# 1 需求分析

# 1.1 需求描述

从Blade的架构来看，Blade提供了相当充足的功能用于简化Java开发。但从开发过程来看，Blade尚有不足。其中最为明显的地方在于Blade并未针对测试提供解决方案。世界上不存在没有漏洞的系统，无一例外地都会在运行时都会出现错误、异常以及其他非期待的运行结果。想要尽可能的减少运行时出现的问题，提前的测试是开发中必不可少的一环。软件测试能够提前发现生产环境中可能出现的问题并提前处理，避免财产、生命的损失。根据软件工程和软件测试的规范，开发人员应先进行单元测试。单元测试中的黑盒测试将测试用例的输入作为模块的输入，将预期结果与和模块输出相比较来判断模块工作是否正常。

人工对一整个系统的测试时非常耗时费力的，因此开发人员可以选用各种常用的测试框架对程序进行测试。JUnit是Java语言中最为广泛使用和强力的测试框架，其能够提供Java运行必须的测试环境，并运行指定的待测试方法。JUnit能够采用断言的方式判断预期结果是否与模块输出结果相同，并循环或批量对不同的方法进行测试。这给开发者测试提供了极大的便利。

Blade是基于IOC技术的JavaWeb框架，其特性就在于开发者无需声明变量即可使用。但IOC必须在Blade框架完全启动才能生效，JUnit无法自动生成IOC环境，因此无法对基于Blade框架编写的应用程序进行测试。这使得Blade开发的应用程序变得不安全，我们小组认为易用的JavaWeb框架不应该有此缺陷，因此致力于完善Blade框架在测试方面的不足。

# 1.2 RUCM建模

测试不应该加重开发者的工作负担，因此借助IOC容器应该同样能够提供简单有效的测试方案。结合Spring的测试方案，使用RUCM描述测试用例的行为模型如图 1至图 2所示：

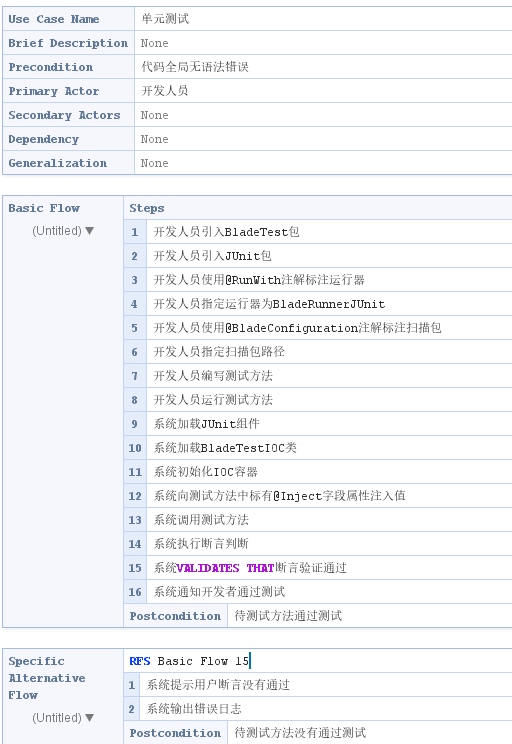


图 1 单元测试用例RUCM（a）



图 2 单元测试用例RUCM（b）

# 1.3 需求目标

根据需求描述，小组计划给予Blade的基本结构设计一套测试组件BladeTest，期望能够满足开发人员的日常测试功能。BladeTest应满足如下要求：

1）开发人员无需关心IoC容器细节，可在测试环境内直接使用IOC容器。

2）开发人员能够在测试环境内获取IOC容器内容

3）IOC容器在单元测试开始和结束时自动初始化

4）测试可以批量进行

5）尽可能减少与Blade的重复代码，保证同步更新的可能性

6）添加其他方便开发人员测试的功能并保证兼容性

# 2 可行性分析

作为一个Java语言开发的框架，Blade本应能与JUnit之间共同合作。但由于IOC容器的缺失，JUnit与Blade之间无法正常的交互。小组认为手动在JUnit运行时提供适合Blade使用的IOC容器能够解决该问题。

小组参考了Spring的Spring是JavaWeb框架的领头羊，其在SpringTest子项目中提供了基于JUnit的测试方案。JUnit在运行之前会通过@RunWith注解获取Spring默认IOC容器，该IOC进针对测试类进行管理，足够支持待测试方法的运行和检查。因此小组认为这是一条可行的技术方案。

# 2 .1测试用例类图

测试用例基于BladeTest类提供前置的IOC容器初始化，并依赖Blade的IOC组件进行执行。JUnit提供了RunWith接口，BladeTest在实现了该接口后能够预加载JUnit的环境，并提供IOC容器的实现。绘制类图如图 3所示。

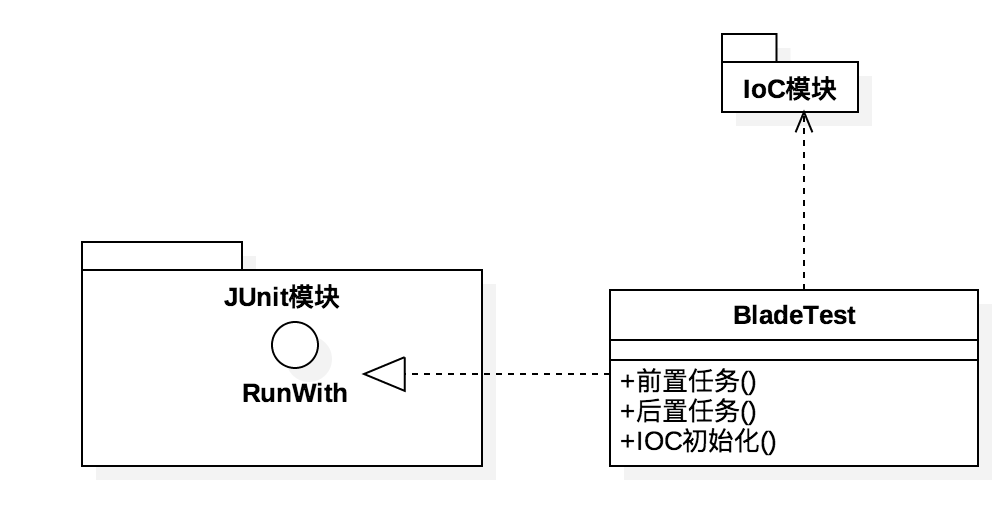


图 3 测试用例类图

# 6. 模块划分

考虑到小组的工作是在JUnit和Blade之前添加连接，因此我们采用适配器模式进行开发。因此划分模块如下：

1）BladeTestIOC类是Blade框架的对象适配器，能够提供初始化接口、清零接口供JUnit调用。保证JUnit无需知道Blade工作的细节。

2）BladeTestIOC内部通过Blade框架的启动器对IOC容器进行初始化。

3）BladeRunner4JUnit实现了IOC容器的初始化的清零，继承了JUnitRunner接口，能够向JUnit注入预操作。

4）BladeConfiguration注解，用于从BladeTest中获取配置参数（扫描包位置等）

# 7. 测评

# 7.1 测评需求

测试需要对如下方面进行测试：

1. 在引入BladeTest的Jar包后，JUnit可以通过BladeRunner4JUnit运行

2. JUnit的测试方法中，能够获取IOC容器中的对象

3. JUnit测试类中可以使用@Inject注解获取对象

# 7.2 测试用例