软件测试文档

目录

一、概述

前端部分代码主要负责完成用户界面和交互，系统化测试较为困难；后台部分代码是业务逻辑的核心，已经形成了一些成熟的测试方案。本文档仅针对后台部分设计测试用例和测试代码。

二、测试用例

项目比较简单，仅仅测试项目的功能。

针对登陆系统、修改个人信息、预定课室、修改预订、取消预订、审批预订、维护课室信息、维护学生信息、维护审批人员信息、维护管理员信息等用例设计了测试用例，采用黑盒测试中等价类的方法设计测试用例。

1. 登陆系统测试用例

用户输入学(工)号和密码，登陆系统。学(工)号为 8 为数字，密码 6~16 为非空格的字符，且必须以字母或数字开头。

等价类划分如下：

标号	**有效等价类**	**标号**	**无效等价类**
U1	{学工号为 8 位，不包含非法字符}	U2	学工号不等于 8 位
		U3	学工号包含非法字符
P1	{密码位数：5<=位数<=16，不包含非法字符}	P2	密码位数：位数<6
		P3	密码位数：位数>16
		P4	密码包含非法字符

弱健壮等价类测试用例如下。输入学工号和密码，返回 http 状态码：

输入	**输出**	**覆盖等价类**
15331124, 123456	200	P1 UI
153311， 123456	500	U2
15**1124, 123456	500	U3
15331124, 1234	500	P2
15331124, 123456789000ersdq	500	P3
15331124, &12312233	500	P4

由于测试用例较多，针对有效等价类的测试代码如下：

```go
func TestStudentSignin(t *testing.T) {
  reqData := struct {
    StudentId  int
    StudentPwd string
  }{15331124, "123456"}

  reqBody, _ := json.Marshal(reqData)
  req := httptest.NewRequest(
    http.MethodPost,
    "/student/sigin",
    bytes.NewReader(reqBody),
  )

  w := httptest.NewRecorder()
  controllers.StudentSignin(w, req)

  resp := w.Result()
  if resp.StatusCode != 200 {
    t.Fail()
    errStdout(t, resp)
  }
  if resp.Header.Get("Set-Cookie") == "" {
    t.Fail()
  }
}
```

2. 修改个人信息

用户输入新密码，系统修改用户的登陆密码。密码 6~16 为非空格的字符，且必须以字母或数字开头。

等价类划分如下：

标号	有效等价类	标号	无效等价类
P1	{密码位数：5<=位数<=16，不包含非法字符}	P2	密码位数：位数<6
		P3	密码位数：位数>16
		P4	密码包含非法字符

弱健壮等价类测试用例如下。输入密码，返回 http 状态码：

输入	输出	覆盖等价类
example123	200	P1
1234	500	P2
123456789000ersdq	500	P3
&12312233	500	P4

由于测试用例较多，针对有效等价类的测试代码如下：

```go
func TestUpdateStudentById(t *testing.T) {
  reqData := struct {
    StudentPwd string
  }{"example123"}
  reqBody, _ := json.Marshal(reqData)
  req := httptest.NewRequest(
    http.MethodPut,
    "/api/users/students/id",
    bytes.NewReader(reqBody),
  )
```

```
  req = mux.SetURLVars(req, map[string]string{"id": "15331124"})
  req.Header.Set("userType", "Admin")
  req.Header.Set("userId", "11331124")

  w := httptest.NewRecorder()
  controllers.UpdateStudentById(w, req)

  resp := w.Result()
  if resp.StatusCode != 200 {
    t.Fail()
    errStdout(t, resp)
  }
}
```

3. 预订课室

用户输入开始时间、结束时间、预定原因、课室 ID、组织社团，预订申请课室。开始时间、结束时间、预定原因、课室 ID、组织社团等信息均不能为空；开始时间、结束时间格式为"xxxx-xx-xx 第 x 节课", string 类型; 课室 ID 为数字; 预订原因和组织社团为 string 类型。

等价类划分如下:

标号	有效等价类	标号	无效等价类
T1	开始及结束时间格式为：xxxx-xx-xx 第 x 节课	T2	开始或(和)结束时间为空
		T3	开始或(和)结束时间不符合格式
R1	预定原因不为空	R2	预订原因为空
ID1	课室 ID 为数字	ID2	课室 ID 包含非法字符
		ID3	课室 ID 为空
O1	组织社团不为空	O2	组织社团为空

弱健壮等价类测试用例如下:

输入	输出	覆盖等价类
“2018-06-13 第三节课”, ”2018-06-13 第五节课”, ”习题辅导课”, 3, ”教务二班”	200	T1 R1 ID1 O2
“”, “”, ”习题辅导课”, 3, ”教务二班”	500	T2
“2018-06-13 第三节”, “2018-6-13 第五节课”, ”习题辅导课”, 3, ”教务二班”	500	T3
“2018-06-13 第三节课”, ”2018-06-13 第五节课”, ””, 3, ”教务二班”	500	R1
“2018-06-13 第三节课”, ”2018-06-13 第五节课”, ”习题辅导课”, 3a, ”教务二班”	500	ID2
“2018-06-13 第三节课”, ”2018-06-13 第五节课”, ”习题辅导课”, , ”教务二班”	500	ID3

“2018-06-13 第三节课”，”2018-06-13 第五节课”，”习题辅导课”，3，””	500	O2

由于测试用例较多，针对有效等价类的测试代码如下：

```
func TestAddRes(t *testing.T) {
  reqData := struct {
    StartTime   string
    EndTime     string
    ResReason   string
    ClassroomId int
  }{"2018-06-13 第三节课", "2018-06-13 第五节课", "习题辅导课", 2}
  reqBody, _ := json.Marshal(reqData)
  req := httptest.NewRequest(
    http.MethodPost,
    "/api/users/student/reservations",
    bytes.NewReader(reqBody),
  )
  req.Header.Set("userType", "Student")
  req.Header.Set("userId", "15331124")

  w := httptest.NewRecorder()
  controllers.AddRes(w, req)

  resp := w.Result()
  if resp.StatusCode != 200 {
    t.Fail()
    errStdout(t, resp)
  }
```

```
  //Test wether the data has in databases
  time.Sleep(10)
  req = httptest.NewRequest(
    http.MethodGet,
    "/api/users/student/reservations",
    nil,
  )
  req.Header.Set("userType", "Student")
  req.Header.Set("userId", "15331124")

  w = httptest.NewRecorder()
  controllers.GetStudentResList(w, req)

  resp = w.Result()
  if resp.StatusCode != 200 {
    t.Fail()
    errStdout(t, resp)
  }
}
```

4. 修改预订

用户输入预订 ID 和预订原因、申请组织，更新预订信息。预订 ID 和预订原因、申请组织均不为空，ID 为数字，预订原因、申请组织为 string 类型。

等价类划分如下：

标号	有效等价类	标号	无效等价类
ID1	ID 为数字	ID2	ID 包含非法字符

		ID3	ID 为空
R1	预定原因不为空	R2	预订原因为空
O1	组织社团不为空	O2	组织社团为空

弱健壮等价类测试用例如下：

输入	输出	覆盖等价类
1, "考试复习", "东校园学生会"	200	ID1, R1, O1
1A, "考试复习", "东校园学生会"	500	ID2
, "考试复习", "东校园学生会"	500	ID3
1, "", "东校园学生会"	500	R2
1, "考试复习", ""	500	O2

由于测试用例较多，针对有效等价类的测试代码如下：

```
func TestUpdateResById(t *testing.T) {
  reqData := struct {
    ResReason        string
    OrganizationName string
  }{"考试复习", "东校园学生会"}
  reqBody, _ := json.Marshal(reqData)
  req := httptest.NewRequest(
    http.MethodGet,
    "/api/reservations/id",
    bytes.NewReader(reqBody),
  )
  req.Header.Set("userType", "Student")
  req.Header.Set("userId", "15331124")
  req = mux.SetURLVars(req, map[string]string{"id": "1"})

  w := httptest.NewRecorder()
  controllers.UpdateResById(w, req)

  resp := w.Result()
  if resp.StatusCode != 200 {
    t.Fail()
    errStdout(t, resp)
  }
}
```

5. 取消预订

用户输入预订 ID，删除预订。ID 为数字，不为空。

等价类划分如下：

标号	有效等价类	标号	无效等价类
ID1	ID 为数字	ID2	ID 包含非法字符
		ID3	ID 为空

弱健壮等价类测试用例如下：

输入	输出	覆盖等价类
1	200	ID1
1	500	ID2
	500	ID3

由于测试用例较多，针对有效等价类的测试代码如下：

```go
func TestDeleteResById(t *testing.T) {
  req := httptest.NewRequest(
    http.MethodGet,
    "/api/reservations/id",
    nil,
  )
  req.Header.Set("userType", "Student")
  req.Header.Set("userId", "15331124")
  req = mux.SetURLVars(req, map[string]string{"id": "1"})

  w := httptest.NewRecorder()
  controllers.DeleteResById(w, req)

  resp := w.Result()
  if resp.StatusCode != 200 {
    t.Fail()
    errStdout(t, resp)
  }
}
```

6. 审批预订

用户输入预订 ID、预订审批状态、预订审批备注，审批预订申请。预订 ID、预订审批状态、预订审批备注均不为空；预订 ID 为数字；预订审批状态为 2(通过)或 3(不通过)；预订备注为 string 类型。

等价类划分如下：

标号	有效等价类	标号	无效等价类
ID1	ID 为数字	ID2	ID 包含非法字符
		ID3	ID 为空
S1	预订审批状态为 2 或 3	S2	预订审批状态包含非法字符
		S3	预订审批状态为空
N1	预订审批备注不为空	N2	预订审批备注为空

弱健壮等价类测试用例如下：

输入	输出	覆盖等价类
1, 2, “活动内容符合规范，予以通过”	200	ID1, S1, N1
1a, 2, “活动内容符合规范，予以通过”	500	ID2
, 2, “活动内容符合规范，予以通过”	500	ID3
1, 2a, “活动内容符合规范，予以通过”	500	S2
1, , “活动内容符合规范，予以通过”	500	S3
1, 2, “”	500	N2

由于测试用例较多，针对有效等价类的测试代码如下：

标号	有效等价类	标号	无效等价类
ID1	ID 为数字	ID2	ID 包含非法字符
		ID3	ID 为空
S1	预订审批状态为 2 或 3	S2	预订审批状态包含非法字符
		S3	预订审批状态为空
N1	预订审批备注不为空	N2	预订审批备注为空

```go
func TestUpdateResById(t *testing.T) {
  reqData := struct {
    ResReason        string
    OrganizationName string
  }{"考试复习", "东校园学生会"}
  reqBody, _ := json.Marshal(reqData)
  req := httptest.NewRequest(
    http.MethodGet,
    "/api/reservations/id",
    bytes.NewReader(reqBody),
  )
  req.Header.Set("userType", "Student")
  req.Header.Set("userId", "15331124")
  req = mux.SetURLVars(req, map[string]string{"id": "1"})

  w := httptest.NewRecorder()
  controllers.UpdateResById(w, req)

  resp := w.Result()
  if resp.StatusCode != 200 {
    t.Fail()
    errStdout(t, resp)
  }
}
```

7. 维护课室信息

维护课室信息包括增加课室信息、更新课室信息、删除课室信息等功能。这里仅涉及增加课室信息的测试用例设计。用户输入校区、楼区、课室号、课室容量等信息，系统添加课室信息。校区、楼区、课室号为 string 类型；课室容量为数字；以上均不为空。

等价类划分如下：

标号	有效等价类	标号	无效等价类
C1	校区不为空	C2	校区为空
B1	楼区不为空	B2	楼区为空
CL1	课室号不为空	CL2	课室号为空
Ca1	课室容量为数字	Ca2	课室容量包含非法字符
		Ca3	课室容量为空

弱健壮等价类测试用例如下：

输入	输出	覆盖等价类
"北校园", "医学院院楼", "D104", 100	200	C1 B1 CL1 Ca1
"", "医学院院楼", "D104", 100	500	C2
"北校园", "", "D104", 100	500	B2
"北校园", "医学院院楼", "", 100	500	CL2
"北校园", "医学院院楼", "D104", 100A	500	Ca2
"北校园", "医学院院楼", "D104",	500	Ca3

由于测试用例较多，针对有效等价类的测试代码如下：

```go
func TestAddClassroom(t *testing.T) {
  reqData := services.ClassroomService.NewClassroom("北校园", "医学院院楼", "D104", 100)
  reqBody, _ := json.Marshal(*reqData)
  req := httptest.NewRequest(
    http.MethodPost,
    "/api/classrooms",
    bytes.NewReader(reqBody),
  )
  req.Header.Set("userType", "Admin")
  req.Header.Set("userId", "11331124")

  w := httptest.NewRecorder()
  controllers.AddClassroom(w, req)

  resp := w.Result()
  if resp.StatusCode != 200 {
    t.Fail()
    errStdout(t, resp)
  }
}
```

三、测试环境部署

为了方便测试，项目中附上了测试环境的部署方案，在项目的 testconfig/README.md 文件中。具体方案如下：

8. 安装 docker 和 docker-compose

docker 安装：http://www.runoob.com/docker/ubuntu-docker-install.html
docker-compose 安装：https://github.com/docker/compose/releases

9. 配置 mysql 数据库

1.1 构建镜像

在项目根目录执行以下命令：

```
$ cd testconfig/mysql/dbmaster
$ docker run build -t registry.cn-
shenzhen.aliyuncs.com/selfmysql/master:latest .
$ cd ../dbslave
$ docker run build -t registry.cn-
shenzhen.aliyuncs.com/selfmysql/slave .
```

运行命令后，你将的到如下两个镜像

```
$ docker images
registry.cn-shenzhen.aliyuncs.com/selfmysql/slave     latest
ad0a4971c1a5        4 weeks ago         371 MB
```

```
registry.cn-shenzhen.aliyuncs.com/selfmysql/master    latest
cfa626cc975d        4 weeks ago         371 MB
```

1.2 修改 docker-compose 文件配置

对 testconfig/mysql`目录下的 docker-compose.yml 文件进行修改。 主要修改 volumes 配置项，根据自己的情况挂载数据卷。例如，文件默认的配置为：

```
volumes:
  - $HOME/Work/data/dbmaster:/var/lib/mysql
```

```
volumes:
  - $HOME/Work/data/dbslave:/var/lib/mysql
```

含义为：

```
将本地目录`$HOME/Work/data/dbslave`挂载到镜像的`/var/lib/mysql`目录下。
将本地目录`$HOME/Work/data/dbmaster`挂载到镜像的`/var/lib/mysql`目录下。
```

1.3 启动

在 testconfig/mysql 目录下运行如下命令：

```
$ docker-compose up -d
```

在 testconfig/mysql 目录下使用如下命令查看是否启动成功：

```
$ docker-compose logs
```

1.3 配置主从

这里尝试了数据库配置主从，实现读写分离，从而减轻数据操作的压力。可以参考以下网址：
https://blog.csdn.net/kiloveyousmile/article/details/79833043

2. 修改部分文件

由于项目是面向后台部署编写，后台部署和测试部署略有不同，为何方便，这里需要对后台项目数据库代码部分进行简单修改：
models/entities/init.go

```
// Just for test
MasterEngine, err = xorm.NewEngine("mysql",
"root:master@tcp(localhost:3307)/ccrsystem?charset=utf8&parseTime=true"
)
// True connection for mysql
// MasterEngine, err = xorm.NewEngine("mysql",
"root:master@tcp(dbmaster:3306)/ccrsystem?charset=utf8&parseTime=true")

// Just for test
```

```
SlaveEngine, err = xorm.NewEngine("mysql",
"root:slave@tcp(localhost:3308)/ccrsystem?charset=utf8&parseTime=true")
// True connection for mysql
// SlaveEngine, err = xorm.NewEngine("mysql",
"root:slave@tcp(dbslave:3306)/ccrsystem?charset=utf8&parseTime=true")
```

3. 启动测试

进入根目录，执行如下命令即可：

```
$ go test .
```

四、测试初始数据

为了使得项目能够正常运行，项目增加了部分初始数据，这些数据在项目初次启动时，会自动添加到数据库当中。实现的文件为 controller/init.go。具体数据如下：

Admin 信息：

id	admin_pwd	admin_name
11331124	858e5a25636bd82ac3c2913039c270a1	赵毅
11331125	afe4e651ba5d760727b7c161da011b1c	李阳
11331126	394b123a38eb57982fc961ba5dce8f6b	万余里
11331127	f9e5f4c7ab57ea2024e482bf481e51de	陈向阳

Student 信息：

id	student_pwd	student_name
15331124	5243627672763cd3d4626ea0ebb7e06d	王芳
15331125	67b6bac54b407c7f69e4d65195515c48	赵红
16331124	878db465451b5f2d1826cf1a3e6fdecb	王芸
16331125	7ae1bc9ac89702e7ccb1ed20ef787557	张晓

4 rows in set (0.00 sec)

Approver 信息：

id	approver_pwd	approver_name	department_id
10331124	09a1eaf4b15f4a3dd2efbbfa617ecb3f	李怡	1
10331125	748ac75f7b9f7375f22521ddce3fb766	宋晓	2
10331126	aeb53f8b315795edf239e3fcc115f154	李芸	3
10331127	32f6e65730c3515db9176ff7f15d9081	张丽	1

4 rows in set (0.00 sec)

Classroom 信息：

id	classroom_campus	classroom_building	classroom_num	capacity
1	东校园	公教楼	A201	50
2	东校园	公教楼	A202	50
3	东校园	公教楼	C102	100
4	东校园	公教楼	C101	200
5	南校园	第一教学楼	A104	100

Department 信息：

id	department_name	introduction	department_order	note
1	团委	位于东校明德园6号，负责学生活动审批	1	initial
2	学院团委	该部门主要负责本院活动的审批	2	middle
3	保卫办	该部门主要负责对于活动和场所的审查	3	final

3 rows in set (0.00 sec)

Reservation 信息：

id	res_reason	start_time	end_time	classroom_id	student_id	organization_name	approver_id	res_state	approval_note
1	社团面试	2018-05-23 09:50:00	2018-05-23 11:40:00	1	15331124	东校园学生会	0	3	
2	生日party	2018-05-24 14:25:00	2018-05-24 16:15:00	2	15331124	软工4班	10331124	3	不允许教学区进行该活动
3	班级会议	2018-06-01 19:00:00	2018-06-01 19:55:00	3	15331124	软工4班	10331126	2	
7	班级会议	2018-06-22 08:00:00	2018-06-22 20:50:00	4	15331124	数据院1班	10331127	0	