class Demo { …}
```

如果你想提供一个自定义的作用域解析策略而不使用基于注解的方法，只需实现 ScopeMetadataResolver接口，确认包含一个默认的没有参数的构造方法。然后在配置扫描器时提供全限定类名：

```
<context:component-scan base-package="a.b"
 scope-resolver="footmark.SimpleScopeResolver" />
```