链滴

[jeeplus]jeeplus 中的单元测试方法论

作者：PeterChu
原文链接：https://ld246.com/article/1559798254877
来源网站：链滴

[初学者级别]

在刚开始接触 jeeplus 时，就一直纠结怎么用自己之前最开始的单元测试方法，对 dao 层，service 的方法进行测试。

发现：

1. Junit 的单元测试在 jeeplus 框架下，使用时会出现需要有 ApplicationContext ，或者需要实现拟 请求对象等各种问题，但是 jeeplus 中的 SpringContextHolder 自己却不太会用，（或者是环境 jar 包的问题引起各种异常），不能进行常用的单元测试方法。

2. 刚开始按照在网上找到的一些博文中的方法，也是各种尝试，同样不能解决问题。后来就搁置了好好久。最近，实在是觉得还是非常有必要调通，会给项目的完成提供很大助力。所以，又开始找了许博文来试、请教大佬，最终，终于调通了。

3. 在后续的使用中，还存在着各种问题，比如 Shiro 的权限问题，还需要继续研究。

总结：

（仅简单说下大致的用法，内中详细自己还没掌握透彻，求大佬补充。）

1. pom.xml 中需要检查是否引用了正确的 jar 包，尤其注意一些环境、jar 包的版本问题。

另外，需要注意可能还有其他 jar 的问题。

```
<!-- TEST begin -->

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-test</artifactId>
      <version>${spring.version}</version>
    </dependency>



    <dependency>
      <groupId>javax.el</groupId>
      <artifactId>javax.el-api</artifactId>
      <version>2.2.4</version>
    </dependency>
    <dependency>
      <groupId>org.glassfish.web</groupId>
      <artifactId>javax.el</artifactId>
      <version>2.2.4</version>
    </dependency>


    <dependency>
      <groupId>org.apache.tomcat.embed</groupId>
```

```
        <artifactId>tomcat-embed-websocket</artifactId>
        <version>7.0.52</version>
        <scope>test</scope>
    </dependency>


<!-- TEST end -->
```

2. 创建单元测试基类

```
package com.jeeplus.core.junit;


import org.junit.runner.RunWith;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.web.WebAppConfiguration;

/**
 * Created by pc on 2019/6/3.
 */
@RunWith(SpringJUnit4ClassRunner.class)//表示整合JUnit4进行测试
@ContextConfiguration(locations={"classpath:spring/spring-context*.xml","classpath:/spring/
pring-mvc*.xml"})
@WebAppConfiguration    //声明以 web 形式进行测试
public class BaseJunit4Test {

}
```

3. 使用 Junit 开始单元测试。

a. 创建的测试类需要继承单元测试基类。

b. 可以正常使用 Spring 的注解方式注入 service 等。

c. 需要注意 Jeeplus 中使用的 Shiro 权限控制问题，很多 Jeeplus框架 中的方法可能有权限控制问题。

eg:

```
package com.jeeplus.modules.junittest;


import com.jeeplus.common.utils.IdGen;
import com.jeeplus.common.utils.SpringContextHolder;
import com.jeeplus.core.junit.BaseJunit4Test;
import com.jeeplus.modules.importxlsx.entity.yifenyiduanbiao.DYifenyiduanbiao;
import com.jeeplus.modules.importxlsx.entity.zhuanyexinxibiao.DZhuanyexinxibiao;
import com.jeeplus.modules.importxlsx.service.zhuanyexinxibiao.DZhuanyexinxibiaoService;
import com.jeeplus.modules.levelformajorsfordetails.entity.LevelForMajorsDetails;
import com.jeeplus.modules.levelformajorsfordetails.entity.LevelForMajorsForDetails;
import com.jeeplus.modules.levelformajorsfordetails.mapper.LevelForMajorsDetailsMapper;
import com.jeeplus.modules.levelformajorsfordetails.service.LevelForMajorsDetailsService;
import com.jeeplus.modules.levelformajorsfordetails.service.LevelForMajorsForDetailsService;
import com.jeeplus.modules.sys.entity.User;
import org.apache.ibatis.annotations.Param;
```

```
import org.junit.Before;
import org.junit.Test;
import org.springframework.beans.factory.annotation.Autowired;

import javax.persistence.Basic;
import java.util.Date;
import java.util.List;

/**
 * Created by pc on 2019/6/3.
 */
public class JunitTestERecode extends BaseJunit4Test{

    @Autowired
    LevelForMajorsDetailsService levelForMajorsDetailsService;
    @Autowired
    LevelForMajorsForDetailsService levelForMajorsForDetailsService;

    @Autowired
    DZhuanyexinxibiaoService dZhuanyexinxibiaoService;


    /**
     * 需要注意的是:
     * shiro 的权限问题，Service 中大多数查询方法都有权限控制。测试时会遇到 shiro 异常。
     */

    @Before
    public void setDate(){

    }

    //测试 LevelForMajorsDetails 中的按更新时间删除记录
    @Test
    public void testDeleteERecord(){
        Date date = new Date();
        LevelForMajorsDetails levelForMajorsDetails = new LevelForMajorsDetails();
        levelForMajorsDetails.setUpdateDate(date);
        levelForMajorsDetails.setId(IdGen.uuid());
        levelForMajorsDetails.setIsNewRecord(true);

        if(levelForMajorsDetails != null){
            System.out.println("date:"+ date);
            levelForMajorsDetailsService.save(levelForMajorsDetails);
            System.out.println("保存成功！");
        } else {
            System.out.println("levelForMajorsDetails 为null");
        }

        List<LevelForMajorsDetails> LevelForMajorsDetailsSaved  = levelForMajorsDetailsService
findByUpdate(levelForMajorsDetails);

        for(LevelForMajorsDetails l : LevelForMajorsDetailsSaved){
            System.out.println("测试 getUpdateDate 方法中按时间获取到的记录的时间："+l.getUpda
```

```
eDate());
        }

        LevelForMajorsDetails newLevelForMajorsDetails = new LevelForMajorsDetails();
        newLevelForMajorsDetails.setUpdateDate(date);
        System.out.println("delect 的 date:"+ date);
        levelForMajorsDetailsService.deleteERecord(newLevelForMajorsDetails);
        System.out.println("执行了删除！");

    }


    //测试 LevelForMajorsForDetails 中的按更新时间删除记录
    @Test
    public void testLevelForMajorsDetailsServiceGet(){
        Date date = new Date();
        LevelForMajorsForDetails levelForMajorsForDetails = new LevelForMajorsForDetails();
        levelForMajorsForDetails.setUpdateDate(date);

        if(levelForMajorsForDetails != null){
            System.out.println("date:"+ date);
            levelForMajorsForDetailsService.save(levelForMajorsForDetails);
            System.out.println("保存成功！");
        } else {
            System.out.println("levelForMajorsDetails 为null");
        }

        List<LevelForMajorsForDetails> LevelForMajorsForDetailsSaved  =
        levelForMajorsForDetailsService.findByUpdate(levelForMajorsForDetails);

        for(LevelForMajorsForDetails l : LevelForMajorsForDetailsSaved){
            System.out.println("测试 getUpdateDate 方法中按时间获取到的记录的时间："+l.getUpda
eDate());
        }

        LevelForMajorsForDetails newLevelForMajorsForDetails = new LevelForMajorsForDetails()

        newLevelForMajorsForDetails.setUpdateDate(date);
        System.out.println("delect 的 date:"+ date);
        levelForMajorsForDetailsService.deleteERecord(newLevelForMajorsForDetails);
        System.out.println("执行了删除！");


    }


    //测试 dZhuanyexinxibiaoService 中的 setAlreadyTransform()方法
    @Test
    public void testDZhuanyexinxibiaoService(){
        int count = dZhuanyexinxibiaoService.setAlreadyTransform();
        System.out.println("成功修改了"+ count +"条记录！");

    }
```

```
    //测试添加 beiyong20 = "" 时的查询结果是否为查询到数据库中 beiyong20=null 的记录
    @Test
    public void testFindListByBeiyong20(){
        DZhuanyexinxibiao dZhuanyexinxibiao = new DZhuanyexinxibiao();
        dZhuanyexinxibiao.setBeiyong20("");
        List<DZhuanyexinxibiao> dZhuanyexinxibiaoList =  dZhuanyexinxibiaoService.findListBy
eiyong20(dZhuanyexinxibiao);

        System.out.println(dZhuanyexinxibiaoList.size());
    }

}
```

4. 好吧，还是忍受不住没有Log4j 了，找了篇大佬博文，照着配置下。

原博文： Junit单元测试使用log4j输出日志

Junit+spring+log4j整合之所以麻烦，是因为spring与log4j的整合，是放在web.xml里的，随tomcat启动后，spring才会加载log4j，而用junit测试是不需要tomcat启动的，所以Junit与log4j的整合才较费劲。Junit使用spring时，若spring没加载到log4j就会报以下警告：

```
1.  log4j:WARN No appenders could be found for logger(org.springframework.test.context.jun
t4.SpringJUnit4ClassRunner).
2.  log4j:WARN Please initialize the log4j system properly.
3.  log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

推荐方法

新建JUnit4ClassRunner类：

```
1.  public class JUnit4ClassRunner extends SpringJUnit4ClassRunner {
2.      static {
3.          try {
4.              Log4jConfigurer.initLogging("classpath:com/config/log4j.properties");
5.          } catch (FileNotFoundException ex) {
6.              System.err.println("Cannot Initialize log4j");
7.          }
8.      }
9.      public JUnit4ClassRunner(Class<?> clazz) throws InitializationError {
10.         super(clazz);
11.     }
12.  }
```

引用此类：

```
1.  @RunWith(JUnit4ClassRunner.class)
2.  @ContextConfiguration(locations = "classpath:com/config/springConfig.xml")
3.  @Transactional
4.  @TransactionConfiguration(transactionManager = "transactionManager", defaultRollback
 true)
5.  public class TestHibernate {
6.      ...
7.  }
```

这样，在启动Junit测试时，spring就会加载log4j了。而且保持了灵活性。

PS：Junit加载spring的runner（SpringJUnit4ClassRunner）要优先于spring加载log4j，因此用普通方法，无法实现spring先加载log4j后被Junit加载。所以我们需要新建JUnit4ClassRunner类修改SpringJUnit4ClassRunner加载log4j的策略。这样加载log4j就会优先于加载spring了。

采用前辈推荐的方法：

```
package com.jeeplus.core.junit;


import org.junit.runners.model.InitializationError;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.util.Log4jConfigurer;

import java.io.FileNotFoundException;

/**
 * Created by pc on 2019/6/6.
 */
public class MyJUnit4ClassRunner extends SpringJUnit4ClassRunner {
    static {
        try {
            Log4jConfigurer.initLogging("classpath:properties/log4j.properties");//F:\pc\CEDS 本地
VN\jeeplus\src\main\resources\properties\log4j.properties properties/log4j.properties
        } catch (FileNotFoundException ex) {
            System.err.println("Cannot Initialize log4j");
        }
    }
    public MyJUnit4ClassRunner(Class<?> clazz) throws InitializationError {
        super(clazz);
    }
}
```

修改原 BaseJunit4Test 中的 @RunWith 注解中类

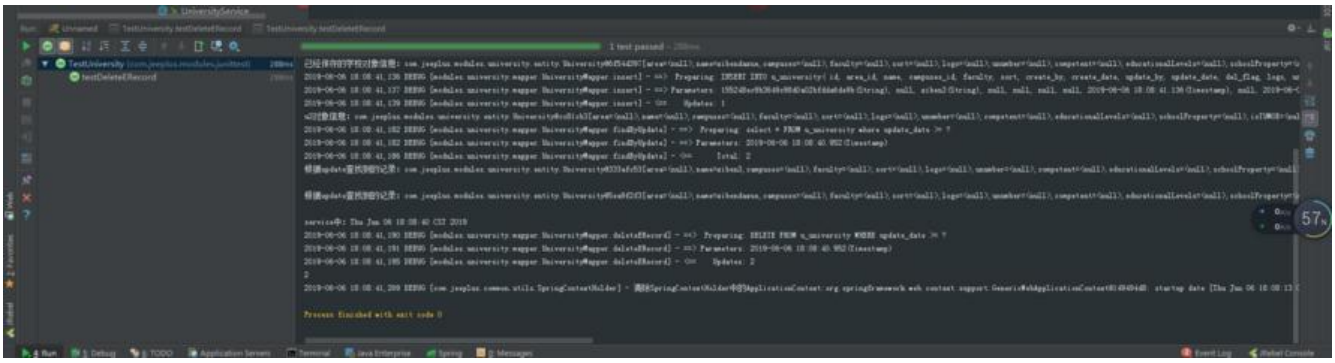
```
package com.jeeplus.core.junit;

import org.junit.runner.RunWith;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.web.WebAppConfiguration;

/**
 * Created by pc on 2019/6/3.
 */
@RunWith(MyJUnit4ClassRunner.class)//表示整合JUnit4进行测试
@ContextConfiguration(locations={"classpath:spring/spring-context*.xml","classpath:/spring/
pring-mvc*.xml"})
@WebAppConfiguration    //声明以 web 形式进行测试
public class BaseJunit4Test {

}
```

这样就可以了。最终效果：



原文链接：[jeeplus]jeeplus 中的单元测试方法论