Biopedia

Online Bioinformatics Data Center

Developed By **Nerdy Buddy**

Biopedia - 测试文档

Nerdy Buddy

组长：宋佳铭

组员：袁源，肖迪，杨煜，滕爽

2014 年 12 月 3 日

目录

1 概述

1.1 Biopedia工程概述

在生命科学以及相关领域如医学研究,药物研发,临床诊断,生态环境研究, 农林渔牧等行业每天都有大量的各类数据产生,使生命科学的研究越来越依赖于数据分析。Biopedia工程将建立了一个生物信息展示平台，即一个基于已采集口腔微生物宏基因组及其分析结果的数据库,展示开放的生物信息宏基因组数据的平台。

1.2 文档概述

单元测试主要针对projects.py即Biopedia后台访问数据库的核心模块中部分关键函数进行测试，对这些函数设定特定的单元测试样例。这些关键函数包括Biopedia项目中后端检测前端发送的关于上传、删除、更新等命令的格式的功能函数，涉及了数据与文件格式检查，插入、删除、更新成功检查等等，共34个单元测试，112个语句。
压力测试主要使用LoadRunner对整个网页的各个页面设计虚拟场景，在其中定义性能测试会话期间发生的事件，用虚拟用户代替真实用户，并进行一定分析。

2 测试方案

2.1 测试环境

2.1.1 软件项

名称	用途	配置/版本	厂商	备注
Pycharm	Python IDE	2.7	JetBrains	
Mongodb	数据库	2.6	开源	
xUnit	单元测试框架		开源	
Coverage	覆盖率统计	3.7	Ned Batchelder and others	
SourceTree	版本控制客户端			
LoadRunner	压力测试客户端	11		

表 1: 软件项

2.1.2 硬件项

名称	用途	配置/版本	备注
Macbook Pro	测试	15-inch	
Dell	测试		
Thinkpad	测试	T430	

表 2: 硬件项

3 单元测试

3.1 单元测试过程

3.1.1 测试内容

测试内容为project上传、删除、更新部分。由于后端部分有设计与用户交互的部分，如从request获取表单信息和文件信息，以及返回界面等，故我们除与用户交互的部分单独抽出作为方法进行测试。主要为检查输入数据的格式，以及相关操作是否已经成功等。

3.1.2 测试人员

袁源，杨煜

3.1.3 测试时间

2014/11/28

3.1.4 覆盖率测试方法

本次覆盖率测试使用coverage，测试范围是projects_to_test.py里的所有语句

3.2 测试方法通过率

3.2.1 上传project部分

被测方法	方法说明	用例总数	通过用例数	通过比例
check_basic_field_legal	检查待插入项目特定字段的格式是否正确	16	16	100%
check_file_legal	检查输入的文件名是否合法	6	6	100%
check_field_is_str	判断输入的每个项目的field是否为str型（若是str则输出True，若为int或者float则为False）	3	3	100%
check_sample_legal	检测输出的sample.json是否为json符合json文件格式，且每个sample是否舒服该project	3	3	100%
save_new_project	在确定所有输入数据合法之后写入相应数据	1	1	100%

表 3: 上传Project部分

3.2.2 删除project部分

被测方法	方法说明	用例总数	通过用例数	通过用力比例
delete_project_do	测试是否能够成功删除一个project	1	1	100%

表 4: 删除projects部分

被测方法	方法说明	用例总数	通过用例数	通过用例比例
check_update_project	测试是否能够成功更新project所传的json文件是否合法	3	3	100%

表 5: 上传project部分

被测方法	方法说明	有效行数	通过行数	覆盖率
check_basic_field_legal	检查待插入项目特定字段的格式是否正确	41	41	100%
check_file_legal	检查输入的文件名是否合法	9	9	100%
check_field_is_str	判断输入的每个项目的field是否为str型（若是str则输出True，若为int或者float则为False）	18	18	100%
check_sample_legal	检测输出的sample.json是否为json符合json文件格式，且每个sample是否舒服该project	11	11	100%
save_new_project	在确定输入数据合法后向写入数据	45	45	100%

表 6: 上传project部分覆盖率

3.3 覆盖率分析

3.3.1 上传project部分

3.3.2 删除project部分

被测方法	方法说明	有效行数	通过行数	覆盖率
check_basic_field_legal	检查待插入项目特定字段的格式是否正确	41	41	100%
check_file_legal	检查输入的文件名是否合法	9	9	100%
check_field_is_str	判断输入的每个项目的field是否为str型（若是str则输出True，若为int或者float则为False）	18	18	100%
check_sample_legal	检测输出的sample.json是否为json符合json文件格式，且每个sample是否舒服该project	11	11	100%
save_new_project	在确定输入数据合法后写入数据	45	45	100%

表 7: 删除project部分覆盖率

3.3.3 更新project部分

被测方法	方法说明	有效行数	通过行数	覆盖率
check_update_project	测试更新一个project是否成功	5	5	100%

表 8: 更新project部分覆盖率

Coverage report: 100%

Module	*statements*	*missing*	*excluded*	*coverage*
F:\university\31\biopedia\models	15	0	0	100%
project_test	132	0	0	100%
projects_to_test	112	0	0	100%
Total	**259**	**0**	**0**	**100%**

coverage.py v3.7.1

图 1: 覆盖率测试报告

3.4 评估和建议

3.4.1 软件评估

本项目成功在所有测试用例中，均正确的给出回应（例如每个project的field类型的自动识别处理），对于可能的错误情况（如日期输入错误，文件格式错误）均进行了合理的处理。

由此看出，被测试部分不仅能正确实现了项目的需求，还具有较好的稳定性，符合软件工程的基本理念。

3.4.2 测试工作评估

本次测试工作完成良好。测试覆盖率高，所有被测的方法的覆盖率均达到了100%。设计样例时我们充分考虑到了各种可能的错误情况。如日期判断，我们不但对随机字符串进行判错，还对不存在的时期，如2/29/1999进行的判断。这进一步帮助了网站的鲁棒性。

4 压力测试

4.1 测试人员

滕爽

4.2 测试时间

2014/11/30

4.3 覆盖率测试方法

测试使用了LoadRunner，通过脚本程序判断网站的负载情况。

4.4 测试内容

录制脚本事务	Transaction
主页登陆负载能力测试	Home_page
中英文双语转换负载能力测试	Convert_language
用户登录负载能力测试	Login_demo
用户登出负载能力测试	Logout_demo
用户注册负载能力测试	Register_demo
多项目加载负载能力测试	Project_page
项目信息列表加载负载能力测试	Project_list
图表加载负载能力测试	Sample_ID
个人信息加载负载能力测试	Sample_ID
Workflow反应速率负载能力测试	Workflow_demo

表 9: 压力测试内容

构建脚本时使用参数化，集合化方法，提高模拟情形的准确、客观性。参数化实现构建了登录和注册参数列表，实现互异用户信息的操作。集合化方法模拟用户等待和思考时间，尽量还原真实的场景模式。

4.5 测试场景设计及说明

使用loadrunner controller模块进行场景的设计与构建，具体实现为:

1. 虚拟用户总数量为：200
2. 运行开始后，每相隔15s，同时登入10个用户。
3. 每个用户都进行上述脚本实现的事务的模拟操作，且进行若干次数的迭代操作。该过程相当于某一时刻有200 * 迭代次数个用户进行同一操作。
4. 全部登入系统后，在系统中操作10分钟。
5. 运行结束时，每相隔30s，退出20个用户，直到所有用户退出，运行终止。

4.6 结果

请见同一文件夹下的Report.html和Report.pdf。

4.7 改进建议

- 从事务反应时间图可以看出，最大的占用量就是中英文双语转换功能，一定程度的降低了服务器的性能。
- 从错误提示信息来看，出错较多的是项目信息列表加载，猜测原因是从后台传输数据量过大，造成服务器瞬间崩溃，损伤了一定的性能。

4.8 总体评价

虽然在测试的过程中出现了服务器无法响应，但是出错的事务数仅占整体事务数量的百分之三至四，这些和本身软件的整体架构和基本运行模式有关，测试结果基本属于掌控范围内。

5 附录：单元测试样例表

这次测试的文件为projects/projects.py。

5.1 上传project部分

5.1.1 check_basic_field_legal

test_project_insert_name_empty +

测试目的 检测待插入的项目名是否为空

输入 name字段设为空

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

test_project_insert_environment_empty +

测试目的 检测待插入的项目environment字段是否为空

输入 environment字段设为空

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

test_project_insert_site_empty +

测试目的 检测待插入的项目site字段是否为空

输入 site字段设为空

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

test_project_insert_sequence_type_empty +

测试目的 检测待插入的项目sequence_type是否为空

输入 sequence_type字段设为空

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

test_project_insert_project_id_empty +

测试目的 检测待插入项目id是否为空

输入 id字段设为空

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

test_project_insert_num_of_total_sequences_digit +

测试目的 检测待插入项目的num_of_total_sequences字段是否为数字

输入 num_of_total_sequences字段设为字符串类型值

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

test_project_insert_num_of_orfs_digit +

测试目的 检测待插入项目的num_of_orfs字段是否为数字

输入 num_of_total_sequences字段设为字符串

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

test_project_insert_read_length_digit +

测试目的 检测待插入项目的read_length字段是否为数字

输入 read_length字段设为字符串类型值

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

test_project_insert_platform_empty +

测试目的 检测待插入项目的platform字段是否为空

输入 Platform字段设为空

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

test_project_insert_wrong_create_date_1 +

测试目的 检测待插入项目的create_data是否合法

输入 输入create_data字段为"13-10-1999/1/2"

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

test_project_insert_wrong_create_date_2 +

测试目的 检测待插入项目的create_data是否合法

输入 输入create_data字段为"29/2/1999"

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

test_project_insert_wrong_create_date_3 +

测试目的 检测待插入项目的create_data是否合法

输入 输入create_data字段为"31/11/1999"

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

test_project_insert_wrong_update_date_1 +

测试目的 检测待插入项目的update_date是否合法

输入 输入update_date字段为"13-10-1999/1/2"

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

test_project_insert_wrong_update_date_2 +

测试目的 检测待插入项目的update_date是否合法

输入 输入update_date字段为"29/2/1999"

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

test_project_insert_wrong_update_date_3 +

测试目的 检测待插入项目的update_date是否合法

输入 输入update_date字段为"31/11/1999"

期望输出/相应 抛出ValueError异常

实际输出/相应 抛出ValueError异常

状态 符合预期

5.1.2 check_sample_legal

test_check_sample_legal_1 +

测试目的 检测sample.json内容不合json语法规则时，是否能抛出异常

期望输出/相应 抛出SyntaxError异常

实际输出/相应 抛出SyntaxError异常

状态 符合预期

test_check_sample_legal_2 +

测试目的 检测sample.json中有测例不属于该project时，是否抛出异常

期望输出/相应 抛出SyntaxError异常

实际输出/相应 抛出SyntaxError异常

状态 符合预期

test_check_sample_legal_3 +

测试目的 检测sample.json中所有数据均合法时，是否能正常通过

期望输出/相应 通过

实际输出/相应 通过

状态 符合预期

5.2 删除project部分

5.2.1 save_new_project

test_delete_succeed +

测试目的 检测删除函数是否正常执行

输入 删除之前检测待删除的project_name是否在对应列表中；删除之后检测待删除的project_name是否已经不存在了

期望输出/相应 正常通过

实际输出/相应 正常通过

状态 符合预期

5.3 更新project部分

test_check_update_1 +

测试目的 检测check_update_project函数第一个参数project_name为空时是否能够正常运行

输入 check_update_project函数的参数为project为”“,”jifow.json”

期望输出/相应 False

实际输出/相应 False

状态 符合预期

test_check_update_2 +

测试目的 检测check_update_project函数第二个参数文件名为”jifow.json.”，即不合法时，是否能返回False

输入 check_update_project函数的参数为project为”Test”,”jifow.json.”

期望输出/相应 False

实际输出/相应 False

状态 符合预期

test_check_update_3 +

测试目的 检测check_update_project函数第二个参数文件名为”jifow.json”，即合法，是否能返回False

输入 check_update_project函数的参数为project为”Test”,”jifow.json”

期望输出/相应 False

实际输出/相应 False

状态 符合预期

图片列表

图表列表