Junit3.8.2设计模式浅谈之模板方法模式

作者：changming
原文链接：https://ld246.com/article/1471154805103
来源网站：链滴

模板方法模式(Template Method)
对于某个逻辑，其操作流程是固定不变的，变化的仅仅是内部的一部分逻辑，比如对于数据库操，建立连接，释放连接，开启关闭事务等等都是每次操作数据库的固定流程，其中变化的仅仅是操作录数据的部分；
在Junit中，单元测试前的准备工作setUp，单元测试开始run，以及测试完成的清理工作tearDown，都是流程性工作，这种场景我们可以使用模板方法模式，下面看代码。

```
public void runBare() throws Throwable {
        Throwable exception= null;
        setUp();
        try {
            runTest();
        } catch (Throwable running) {
            exception= running;
        }
        finally {
            try {
                tearDown();
            } catch (Throwable tearingDown) {
                if (exception == null) exception= tearingDown;
            }
        }
        if (exception != null) throw exception;
}
    /**
     * Override to run the test and assert its state.
     * @exception Throwable if any exception is thrown
     */
    protected void runTest() throws Throwable {
        assertNotNull(fName); // Some VMs crash when calling getMethod(null,null);
        Method runMethod= null;
        try {
            // use getMethod to get all public inherited
            // methods. getDeclaredMethods returns all
            // methods of this class but excludes the
            // inherited ones.
            runMethod= getClass().getMethod(fName, (Class[])null);
        } catch (NoSuchMethodException e) {
            fail("Method \""+fName+"\" not found");
        }
        if (!Modifier.isPublic(runMethod.getModifiers())) {
            fail("Method \""+fName+"\" should be public");
        }



    try {
        runMethod.invoke(this, (Object[])new Class[0]);
    }
    catch (InvocationTargetException e) {
        e.fillInStackTrace();
        throw e.getTargetException();
    }
    catch (IllegalAccessException e) {
        e.fillInStackTrace();
        throw e;
```

```
    }

}</pre>

<p><span>执行到TestCase的runBare()方法，该方法就是一个模板方法，其中逻辑骨架已经确定即setUp和runTest以及tearDown方法，这三个方法对于每个测试用例都会依次执行，执行时会调用个测试用例子类实现的具体实现逻辑。</span></p>
```

原文链接：Junit3.8.2设计模式浅谈之模板方法模式