**weiweibo软件测试脚本**

**一、基本信息**

**产品名称：**weiweibo软件测试

**项目承担部门：**东南大学

**撰写人：**09017224 高钰铭；09017225 沈汉唐；09017227 卓旭

**完成日期：**2020-03-30

**二、测试用例**

**2.1、功能测试用例**

根据《weiweibo测试计划》中的测试重点，编写如下测试用例：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **用例编号** | C1.1 | | **用例说明** | 用户注册 – 用户重复 | | |
| **描述** | | **数据** | | | **期望结果** | **实际结果** |
| 注册一个用户 | | username: “test”，其余数据合法 | | | 状态码0 |  |
| 再次注册该用户 | | username: “test”，其余数据合法 | | | 状态码1 |  |
| **用例编号** | C1.2 | | **用例说明** | 用户注册 – 用户名不合规 | | |
| **描述** | | **数据** | | | **期望结果** | **实际结果** |
| 数字开头，特殊符号 | | username: “123%abc”，其余数据合法 | | | 状态码2 |  |
| 超过10位 | | username: “aaaaaaaaaab”，其余数据合法 | | | 状态码2 |  |
| **用例编号** | C1.3 | | **用例说明** | 用户注册 – 密码不合规 | | |
| **描述** | | **数据** | | | **期望结果** | **实际结果** |
| 纯数字密码 | | password: “123456”，其余数据合法 | | | 状态码3 |  |
| 过短密码 | | password: “abc”，其余数据合法 | | | 状态码3 |  |
| 过长密码 | | password: “123456789012345678901”，其余数据合法 | | | 状态码3 |  |
| **用例编号** | C1.4 | | **用例说明** | 用户注册 – 手机号不合规 | | |
| **描述** | | **数据** | | | **期望结果** | **实际结果** |
| 手机号前缀不合规 | | mobile: “66612345678”，其余数据合法 | | | 状态码4 |  |
| 手机号过长 | | mobile: “139123456789”，其余数据合法 | | | 状态码4 |  |
| 手机号过短 | | mobile: “1391234567”，其余数据合法 | | | 状态码4 |  |
| **用例编号** | C1.5 | | **用例说明** | 用户注册 – 电子邮箱不合规 | | |
| **描述** | | **数据** | | | **期望结果** | **实际结果** |
| 不合规的电子邮箱 | | email: “test@baidu”，其余数据合法 | | | 状态码5 |  |
| **用例编号** | C1.6 | | **用例说明** | 用户注册 – 年龄不合规 | | |
| **描述** | | **数据** | | | **期望结果** | **实际结果** |
| 过大的年龄 | | age: 201 | | | 状态码6 |  |
| 过小的年龄 | | age: -1 | | | 状态码6 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **用例编号** | C1.7 | | **用例说明** | 用户注册 – 正常注册 | | |
| **描述** | | **数据** | | | **期望结果** | **实际结果** |
| 合规的用户注册 | | username: “test2”, password: “southeast”,  mobile: “13898765432”, age: 30,  email: “hello@xxx.edu.hk” | | | 状态码0 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **用例编号** | C2.1 | | **用例说明** | 发送新微博 – 用户不存在 | | |
| **描述** | | **数据** | | | **期望结果** | **实际结果** |
| 不存在的用户 | | username: “xxx” ，其余数据合法 | | | 状态码2 |  |
| **用例编号** | C2.2 | | **用例说明** | 发送新微博 – 微博长度不正确 | | |
| **描述** | | **数据** | | | **期望结果** | **实际结果** |
| 过短的微博 | | content: “”，其余数据合法 | | | 状态码1 |  |
| 过长的微博 | | content: “…”（共141字），其余数据合法 | | | 状态码1 |  |
| **用例编号** | C2.3 | | **用例说明** | 发送新微博 – 正常发送 | | |
| **描述** | | **数据** | | | **期望结果** | **实际结果** |
| 合规的发送新微博 | | username: “test2”, content: “…”（共20字） | | | 状态码0 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **用例编号** | C3.1 | | **用例说明** | 获取微博 – 用户不存在 | | |
| **描述** | | **数据** | | | **期望结果** | **实际结果** |
| 不存在的用户 | | username: “xxx” | | | []（空数组） |  |
| **用例编号** | C3.2 | | **用例说明** | 获取微博 – 用户存在 | | |
| **描述** | | **数据** | | | **期望结果** | **实际结果** |
| 无微博的用户 | | username: “test3” （某个无微博的用户） | | | []（空数组） |  |
| 有微博的用户 | | username: “test2”（某个有微博的用户） | | | […]（微博内容数组） |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **用例编号** | C4.1 | | **用例说明** | 获取所有用户微博 | | |
| **描述** | | **数据** | | | **期望结果** | **实际结果** |
| 所有用户 | | —— | | | [{用户名，微博数组}]，且结果与单独获取微博的结果取并集一致 |  |

**2.2、性能测试用例**

根据《weiweibo测试计划》中的相关内容，编写如下测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **用例编号** | P1.1 | **用例说明** | 并发能力测试 | |
| **执行内容** | | **检查依据** | | **实际结果** |
| 模拟100名用户同时进行注册操作 | | 系统是否正常运转；是否均收到正确状态码；响应时间是否低于5秒 | |  |

**三、执行测试脚本**

我们为软件测试开发的mini-jest测试框架主要支持如下五种测试断言：

* toBe断言：校验src是否等于expect
* toBeArray断言：校验src是否与expect具有相同的数组元素
* toHaveLength断言：校验src长度是否为expect
* toContain断言：校验src是否包含expect
* toNotContain断言：校验src是否不包含expect

基于mini-jest框架和上述测试脚本，编写了测试脚本程序，其部分源代码摘录如下：

|  |
| --- |
| // Mini-jest test script.  // manual  const mj = require('F:\\testit\\mini-jest\\mini-jest.js')  const \_bt = require('F:\\testit\\weiweibo\\main.js')  let \_projectName = 'weiweibo'  let \_result = [], \_total = 0, \_pass = 0, \_passCb = () => { \_pass += 1 }  mj.test(  'C1.1-用户注册-用户重复',  [ mj.toBe(\_bt.newUser('test', 'zhuoxu', '13912345678', 'a@b.com', '20'), 0),  mj.toBe(\_bt.newUser('test', 'zhuoxu', '13912345678', 'a@b.com', '20'), 1) ],  \_result, \_passCb  )  \_total += 1  // --- 此处省略其余测试用例代码 ---  let \_toDump = {  "mjInfo": "mini-jest 0.1 manual",  "projectName": \_projectName,  "passRate": \_pass / \_total,  "result": \_result  }  mj.dump(\_toDump, `F:/testit/mini-jest/testscript/manual`, 'weiweibo')  console.log("[mini-jest] Test finished.") |

执行测试后，其生成的结果文件如下：

|  |
| --- |
| {  "mjInfo": "mini-jest 0.1 manual",  "projectName": "weiweibo",  "passRate": 0.9166666666666666,  "result": [  {  "description": "C1.1-用户注册-用户重复",  "body": ["PASS", "PASS"]  },  {  "description": "C1.2-用户注册-用户名不合规",  "body": ["FAIL", "PASS"]  },  // --- 此处省略其余测试用例结果 --- |

该文件表达了所有测试用例的通过或失败情况（PASS、FAIL），并计算了通过率。这些信息可以用于测试总结报告的撰写。