**软件测试 实验报告 — EasyMock**

# 1、实验目的

1. 使用 EasyMock+Junit 对附件中的LoginServlet 进行测试：分别对登录成功和失败进行测试， 测试方法名分别为testLoginSuccess()和 testLoginFailed()学习并掌握 Junit 的使用方法；
2. 理解应用EasyMock 的测试流程和用途；
3. 学会编写合理的测试用例、使用合适的方法进行测试。

# 2、测试范围

观察 LoginServlet 类中的方法后，分别对登录成功和失败进行测试。登录输入条件为用户名和密码，对应代码中的req.getParam

eter("username") 和 req.getParameter("password")。

对doPost()方法测试用例的等价类划分

表 1 doPost()方法测试用例等价类划分

|  |  |  |  |
| --- | --- | --- | --- |
| 输入/输出 | | 有效等价类 | 无效等价类 |
| 输入条件 | 用户名 | ①非空字符串  ②等于 admin | ③输入为空  ④输入字符串不等于 admin |
| 密码 | ⑤非空字符串  ⑥等于 123456 | ⑦输入为空  ⑧输入字符串不等于 123456 |

|  |  |  |  |
| --- | --- | --- | --- |
| 输出条件 | 登录成功 | ⑨登录成功 | 无 |
| 登录失败 | ⑩登录失败 | 无 |

根据上述分析情况，可以通过等价类分析 doPost()方法的测试用例。大致设计以下测试用例，如表 2 所示：

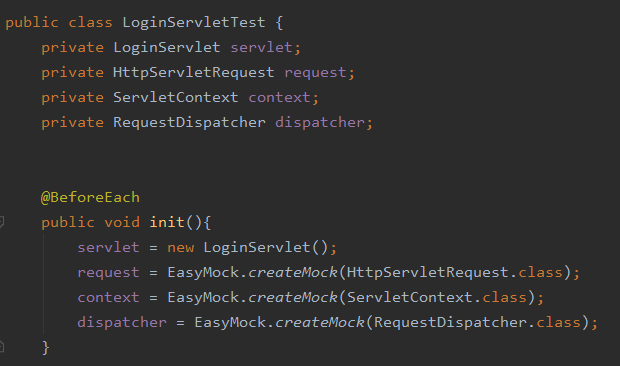
表 2 doPost ()测试用例设计

|  |  |  |  |
| --- | --- | --- | --- |
| 测试方法 | 测试编号 | 输入 | 覆盖等价类 |
| doPost() | 1 | (“admin”, “123456”) | ①、②、⑤、  ⑥、⑨ |
| 2 | (“admin”,  “1234”) | ⑧、⑩ |
| 3 | (“admin”,  “”) | ⑧ |
| 4 | (“admin”,  NULL) | ⑦ |
| 5 | (“rsq”,  “123456”) | ④ |
| 6 | (“”,  “123456”) | ④ |
| 7 | (NULL,  “123456”) | ③ |
| 8 | (NULL,  NULL) | ⑩ |

# 3、测试过程

* 1. **初始化测试代码**

查看LoginServlet.java 源代码后，分析得到需要mock 的接口，在init()方法中进行 mock。代码如下：

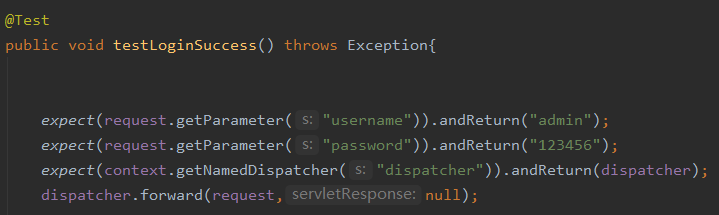


# 登录成功测试

* + 1. **测试流程**

**①设置期望值**

对request.getParameter()等方法设置期望值，代码如下：



# ②激活mock 对象

使用replay 方法，激活request、context 和dispatcher，代码如下：



**③测试并验证**

为了让 getServletContext()方法返回创建的 ServletContext Mock 对象，定义一个匿名类并覆写 getServletContext()方法。最后， 测试doPost()方法，并通过 verify 验证。代码如下：

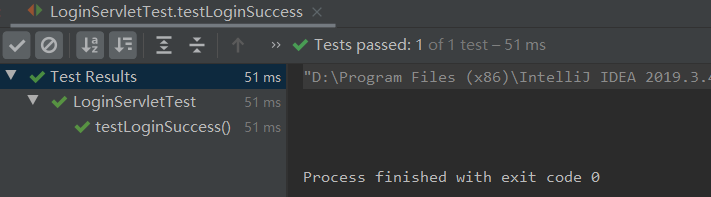


# 测试结果

1. **测试编号 1**
   1. **测试代码**



* 1. **测试结果**



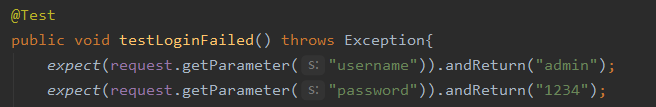
由反馈信息可知，通过测试。用户名为“admin”、密码为“123456”符合登录条件，成功登录，与预期一致。

# 登录失败测试

* + 1. **测试流程**

**①设置期望值**

登录失败相关代码中只涉及对request.getParameter()方法设置期望值，代码如下：



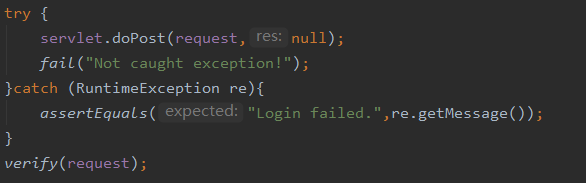
# ②激活mock 对象

使用replay 方法，激活request；



# ③测试并验证

测试doPost()方法，并通过 verify 验证。检测是否抛Runtime Exception 异常，代码如下：

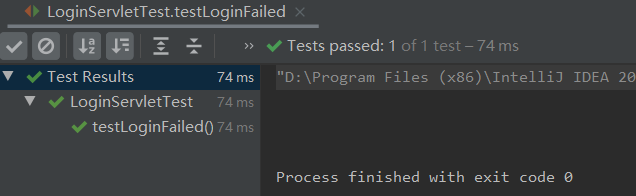


# 测试结果

1. **测试编号 2**
   1. **测试代码**



* 1. **测试结果**



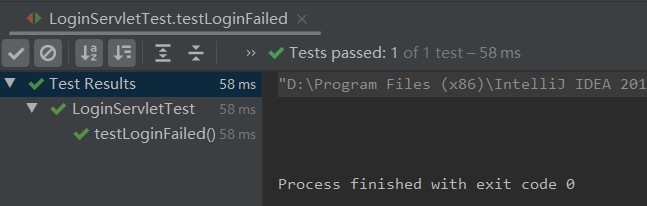
由反馈信息可知，通过测试。用户名为“admin”、密码为“1234”不符合登录条件，抛出 RuntimeException 异常，与预期一致。

# 测试编号 3

* 1. **测试代码**



* 1. **测试结果**



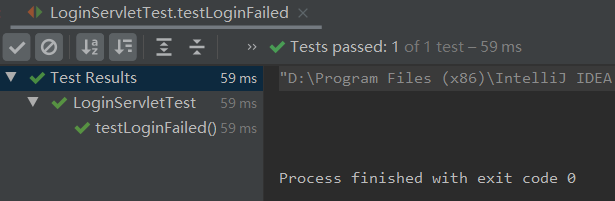
由反馈信息可知，通过测试。用户名为“admin”、密码为“”不符合登录条件，抛出 RuntimeException 异常，与预期一致。

# 测试编号 4

* 1. **测试代码**



* 1. **测试结果**



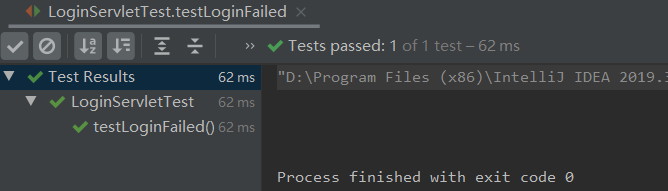
由反馈信息可知，通过测试。用户名为“admin”、密码为空，不符合登录条件，抛出 RuntimeException 异常，与预期一致。

# 测试编号 5

* 1. **测试代码**



* 1. **测试结果**



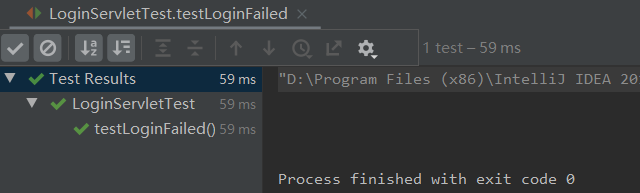
由反馈信息可知，通过测试。用户名为“rsq”、密码“123456”，不符合登录条件，抛出 RuntimeException 异常，与预期一致。

# 测试编号 6

* 1. **测试代码**



* 1. **测试结果**



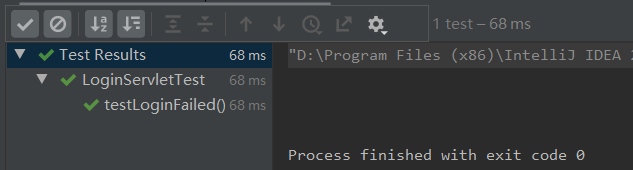
由反馈信息可知，通过测试。用户名为“”、密码“123456”，不符合登录条件，抛出 RuntimeException 异常，与预期一致。

# 测试编号 7

* 1. **测试代码**



* 1. **测试结果**



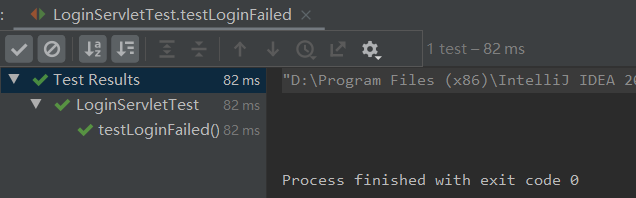
由反馈信息可知，通过测试。用户名为空、密码“123456”，不符合登录条件，抛出RuntimeException 异常，与预期一致。

# 测试编号 8

* 1. **测试代码**



* 1. **测试结果**



由反馈信息可知，通过测试。用户名为空、密码为空，不符合登录条件，抛出RuntimeException 异常，与预期一致。

# 4、总结与分析

通过本次实验，我学会了在 IDEA 中使用EasyMock+Junit 测试代码。测试驱动开发是软件开发的重要部分，单元测试是保证软件质量的重要方法。Mock Object 用于模仿真实对象的方法调用，从而使得测试不需要真正的依赖对象。Mock Object 只为某个特定的测试用例的场景提供刚好满足需要的最少功能。它们还可以模拟错误的条件， 例如抛出指定的异常等；还可以进行次序检查等测试。

在本次实验中，前期遇到了少许问题，如在已导入 servlet-api.jar的情况下仍然出现java.lang.NoClassDefFoundError: javax/servlet/http/ HttpServletRequest 错误。经上网查阅资料后发现未将 Tomcat 加入当前环境中。

在分析测试数据时，考虑到 request.getParameter()的参数有可能为Null，若编程习惯不好，可能会导致错误，故对输入为空的情况也做了测试。考虑到 EasyMock 不仅能够检查是否用正确的参数调用预期的方法，还可以检查是否以正确的次序调用这些方法，本次实验，收获颇丰。