03_Spring

笔记本:	Spring&SpringMVC
创建时间:	2019/9/3 17:55
作者:	韩延兵

1.HelloWorld

- 1）创建一个Maven版的java工程，通过依赖导入以下jar包

```
<dependencies>
        <!-- 只需要将context的jar包的依赖配置上，其他依赖的jar包会自动导入 -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>4.0.0.RELEASE</version>
        </dependency>
        <!-- junit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

- 2）创建一个类HelloWorld
  - 给当前类添加@Component注解

```
@Component
public class HelloWorld {
    public void sayHello() {
        System.out.println("Hello Spring!");
    }
}
```

- 3）创建Spring的配置文件applicationContext.xml，并在该文件中配置自动扫描的包
  - 直接右键→new→Spring Bean Configuration File

```
<?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/beanshttp://www.springfra
beans.xsd
http://www.springframework.org/schema/contexthttp://www.springframework.org/schema/cont
context-4.0.xsd">
    <!-- 设置自动扫描的包
        base-package属性：设置一个基础包，Spring会自动扫描该包及其子包
    -->
    <context:component-scan base-package="com.atguigu.spring.helloworld">
</context:component-scan>

</beans
```

- 4）测试

```
public class HelloWorldTest {
    @Test
    public void testHelloWorld() {
        //1.创建IOC容器对象
        ApplicationContext ioc = new
ClassPathXmlApplicationContext("applicationContext.xml");
        //2.从IOC容器中获取HelloWorld对象
        HelloWorld helloWorld = (HelloWorld) ioc.getBean("helloWorld");
        //3.调用sayHello方法
        helloWorld.sayHello();
    }
}
```

2.获取bean的方式

- 1）根据bean的名称（id属性值）获取
- 2）根据bean的类型获取，但要保证IOC容器中该类型的bean是唯一的，否则会抛出异常

```
/*
         * ★获取bean的方式：
         *  1）根据bean的名称（id属性值）获取
         *  2）根据bean的类型获取
         *      -注意：一定要保证IOC容器中该类型的bean是唯一的，否则会抛出异常
         */
        //根据bean的名称（id属性值）获取
        HelloWorld helloWorld = (HelloWorld) ioc.getBean("helloWorld");
        //根据bean的类型获取
        HelloWorld helloWorld2 = ioc.getBean(HelloWorld.class);
```

3.基于注解的方式配置bean

- 常用的注解
  - @Component
    - 标识一个普通组件
  - @Repository
    - 标识一个持久化层的组件
  - @Service
    - 标识一个业务逻辑层的组件
  - @Controller
    - 标识一个表现层的组件
  - @Autowired
    - 设置类中需要自动装配的属性
    - 添加了该注解的属性默认必须装配成功，否则会抛出异常
    - 如果要设置某个属性可以不装配，可以设置该注解中的required的属性值是false
  - @Qualifier
    - 通过该注解的value属性设置根据哪个id实现自动装配
- 使用注解的方式配置bean的步骤：
  - 1）在类上添加对应的注解
    - 不带属性的类

```
/*
 * 添加了注解的类会交给Spring的IOC容器管理，默认在IOC容器中的bean的id是类的首字母小写，
 * 我们也可以通过注解的value属性来指定该id，value的属性名可以省略
 */
```

```
//@Repository(value="userDao")
@Repository("userDao")
public class UserDaoImpl implements UserDao {
```

- 带属性的类

```
@Service("userService")
public class UserServiceImpl implements UserService {
    /*
     * 自动装配的步骤:
     *  1.根据属性的类型实现自动装配
     *  2.以属性名作为id从IOC容器中寻找
     *  3.我们还可以通过@Qualifier注解的value属性来指定根据那个id实现自动装配
     *
     */
    @Qualifier(value="userDao2")
    @Autowired
    private UserDao userDao;

    /*
     * 添加了@Autowired注解的属性默认必须装配成功，否则会抛出异常，
     * 我们可以通过指定@Autowired注解中的required属性值为false来告诉Spring当前属性可以不装配
     */
    @Autowired(required=false)
    private User user;
```

- 2）在Spring的配置文件中设置自动扫描的包

```
<?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/beanshttp://www.springframbeans.xsd
http://www.springframework.org/schema/contexthttp://www.springframework.org/schema/contcontext-4.0.xsd">
    <!-- 设置自动扫描的包
        base-package属性：设置一个基础包，Spring会自动扫描该包及其子包
    -->
    <context:component-scan base-package="com.atguigu.spring.annotation">
</context:component-scan>
</beans>
```

6.设置扫描或不扫描那些类

- 设置扫描那些类

```
<context:component-scan base-package="com.atguigu.spring.annotation" use-default-filters="false">
        <!--
            ★子标签context:include-filter：用来设置扫描那个包下的类
                注意：此时需要将父标签中的use-default-filters属性值设置为false
                  -如果type的值是annotation，那么expression表达式的值是注解的全类名
                  -如果type的值是assignable，那么expression表达式的值是接口或实现类的全类名
```

```
         -->
        <context:include-filter type="annotation"
expression="org.springframework.stereotype.Repository"/>
<!--        <context:include-filter type="assignable"
expression="com.atguigu.spring.annotation.dao.UserDao"/> -->
    </context:component-scan>
```

- 设置不扫描那些类

```
<context:component-scan base-package="com.atguigu.spring.annotation">
        <!--
            ★子标签context:exclude-filter：用来设置不扫描那个包下的类
                  -如果type的值是annotation，那么expression表达式的值是注解的全类
名
                  -如果type的值是assignable，那么expression表达式的值是接口或实现
类的全类名
         -->
        <context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Repository"/>
<!--        <context:exclude-filter type="assignable"
expression="com.atguigu.spring.annotation.dao.UserDao"/> -->
    </context:component-scan>
```