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() |  |
| Unit\_002\_Fuction\_002 | String login() |  |
| Unit\_002\_Fuction\_003 | String login(String userID, String password, HttpServletRequest request) |  |
| Unit\_002\_Function\_004 | void register(String userID, String userName, String password, String email, String phone, HttpServletResponse response) |  |
| Unit\_002\_Fuction\_005 | void logout (HttpServletRequest resquest, HttpServletResponse response) |  |
| Unit\_002\_Fuction\_006 | void quit (HttpServletRequest resquest, HttpServletResponse response) |  |
| Unit\_002\_Fuction\_007 | void updateUser(String userName, String UserId, String passwordNew,String email, String phone, MultipartFile picture,HttpServletRequest request, HttpServletResponse response) |  |
| Unit\_002\_Fuction\_008 | boolean checkPassword(String userID, String password) |  |
| Unit\_002\_Function\_009 | String user\_info(Model model) |  |

# 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 | 输入变量 | 预期输出 | 执行结果 |
| 1 | *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 | 输入变量 | 预期输出 | 执行结果 |
| 1 | *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 | 输入变量 | 预期输出 | 执行结果 |
| 1 | *post*("/loginCheck.do").param("userID","test").param("password","test") | andExpect(status().isOk())通过  andExpect(*content*().string("/index")通过 |  |
| 2 | *post*("/loginCheck.do").param("userID","admin").param("password","admin") | andExpect(status().isOk())通过  andExpect(*content*().string("/admin\_index")通过 |  |
| 3 | *post*("/loginCheck.do").param("userId","noUser").param("password","noPassword") | andExpect(*status*().isOk())  通过  andExpect(*content*().string("false")通过 |  |
| 4 | *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 | 输入变量 | 预期输出 | 执行结果 |
| 1 | (*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 | 输入变量 | 预期输出 | 执行结果 |
| 1 | *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 | 输入变量 | 预期输出 | 执行结果 |
| 1 | *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 | 输入变量 | 预期输出 | 执行结果 |
| 1 | *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 |  |