Spring Test+JUnit整合使用

# spring test简介

在做spring相关测试时比较麻烦，如果只用JUnit测试，需要手动初始化一下applicationContext，效率比较低下，而且也有不足之处。具体如下：

## 导致多次Spring容器初始化问题

根据JUnit测试方法的调用流程，每执行一个测试方法都会创建一个测试用例的实例并调用setUp()方法。由于一般情况下，我们在setUp()方法 中初始化Spring容器，这意味着如果测试用例有多少个测试方法，Spring容器就会被重复初始化多次。虽然初始化Spring容器的速度并不会太 慢，但由于可能会在Spring容器初始化时执行加载Hibernate映射文件等耗时的操作，如果每执行一个测试方法都必须重复初始化Spring容 器，则对测试性能的影响是不容忽视的；使用Spring测试套件，Spring容器只会初始化一次！

## 需要使用硬编码方式手工获取Bean

在测试用例类中我们需要通过ctx.getBean()方法从Spirng容器中获取需要测试的目标Bean，并且还要进行强制类型转换的造型操作。这种乏味的操作迷漫在测试用例的代码中，让人觉得烦琐不堪；

使用Spring测试套件，测试用例类中的属性会被自动填充Spring容器的对应Bean，无须在手工设置Bean！

## 数据库现场容易遭受破坏

测试方法对数据库的更改操作会持久化到数据库中。虽然是针对开发数据库进行操作，但如果数据操作的影响是持久的，可能会影响到后面的测试行为。举个例子， 用户在测试方法中插入一条ID为1的User记录，第一次运行不会有问题，第二次运行时，就会因为主键冲突而导致测试用例失败。所以应该既能够完成功能逻 辑检查，又能够在测试完成后恢复现场，不会留下"后遗症"；

使用Spring测试套件，Spring会在你验证后，自动回滚对数据库的操作，保证数据库的现场不被破坏，因此重复测试不会发生问题！

## 不方便对数据操作正确性进行检查

假如我们向登录日志表插入了一条成功登录日志，可是我们却没有对t\_login\_log表中是否确实添加了一条记录进行检查。一般情况下，我们可能是打开 数据库，肉眼观察是否插入了相应的记录，但这严重违背了自动测试的原则。试想在测试包括成千上万个数据操作行为的程序时，如何用肉眼进行检查？

只要你继承Spring的测试套件的用例类，你就可以通过jdbcTemplate在同一事务中访问数据库，查询数据的变化，验证操作的正确性！

# spring test使用

## maven配置依赖

|  |
| --- |
| <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.12</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-test</artifactId>  <version>4.3.4.RELEASE</version>  </dependency> |

## java测试类中使用

@RunWith(SpringJUnit4ClassRunner.class)SpringJUnit支持，由此引入Spring-Test框架支持。

@ContextConfiguration(locations = "classpath:applicationContext.xml") 多个配置文件的话可以用数组表示{"applicationContext.xml","application  
Context1.xml"}。也可以使用一个配置文件，在配置文件中通过import引入其他配置文件。例如：@ContextConfiguration("/spring-context.xml")放在根路径下（即类路径下），然后<import resource="spring-dao.xml" />所有的配置文件和资源文件。

|  |
| --- |
| @RunWith(SpringJUnit4ClassRunner.**class**)  @ContextConfiguration("/spring-core.xml")  **public** **class** AspectTest {  @Autowired  MyBean myBean;  @Test  **public** **void** testAspect() {  myBean.sayHello("你好！");  }  }  注意：在spring-core.xml要开启component扫描  <context:component-scan base-package="com.imooc"/>  <!-- 启用ASpectJ自动代理-->  <aop:aspectj-autoproxy></aop:aspectj-autoproxy> |