Spring 注解使用笔记

作者：XiaoCoder

原文链接：https://ld246.com/article/1610344643608

来源网站：链滴

1. **@Configuration**

作用：指定当前类是一个配置类

细节：当配置类作为AnnotationConfigApplicationContext对象创建的参数时，该注解可以不写。

```
*/
@Configuration
/**
 * 当其中是父与子关系的时候需要导入
 * @Import 其中是字节码类型的文件
 */
@ComponentScan({"com.xiao","config"})
@PropertySources(@PropertySource("classpath:jdbcConfig.properties"))
@Import(value = {JdbcConfig.class})
public class SpringConfiguration {

}
```

可以不写


```
public static void main(String[] args) {
    ApplicationContext ac = new AnnotationConfigApplicationContext(SpringConfiguration.class);
    AccountService accountService = (AccountService)ac.getBean( name: "accountService");
    List<Account> allAccount = accountService.findAllAccount();
    for (Account account : allAccount) {
        System.out.println(account);
    }
}
```

如果配置了

2. **@ComponentScan**

作用：用于**通过注解指定spring在创建容器时要扫描的包**

属性：value：它和base-Packages的作用是一样的，都是用于指定创建容器时要扫描的包。

我们使用此注解就等同于在xml中配置了:

<context:component-scan base-package="com.itheima"></context:component-scan>

注意：这里我是导入了两个包结构

```
@ComponentScan({"com.xiao","config"})
@PropertySources(@PropertySource("classpath:jdbcConfig.properties"))
@Import(value = {JdbcConfig.class})
public class SpringConfiguration {
```

3. **@Bean**

作用:用于把当前方法的返回值作为bean对象存入spring的ioc容器中

属性:name:用于指定bean的id。当不写时，默认值是当前方法的名称

细节：当我们使用注解配置方法时，如果方法有参数，spring框架会去容器中查找有没有可用的bea对象。

查找的方式和Autowired注解的作用是一样的

```
/**
 * 创建一个数据源并存入spring容器中(相当于)
 * <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
 *         <!--注入数据-->
 *         <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
 *         <property name="url" value="jdbc:mysql://localhost:3306/eesy_spring_account?useUnicode=true&am
 *         <property name="username" value="root"/>
 *         <property name="password" value="1234"/>
 *     </bean>
 * @return
 */
@Bean(name = "dataSource")
public DataSource createDataSource(){
    DruidDataSource dataSource = new DruidDataSource();
    dataSource.setUrl(url);
    dataSource.setPassword(password);
    dataSource.setDriverClassName(driver);
    dataSource.setUsername(username);
    return dataSource;
}
```


4. **@Import**

作用：用于导入其他的配置类

属性：value：用于指定其他配置类的字节码。

当我们使用Import的注解之后，有Import注解的类就父配置类，而导入的都是子配置类

```
 */
@ComponentScan({"com.xiao","config"})
@PropertySources(@PropertySource("classpath:jdbcConfig.properties
@Import(value = {JdbcConfig.class})
public class SpringConfiguration {
```

5. **@PropertySource**

作用：用于指定properties文件的位置

属性：value：指定文件的名称和路径。

关键字：classpath，表示类路径下

```
@PropertySources(@PropertySource("classpath:jdbcConfig.properties"))
```

```
package config;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Import;
import org.springframework.context.annotation.PropertySource;

/**
 * 该类是一个配置类，它的作用和bean.xml是一样的
 * spring中的新注解
 * Configuration
 *     作用：指定当前类是一个配置类
 *     细节：当配置类作为AnnotationConfigApplicationContext对象创建的参数时，该注解可以不
。
 * ComponentScan
 *     作用：用于通过注解指定spring在创建容器时要扫描的包
 *     属性：
 *         value：它和basePackages的作用是一样的，都是用于指定创建容器时要扫描的包。
 *                 我们使用此注解就等同于在xml中配置了:
 *                     <context:component-scan base-package="com.itheima"></context:compone
t-scan>
 *  Bean
 *     作用：用于把当前方法的返回值作为bean对象存入spring的ioc容器中
 *     属性:
 *         name:用于指定bean的id。当不写时，默认值是当前方法的名称
 *     细节：
 *         当我们使用注解配置方法时，如果方法有参数，spring框架会去容器中查找有没有可用的bea
对象。
 *         查找的方式和Autowired注解的作用是一样的
 *  Import
 *     作用：用于导入其他的配置类
 *     属性：
 *         value：用于指定其他配置类的字节码。
 *                 当我们使用Import的注解之后，有Import注解的类就父配置类，而导入的都是子配置类
 *  PropertySource
 *     作用：用于指定properties文件的位置
 *     属性：
 *         value：指定文件的名称和路径。
 *                 关键字：classpath，表示类路径下
 */
//@Configuration
@ComponentScan("com.itheima")
@Import(JdbcConfig.class)
@PropertySource("classpath:jdbcConfig.properties")
public class SpringConfiguration {
```

```
}
```

使用Junit单元测试：测试我们的配置

Spring整合junit的配置

1. 导入spring整合junit的jar(坐标)

2、使用Junit提供的一个注解把原有的main方法替换了，替换成spring提供的

@Runwith

3、告知spring的运行器，spring和ioc创建是基于xml还是注解的，并且说明位置

@ContextConfiguration

locations：指定xml文件的位置，加上classpath关键字，表示在类路径下

classes：指定注解类所在地位置

当我们使用spring 5.x版本的时候，要求junit的jar必须是4.12及以上

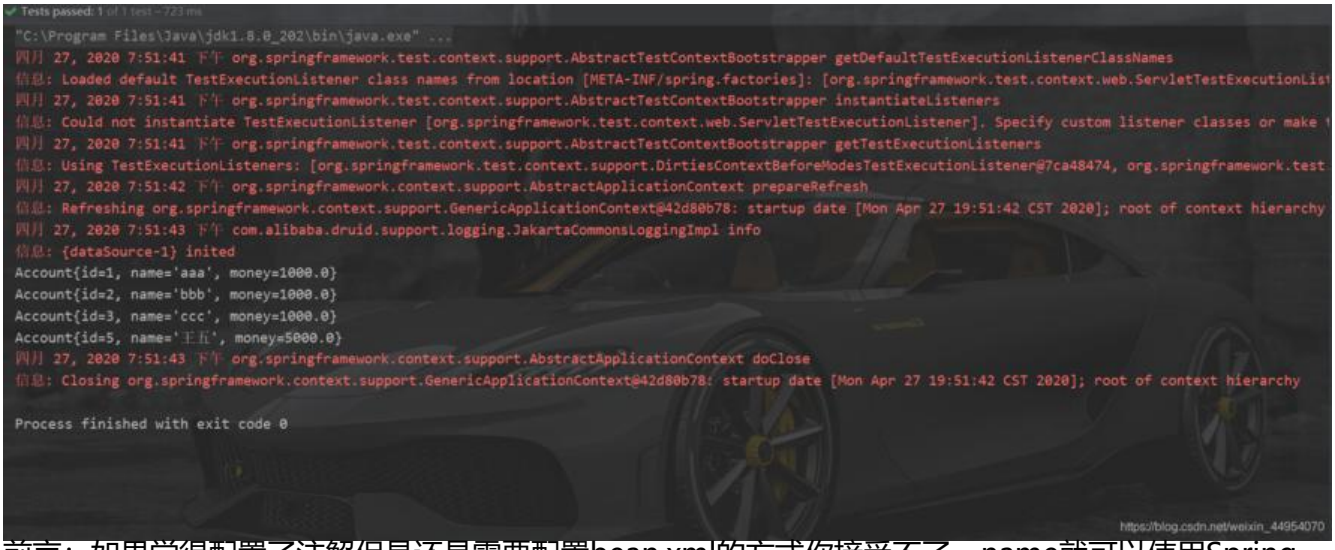
坐标

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>5.0.2.RELEASE</version>
</dependency>
```

```
package com.itheima.test;

import com.itheima.domain.Account;
import com.itheima.service.IAccountService;
import config.SpringConfiguration;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import java.util.List;

/**
 * 使用Junit单元测试：测试我们的配置
 * Spring整合junit的配置
 *      1、导入spring整合junit的jar(坐标)
 *      2、使用Junit提供的一个注解把原有的main方法替换了，替换成spring提供的
 *          @Runwith
 *      3、告知spring的运行器，spring和ioc创建是基于xml还是注解的，并且说明位置
 *          @ContextConfiguration
 *                  locations：指定xml文件的位置，加上classpath关键字，表示在类路径下
 *                  classes：指定注解类所在地位置
 *
 *   当我们使用spring 5.x版本的时候，要求junit的jar必须是4.12及以上
 */
```

```
**
 * @_PackageName:com.xiao.test
 * @_ClassName:JunitTest
 * @_Description:
 * @_Author：笑老二
 * @_data 2020/4/27 19:47
 */

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes = SpringConfiguration.class)
public class JunitTest {
    //导入serviceImpl资源
    @Autowired
    private AccountService as ;
    @Test
    public void findAllTest(){
        List<Account> allAccount = as.findAllAccount();
        for (Account account : allAccount) {
            System.out.println(account);
        }
    }
}
```

运行结果



前言：如果觉得配置了注解但是还是需要配置bean.xml的方式你接受不了，name就可以使用Spring的注解类来完成数据的输入以及，配置文件的输入。（简而言之就是所有配置文件全部通过注解的方式，用xml方式）

用到的几个注解(Spring新注解它作用和bean.xml一样)