测试对象：UserController类

测试人员：刘畅

测试日期：2020/6/23

# 一、测试对象描述

该类作为Controller层的一个类，主要功能为接受前端传来的数据，经过处理后传给UserService进行处理，得到UserService返回的数据后经过处理再返回给前端。

为了方便测试，我们使用mockmvc对前端进行模拟，使用mockBean对UserService进行模拟，将mockBean作为待测对象的桩来进行测试。

该类需要测试的方法如下：

|  |  |  |
| --- | --- | --- |
| 方法标识符 | 方法名 | 测试关注点 |
| Unit\_002\_Function\_001 | String signUp() | 是否能正确响应 |
| Unit\_002\_Fuction\_002 | String login() | 是否能正确响应 |
| Unit\_002\_Fuction\_003 | String login(String userID, String password, HttpServletRequest request) | 是否能正确响应并跳转到正确的界面 |
| Unit\_002\_Function\_004 | void register(String userID, String userName, String password, String email, String phone, HttpServletResponse response) | 是否能正确响应并跳转到正确的界面 |
| Unit\_002\_Fuction\_005 | void logout (HttpServletRequest resquest, HttpServletResponse response) | 是否能正确响应并跳转到正确的界面 |
| Unit\_002\_Fuction\_006 | void quit (HttpServletRequest resquest, HttpServletResponse response) | 是否能正确响应并跳转到正确的界面 |
| Unit\_002\_Fuction\_007 | void updateUser(String userName, String UserId, String passwordNew,String email, String phone, MultipartFile picture,HttpServletRequest request, HttpServletResponse response) | 是否能正确响应并对数据进行正确的处理，最后返回到正确的界面 |
| Unit\_002\_Fuction\_008 | boolean checkPassword(String userID, String password) | 是否能正确判断数据并返回正确的信息。 |
| Unit\_002\_Function\_009 | String user\_info(Model model) | 是否能正确响应 |

# 二、测试用例设计

## 2.1、Unit\_002\_Fuction\_001

待测试的方法为UserController类中的String signUp()方法。

该方法接收到前端传来的get请求后，返回对应的signup界面。为了确认返回界面是否成功，我们只用设计如下单个测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 驱动函数 | 输入变量 | 预期结果 | 执行结果 |
| 1 | *return\_sign\_up\_html()* | *get*("/signup") | andExpect(status().isOk())通过 |  |

## 2.2、Unit\_002\_Fuction\_002

待测试的方法为UserController类中的String login()方法

该方法接受到前端传来的get请求后，返回对应的login界面。为了确认返回界面是否成功，我们只用设计如下单个测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 驱动函数 | 输入变量 | 预期结果 | 执行结果 |
| 1 | *return\_login\_html()* | *get*("/login") | andExpect(status().isOk())通过 |  |

## 2.3、Unit\_002\_Function\_003

待测试的方法为UserController类中的String login(String userID, String password, HttpServletRequest request)方法

该方法接收到前端传来的userID，password参数后，调用userService类中的checkLogin方法，当service返回不为空时，对用户类型进行判断后返回对应的登录成功界面；当service返回空时，该方法返回false。

为了实现分支覆盖，我们需要对每个判断语句取其真假情况，于是我们设计如下4个测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 驱动函数 | 输入变量 | 预期结果 | 执行结果 |
| 1 | *user\_does\_not\_exist\_or\_wrong\_password()* | *post*("/loginCheck.do").param("userID","test").param("password","test") | andExpect(status().isOk())通过  andExpect(*content*().string("/index")通过 |  |
| 2 | *user\_login\_succeeded()* | *post*("/loginCheck.do").param("userID","admin").param("password","admin") | andExpect(status().isOk())通过  andExpect(*content*().string("/admin\_index")通过 |  |
| 3 | *admin\_login\_succeeded()* | *post*("/loginCheck.do").param("userId","noUser").param("password","noPassword") | andExpect(*status*().isOk())  通过  andExpect(*content*().string("false")通过 |  |
| 4 | *invalid\_Isadmin()* | *post("/loginCheck.do").param("userID","admin").param("password","admin")*  *user.setIsadmin(5);* | andExpect(*status*().isOk())  通过  andExpect(*content*().string("false")通过 |  |

用例1覆盖if语句一真，if语句二真

用例2覆盖if语句一真，if语句二假，elseif语句真

用例3覆盖if语句一假

用例4覆盖if语句一真，if语句二假，elseif语句假

## 2.4、Unit\_002\_Fuction\_004

待测试的方法为UserController类中的void register(String userID, String userName, String password, String email, String phone, HttpServletResponse response)方法

该方法接受前端传来的userID，userName，password，email，phone信息，调用Service层UserService类的create函数后返回到前端login界面。

为了确认是否调用正确的service函数和返回正确的界面，我们设计了如下单个用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 驱动函数 | 输入变量 | 预期结果 | 执行结果 |
| 1 | register\_a\_new\_user() | (*post*("/register.do").param("userID","user").param("userName","name").param("password","password")  .param("email","email").param("phone","phone")); perform.andExpect(*redirectedUrl*("/login") | andExpect(*redirectedUrl*("/login")通过  *verify*(userService).create(*any*());通过 |  |

## 2.5、Unit\_002\_Fuction\_005

待测试的方法为UserController类中的logout(HttpServletRequest request, HttpServletResponse response)方法

该方法收到用户前端传来的get请求后，注销session，打印“log out success！”后返回到index界面。

为了确认是否返回到正确的界面，我们设计了如下单个测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 驱动函数 | 输入变量 | 预期结果 | 执行结果 |
| 1 | *user\_logout()* | *get*("/logout.do") | andExpect(*redirectedUrl*("/index")通过 |  |

## 2.6、Unit\_002\_Fuction\_006

待测试的方法为UserController类中的void quit (HttpServletRequest resquest, HttpServletResponse response)方法

该方法收到管理员前端传来的get请求后，注销session，打印“log out success！”后返回到index界面。

为了确认是否返回到正确的界面，我们设计了如下单个测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 驱动函数 | 输入变量 | 预期结果 | 执行结果 |
| 1 | *admin\_quit()* | *get*("/logout.do") | andExpect(*redirectedUrl*("/index")通过 |  |

## 2.7、Unit\_002\_Fuction\_007

待测试的方法为UserController类中的void updateUser(String userName, String UserId, String passwordNew,String email, String phone, MultipartFile picture,HttpServletRequest request, HttpServletResponse response)方法

该方法收到前端传来的userName、userID、passwordNew、email、phone、picture参数后，对passwordNew和picture进行判断，调用userService中的updateUesr函数后对session中的user信息进行更新，最后返回到user\_info界面。

为了实现条件组合覆盖和测试该方法是否调用正确的函数和返回正确的界面，我们需要对每中条件组合的情况设计测试用例，于是我们设计如下6个测试用例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 驱动函数 | 输入变量 | 预期结果 | 执行结果 |
| 1 | *user\_update\_passwordNew\_is\_null\_and\_picture\_is\_null()* | *multipart*("/updateUser.do").file(mockMultipartFile).param("userID","user").param("userName","name")  .param("passwordNew", (String) null)  .param("email","email").param("phone","phone"); | andExpect(*redirectedUrl*("user\_info"))  *verify*(userService).findByUserID(*anyString*()); *verify*(userService).updateUser(*any*());  通过 |  |
| 2 | *user\_update\_passwordNew\_is\_\_and\_picture\_is\_null()* | *multipart("/updateUser.do").file(mockMultipartFile).param("userID","user").param("userName","name")  .param("passwordNew", "")  .param("email","email").param("phone","phone");* | andExpect(*redirectedUrl*("user\_info"))  *verify*(userService).findByUserID(*anyString*()); *verify*(userService).updateUser(*any*());  通过 |  |
| 3 | *user\_update\_password\_exists\_and\_picture\_is\_null()* | *multipart("/updateUser.do").file(mockMultipartFile).param("userID","user").param("userName","name")  .param("passwordNew", "passwordNew")  .param("email","email").param("phone","phone");* | andExpect(*redirectedUrl*("user\_info"))  *verify*(userService).findByUserID(*anyString*()); *verify*(userService).updateUser(*any*());  通过 |  |
| 4 | *user\_update\_passwordNew\_is\_null\_and\_picture\_exists()* | *multipart("/updateUser.do").file(mockMultipartFile).param("userID","user").param("userName","name")  .param("passwordNew", (String) null)  .param("email","email").param("phone","phone");* | andExpect(*redirectedUrl*("user\_info"))  *verify*(userService).findByUserID(*anyString*()); *verify*(userService).updateUser(*any*());  通过 |  |
| 5 | *user\_update\_passwordNew\_is\_\_and\_picture\_exists()* | *multipart("/updateUser.do").file(mockMultipartFile).param("userID","user").param("userName","name")  .param("passwordNew", "")  .param("email","email").param("phone","phone");* | andExpect(*redirectedUrl*("user\_info"))  *verify*(userService).findByUserID(*anyString*()); *verify*(userService).updateUser(*any*());  通过 |  |
| 6 | *user\_update\_passwordNew\_and\_picture\_exist()* | *multipart("/updateUser.do").file(mockMultipartFile).param("userID","user").param("userName","name")  .param("passwordNew", "passwordNew")  .param("email","email").param("phone","phone");* | andExpect(*redirectedUrl*("user\_info"))  *verify*(userService).findByUserID(*anyString*()); *verify*(userService).updateUser(*any*());  通过 |  |

用例1覆盖条件一假、条件二假和条件三假

用例2覆盖条件一真、条件二假和条件三假

用例3覆盖条件一真、条件二真和条件三假

用例4覆盖条件一假、条件二假和条件三真

用例5覆盖条件一真、条件二假和条件三真

用例6覆盖条件一真、条件二真和条件三真

由于条件一与条件而存在相干性，不存在条件一假、条件二真的情况。

## 2.8、Unit\_002\_Fuction\_008

待测试的方法为UserController类中的boolean checkPassword(String userID,String password)方法

该方法收到前端传来的userID、password，通过调用userService函数获取正确的password进行比对，返回比对的结果。

为了测试是否能返回正确的比对结果，我们设计了如下两个测试用例。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 驱动函数 | 输入变量 | 预期结果 | 执行结果 |
| 1 | *true\_password()* | *get*("/checkPassword.do").param("userID","User").param("password","password") | andExpect(*status*().isOk())通过  andExpect(*content*().string("true"))通过 |  |
| 2 | *wrong\_password()* | *get("/checkPassword.do").param("userID","User").param("password","pass")* | andExpect(*status*().isOk())通过  andExpect(*content*().string("false"))通过 |  |

其中，userService返回的值为：

User user=new User();  
user.setUserID("User");  
user.setPassword("password");

## 2.8、Unit\_002\_Fuction\_009

待测试的方法为UserController类中的String user\_info(Model model)

方法

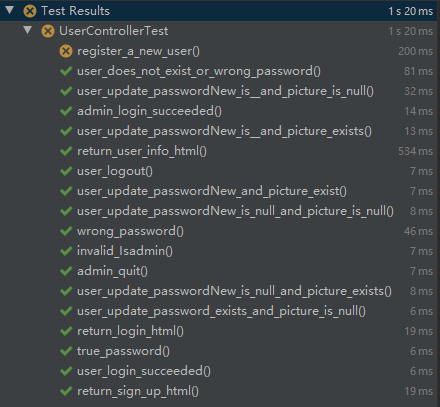
该方法在获取前端的user信息后，跳转到user\_info界面。

为了测试是否能正确响应，我们设计了如下测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 驱动函数 | 输入变量 | 预期结果 | 执行结果 |
| 1 | *return\_user\_info\_html()* | *get*("/user\_info").sessionAttr("user",new User()) | andExpect(*status*().isOk())通过 |  |

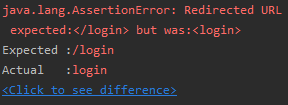
# 三、测试用例的执行与结果

测试用例执行如下图所示：



1 测试用例不通过、17 测试用例通过

不通过测试用例exception如下：



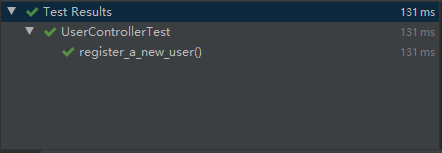
# 四、测试用例failed原因分析

通过junit对比工具我们可知：函数返回的界面与我们所期望的界面有出入，函数返回的界面中缺少/符号，导致failed的出现。

该bug已提交到trucup bug追踪工具中

Bug修复：对register函数中的sendRedirect内容修改为/login

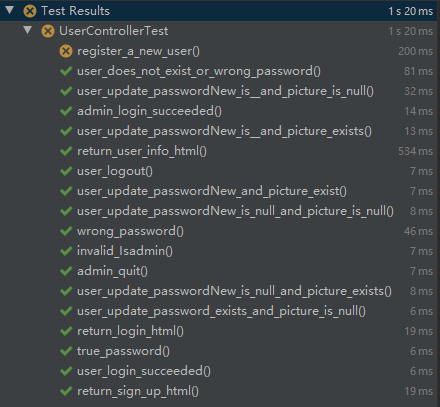
再次执行该测试用例：



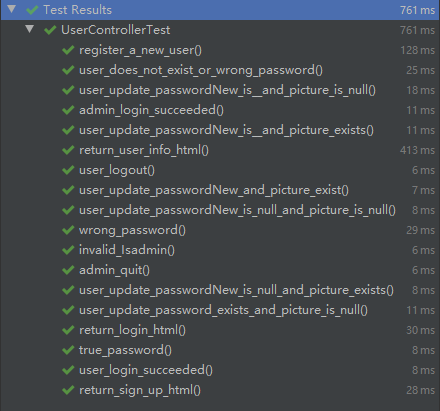
测试通过。

# 五、测试结果分析

修复前执行的测试用例结果如下：

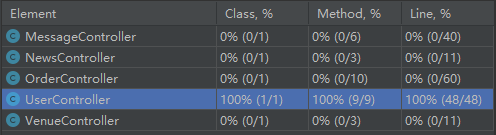


修复后执行测试用例的结果如下：



所有测试用例均执行成功。

测试用例对于UserController类的覆盖度如下：



由上述信息可得知，测试对于UserController类的方法覆盖度（Method）达到了100%，对语句覆盖度（Line）也达到了100%，并且修复后所有测试用例均执行成功，经测试经理评审，该单元通过了单元测试。