


# 用junit5编写一个简易接口自动化测试框架

 antony已经被占用 

关注

2019.11.05 22:23:54 字数 1,871 阅读 212

## 技术点:

最近笔者在尝试基于应用日志来自动生成测试用例。这其中就需要一个配套的简易测试框架。梳理了一下，其中的技术点有：

- 0.使用csv文件来定义测试用例及步骤
- 1.使用自定义测试注解来定义测试用例（参考ZeroCode）
- 2.使用JUnit5提供的extension机制来实现测试执行
- 3.使用简单工厂类提供执行驱动
- 4.使用OpenCsv来实现解析
- 5.使用Lombok来定义Java Bean
- 6.使用JUnit5提供的参数化测试解决方案junit-jupiter-params来实现测试用例集

## 来自ZeroCode的参考

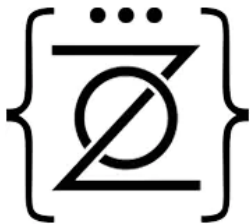


image.png

ZeroCode是一个轻量级的开源测试框架。它通过使用JSON或者YAML文件格式来定义测试用例，进而让测试用例的编写变得更为容易。

以下是其github项目首页提供的案例

```
1  @Test
2  @Scenario("test_customer_get_api.yml")
3  public void getCustomer_happyCase(){
4      // No code goes here. This remains empty.
5  }
```

其中test\_customer\_get\_api.yml中就描述了这个接口测试用例的全部要素，具体如下：

```
1  ---
2  url: api/v1/customers/123
3  method: GET
4  request:
5    headers:
6      Content-Type: application/json
7  retry:
8    max: 3
9    delay: 1000
10 verify:
11   status: 200
12   headers:
13     Content-Type:
14       - application/json; charset=utf-8
15   body:
16     id: 123
```



 antony已经被占用 

关注

总资产88 (约7.87元)

三种mock注入方式了解一下  
阅读 69

spy  
阅读 2

Void ?  
阅读 2

### 推荐阅读

10\_Pytest框架  
阅读 1,345

中高级测试工程师68道面试题  
阅读 8,185

项目-无侵入代码方式使用Redis实现缓存功能  
阅读 3,279

python学习科普--文件、异常、单元测试  
阅读 79

Prometheus + Grafana 监控  
SpringBoot  
阅读 2,988

这个YAML文件中包括了http接口测试中所需要的请求（含url\head、类型）以及返回、验证模式等内容，是一个不错的用例DSL。本身这是一个很好的开源测试框架，涵盖的测试类型也比较多，参与维护的人员和更新速度也不错。

A community-developed, free, open source, declarative API automation and load testing framework built using Java JUnit core runners for Http REST, SOAP, Security, Database, Kafka and much more. It enables to create and maintain test-cases with absolute ease.

### 实际项目中的需求

在实际的测试过程中，对于文本格式的测试用例，往往有以下的需求：

- 1. 测试用例的步骤描述通常是自定义的，而不是根据工具提供的DSL来编写。
- 2. 用例编写尽可能少一些冗余的内容，以便节约用例编写时间。  
例如在前述接口测试案例中的head,Content-Type等等，在某个系统的接口规范中，往往都是规定了固定格式的。
- 3. 可能的话，@Test之类的Java代码也不用写了  
测试人员只写用例文件，框架通过扫码文件目录和文件来执行用例。

为了实现上述需求，这就要求根据测试的特点，来定制一个类似的简易测试框架。

### 使用文件来定义测试用例和步骤

当设计一个自动化测试用力框架时，有一个很重要的三联问问题:

如何定义一个用例？如何定义用例的步骤？如何定义一个用例集？

在本案例中，我们约定

- 一个文件（csv）是一个用例
- 文件中的一行是用例的一个步骤
- 包含若干文件的目录，组成了一个用例集

至于用csv文件来作为用例的载体，而不是json/yaml等更新的文件类型，或者xml/excel等传统文件类型，主要是基于以下两方面考虑

- 接口自动化测试，尤其是面向业务功能的测试，其请求、入参、出参的结构相对固定。
- csv擅长表达结构固定的数据内容，且格式冗余最小。

因此，如果以前述ZeroCode的接口为例，一个简单的接口自动化测试的用例格式可以是

n u m	ty p e	url	par am s	response
1	g et	"api/v1/customers/123"		{"id": 123,"type": "Premium High Value","addresses": [{"type":"home","line1":"10 Random St"}]}

读者可能会问，那么head,content-type这些不要了么？ status code 都等于200么？ 实际项目中经常用到的token怎么没有体现？ 等等问题。

每个框架都有其应用场合。上面这样的框架，主要是应用于业务层面的测试，而不是接口自身的鲁棒性测试等场合。这样简单的格式，也非常适合不太能写代码的同学来写自动化测试用例。

## 测试样例

以下是完成框架开发之后的测试用例样例。sample.csv中是用前述表格中的内容。

```
1 package com.demo.junit5.extensions;
2 import org.junit.jupiter.api.Test;
3 import com.demo.junit5.Scenario;
4
5 class ScenarioTest {
6     @Test
7     @Scenario(value="\\.\\tests\\demo1\\sample.csv")
8     public void sampleTest() {
9     }
10 }
11
12
```

## 自定义注解

我们来看下测试用例中的注解的具体实现：

```
1 package com.demo.junit5;
2 import java.lang.annotation.*;
3 import org.junit.jupiter.api.extension.ExtendWith;
4 @Target({ElementType.METHOD})
5 @Retention(RetentionPolicy.RUNTIME)
6 @ExtendWith(ScenarioExtension.class)
7 public @interface Scenario {
8     String[] value() default "";
9 }
```

通过@Scenario 在某个方法上的注解，可以调用

@ExtendWith(ScenarioExtension.class)

中的具体功能。这也是JUnit5提供的一种回调机制，来扩展JUnit5测试框架的功能。

## 具体的Extension

JUnit5提供了非常友好的扩展性，最常用的是Before/After配套的一些Callback接口上，如下图所示：

image.png

这里我们就使用了一个BeforeTestExecutionCallback的接口来进行扩展，在被注解的用例执行之前，JUnit5会首先调用该接口，实现自定义的功能。

```
1 package com.demo.junit5;
2
3 import java.io.IOException;
4 import java.io.Reader;
5 import java.nio.file.Files;
6 import java.nio.file.Paths;
7 import java.util.Iterator;
8
9 import org.junit.jupiter.api.extension.BeforeTestExecutionCallback;
10 import org.junit.jupiter.api.extension.ExtensionContext;
11 import org.junit.platform.commons.util.AnnotationUtils;
12 import com.demo.junit5.bean.TestStep;
13 import com.demo.junit5.runner.Runner;
14 import com.demo.junit5.runner.RunnerFactory;
15 import com.opencsv.bean.CsvToBean;
16 import com.opencsv.bean.CsvToBeanBuilder;
17 class ScenarioExtension
18     implements BeforeTestExecutionCallback {
19     private Runner runner = RunnerFactory.getRunner("");
20     //实际项目中一般通过配置来传入具体的runner类型。这里只是一个Dummy样例。
21     // EXTENSION POINTS
22     public void beforeTestExecution(ExtensionContext context) throws Exception{
23         Scenario scenario = AnnotationUtils.findAnnotation(context.getRequiredTestMethod(), Scenario.class);
24         if (scenario == null) {
25             scenario = AnnotationUtils.findAnnotation(context.getRequiredTestClass(), Scenario.class);
26         }
27         for(String v:scenario.value()) {
28             runCase(runner,v);
29         }
30     }
31     private static void runCase(Runner runner,String testCase) throws IOException {
32         Reader reader = Files.newBufferedReader(Paths.get(testCase));
33         CsvToBean<TestStep> csvToBean = new CsvToBeanBuilder<TestStep>(reader)
34             .withType(TestStep.class)
35             .withIgnoreLeadingWhiteSpace(true)
36             .withSeparator(',')
37             .build();
38         Iterator<TestStep> csvIterator = csvToBean.iterator();
39         while(csvIterator.hasNext()) {
40             TestStep testStep = csvIterator.next();
41             runner.run(testStep);
42         }
43     }
44 }
45
46 }
```

通过实现BeforeTestExecutionCallback 接口中的beforeTestExecution方法，可以将传入的用例文件内容（测试步骤）进行解析，并交给一个Runner进行执行。

## 执行器Runner相关

再来看一下Runner接口

```
1 package com.demo.junit5.runner;
2 import com.demo.junit5.bean.TestStep;
3 public interface Runner {
4     public void run(TestStep testStep);
5 }
```

写下你的评论...

评论0

赞

...

```
1 package com.demo.junit5.runner;
2
3 import com.alibaba.fastjson.JSON;
4 import static org.assertj.core.api.Assertions.assertThat;
5 import com.demo.junit5.bean.TestStep;
6
7 import lombok.extern.slf4j.Slf4j;
8 @Slf4j
9 public class MockRunner implements Runner {
10     public void run(TestStep testStep) {
11         log.info("mock runner called");
12         log.info("step #{}", testStep.getNum());
13         log.info(JSON.toJSONString(testStep));
14         assertThat(testStep.getResponse()).isEqualToIgnoringCase(testStep.getResponse());
15     }
16 }
```

实际工作中可以使用Rest-Assured等工具来实现HTTP接口的调用，并进行结果的验证。如果是TCP等类型的接口，换一种具体实现即可。

有经验的读者可能已经想到了，这就是一个典型的工厂设计模式的使用场景。我们用一个简单工厂作为示例：

```
1 package com.demo.junit5.runner;
2 public class RunnerFactory {
3     public static Runner getRunner(String runner) {
4         return new MockRunner();
5     }
6 }
```

目前这个工厂只提供MockRunner一种实现。

## 业务Bean - TestStep

测试步骤的Bean 如下：

```
1 package com.demo.junit5.bean;
2 import com.opencsv.bean.CsvBindByName;
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6 @Data
7 @NoArgsConstructor
8 @AllArgsConstructor
9 public class TestStep {
10     @CsvBindByName(column="num")
11     private int num;
12     @CsvBindByName(column="type")
13     private String type;
14     @CsvBindByName(column="url")
15     private String url;
16     @CsvBindByName(column="params")
17     private String params;
18     @CsvBindByName(column="response")
19     private String response;
20 }
```

通过Lombok极大地简化了代码。而通过opencsv,可以极为方便地实现csv文件和bean之间地转换。

```
1 CsvToBean<TestStep> csvToBean = new CsvToBeanBuilder<TestStep>(reader)
2     .withType(TestStep.class)
3     .withIgnoreLeadingWhiteSpace(true)
4     .withSeparator(',')
5     .build();
```

写下你的评论...

 评论0

 赞

...

## 小节

至此，一个简单的自定义文件的测试框架就构建完毕了。总结一下feature:

- 0.使用csv文件来定义测试用例及步骤
- 1.使用自定义测试注解来定义测试用例（参考ZeroCode）
- 2.使用Junit5提供的extension机制来实现测试执行
- 3.使用简单工厂类提供执行驱动
- 4.使用OpenCsv来实现解析
- 5.使用Lombok来定义Java Bean

至于参数化构建，我们将在后续完成。

[未完待续]

## TODO- 测试用例定义及读取-csv文件和对象Bean -done

- 0.使用csv文件来定义测试用例及步骤
- 1.使用OpenCsv来实现解析
- 2.使用Lombok来定义Java Bean

## 如何标识参数化用例？

使用注解

@ParameterizedTest

## 入参有哪些类型？

- @ValueSource
- @EnumSource
- @CsvSource
- @CsvFileSource
- @MethodSource

## 如何自定义用例名称

The following placeholders are available when customizing the display name:

{index} will be replaced with the invocation index – simply put, the invocation index for the first execution is 1, for the second is 2, and so on

{arguments} is a placeholder for the complete, comma-separated list of arguments

{0}, {1}, ... are placeholders for individual arguments

## 各类型入参案例

0人点赞 >

软件测试/测试框架

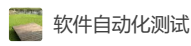
"小礼物走一走，来简书关注我"

赞赏支持

还没有人赞赏，支持一下

写下你的评论... 评论0 赞 ...

被以下专题收入，发现更多相似内容



推荐阅读

更多精彩内容 >

### Spring 5.0.0框架介绍\_中英文对照\_3.4

文章作者: Tyan博客: noahsnail.com 3.4 Dependencies A typical ente...

 SnailTyan 阅读 2,280 评论 2 赞 7

### Lua 5.1 参考手册

Lua 5.1 参考手册 by Roberto Ierusalimschy, Luiz Henrique de F...

 苏黎九歌 阅读 7,840 评论 0 赞 39

### pySpark 中文API (2)

pyspark.sql模块 模块上下文 Spark SQL和DataFrames的重要类: pyspark.sql...

 mpro 阅读 5,041 评论 0 赞 8


### 编写一个框架的详细步骤

转自:[http://blog.csdn.net/liu88010988/article/details/5154...

 hackywit 阅读 2,686 评论 0 赞 26

### 谈今天的运动会

今天下午，我们大五中的运动会开幕了，今时不同往日，有钱了！学校弄了个航拍机，虽然就一个，但是也算震撼了我们这群从小...

 尼卡NANA 阅读 142 评论 0 赞 1

