1. 数据库抽象实现

//数据库增删改查标准接口实现

public interface Manager<T> {

Key<T> insert(T t);

Iterable<Key<T>> insertList(List<T> tList);

void delete(String id);

List<?> findByColumn(String key, String value);

long findExist(String t);

T findById(String t);

void update();

List<?> read();

}

//通用化实体实现初始化

public class ManagerBuilder {

private static String pakName = "com.wyc.pims.model";

public static Manager build(String beanName) throws ClassNotFoundException {

String fullName = pakName + "." + beanName;

Class claz = Class.forName(fullName);

return new ManagerImpl(claz,beanName);

}

}

//数据库链接实例初始化

public class DAOImpl implements DAO {

private static MongoClient mongoClient = null;

public DAOImpl() {

if (mongoClient == null){

//获取服务器地址

ServerAddress serverAddress=new ServerAddress(DALConfigBean.getInstance().getHost(),DALConfigBean.getInstance().getPort());

//获取服务器用户名和密码

if (DALConfigBean.getInstance().getUsername() != null && DALConfigBean.getInstance().getPassword() != null) {

MongoCredential credential = MongoCredential.createCredential(DALConfigBean.getInstance().getUsername(), DALConfigBean.getInstance().getDatabaseName(), DALConfigBean.getInstance().getPassword().toCharArray());

//获取数据库实例

mongoClient=new MongoClient(serverAddress, Arrays.asList(credential));

}else {

mongoClient=new MongoClient(serverAddress);

}

}

}

public void resetMongoClient(MongoClient mc){

//刷新mongo数据库实例

mongoClient=mc;

}

public MongoClient getMongoClient() {

//动态获取mongo数据库实例

return mongoClient;

}

public Datastore createDatastore(Class clazz) {

//使用mophia包装实例

Morphia morphia = new Morphia();

morphia.mapPackage("com.wyc.pims.model");

return morphia.createDatastore(mongoClient, "pims");

}

}

2.登录模块重点代码

前台：

PRM.factory('QueryToolService', ['$resource', 'PRMconf', function ($resource, config) {  
 return $resource('', {}, {

//前台注册用户入口

'register':{

//标准本地URL获取  
 url:config.MONGOQUERY + 'register',

//前后台ajax请求类型设置  
 method: 'POST',

//ajax请求数据接收类型定义  
 headers: {'Content-Type': "application/json;charset=utf-8"},

//是否直接将返回结果序列化成列队

isArray: false,

//是否支持缓存  
 cache: false  
 },

//登录前台ajax封装  
 'login':{  
 url:config.MONGOQUERY + 'login',  
 method: 'POST',  
 headers: {'Content-Type': "application/json;charset=utf-8"},  
 isArray: false,  
 cache: false  
 }  
 });  
}]);

$scope.login = function(){

//angular插件，遮盖层  
 $loading.start("loadingMask");

//angular前台ajax封装类，调用登录方法  
 QueryToolService.login($rootScope.login,function(res){  
//如果后台返回登录成功  
 if(res.successful){

//前台设置登陆后权限和用户信息，并且提示是否登录成功  
 $rootScope.login = res.data[0];  
 $rootScope.role = res.data[0].role;  
 $rootScope.logged = true;  
 noty.show(res.message, 'success');  
 }else{

//如果后台返回错误，类似500，后台会封装成错误  
 noty.show(res.message, 'error');  
 }  
 $loading.finish("loadingMask");  
 },function(error){

//如果http call出现错误，类似404之类错误  
 noty.show("登录失败", 'error');  
 $loading.finish("loadingMask");  
 });  
  
 };

后台：

//Spring Controller 注解进行对login 请求的重定向，只接受POST请求，和json数据类型

@RequestMapping(value = "/login", method = RequestMethod.POST, produces = "application/json")  
 public

//返回数据由spring进行序列化封装并返回到前台http response中  
 @ResponseBody  
 ResponsePackage login(@RequestBody User user) {  
  
 try {  
 if (user.getId().equals("10000000") && user.getPassword().equals("1234")) {  
 List<User> adminUserList = new ArrayList<User>();  
 User admin = new User();  
 admin.setId("10000000");  
 admin.setPassword("");  
 admin.setRole("admin");  
 adminUserList.add(admin);  
 return UnifiedFunctions.buildQuickSuccess(adminUserList, "登录成功");  
 }

//使用mongo实例获取User table并且调用查询方法  
 User findUser = (User) ManagerBuilder.build("User").findById(user.getId());  
 List<User> loggedUser = new ArrayList<User>();

//用户名密码比对  
 if (findUser.getPassword().equals(user.getPassword())) {  
 findUser.setPassword("");  
 loggedUser.add(findUser);  
 return UnifiedFunctions.buildQuickSuccess(loggedUser, "登录成功");  
 } else {  
 return UnifiedFunctions.buildQuickError("登录失败");  
 }  
  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 return UnifiedFunctions.buildQuickError("登录失败:" + e.getMessage());  
 }  
  
 }

3.专业选课

前台：

function getAllMajors(major) {

$loading.start("loadingMask");

QueryToolService.queryAllMajors({}, function (res) {

if (res.data) {

$scope.allMajorsInDetails = res.data;

$scope.allMajorsInDetails.sort();

if (major != null) {

for (var i = 0; i < $scope.allMajorsInDetails.length; i++) {

if ($scope.allMajorsInDetails[i].major == major) {

//前台对选课进行数据模型二次封装，便于现实和前台使用

$scope.selectedCourseList = $scope.allMajorsInDetails[i].majorCourse;

break;

}

}

}

}

$loading.finish("loadingMask");

}, function (error) {

$loading.finish("loadingMask");

});

}

$scope.addCourseToMajor = function () {

if(!$scope.selectedCourseToAdd){

noty.show("请先选择课程再按添加!", 'info');

return;

}

//对前台页面数据双向绑定的数值进行实时复制

var pureCourseName = $scope.selectedCourseToAdd.substr(0, $scope.selectedCourseToAdd.indexOf('('));

if($scope.selectedCourseList && $scope.selectedCourseList.indexOf(pureCourseName)>=0){

noty.show("该课程已经添加到该专业中,请勿重复添加!", 'info');

return;

}

var reqObjList = [];

reqObjList.push(getCurrentMajorObject(pureCourseName));

$loading.start("loadingMask");

QueryToolService.addMajors(reqObjList, function (res) {

noty.show("添加成功!", 'success');

getAllMajors($scope.selectedMajor);

$loading.finish("loadingMask");

}, function (err) {

$loading.finish("loadingMask");

});

};

后台：

//添加课程spring controller方法

@RequestMapping(value = "/addCourses", method = RequestMethod.POST, produces = "application/json")

public

@ResponseBody

ResponsePackage addCourses(@RequestBody List<Course> courses) {

try {

ManagerBuilder.build("Course").insertList(courses);

return UnifiedFunctions.buildQuickEmptySuccess("成功");

} catch (ClassNotFoundException e) {

e.printStackTrace();

return UnifiedFunctions.buildQuickError("插入失败:" + e.getMessage());

}

}

4.课程表

前台：

//对数据表的信息进行刷新处理

$scope.refreshTable = function () {

if ($scope.selectedMajor == null)

return;

console.log($scope.selectedMajor);

$loading.start("loadingMask");

//每次刷新表的同事会去进行一次异步请求，动态获取最新课程表信息

QueryToolService.queryAllSchedule({}, function (res) {

if (!res.data) {

noty.show("没有找到相关记录", "error");

}

for (var j = 0; j < res.data.length; j++) {

if (res.data[j].className.indexOf($scope.selectedMajor) >= 0) {

console.log("fid");

$scope.allScheduleInDetails = res.data[j];

break;

}

}

console.dir($scope.allScheduleInDetails);

//由于课程表后台数据类型和前台显示数据类型不一致，所以需要二次进行对数据编排以在页面中显示

var oneClass = $scope.allScheduleInDetails.mondayClass || [];

var twoClass = $scope.allScheduleInDetails.tuesdayClass || [];

var threeClass = $scope.allScheduleInDetails.wednesdayClass || [];

var fourClass = $scope.allScheduleInDetails.thursdayClass || [];

var fiveClass = $scope.allScheduleInDetails.fridayClass || [];

$scope.dataz = [];

//上午

for (var i = 0; i < 2; i++) {

$scope.dataz.push({

timePeriod: (i \* 2 + 7) + ":00" + "~" + (i \* 2 + 9) + ":00",

mondayClass: oneClass[i] || "休息",

tuesdayClass: twoClass[i] || "休息",

wednesdayClass: threeClass[i] || "休息",

thursdayClass: fourClass[i] || "休息",

fridayClass: fiveClass[i] || "休息"

});

}

//下午

for (var i = 0; i < 2; i++) {

$scope.dataz.push({

timePeriod: (i \* 2 + 13) + ":00" + "~" + (i \* 2 + 15) + ":00",

mondayClass: oneClass[i + 2] || "休息",

tuesdayClass: twoClass[i + 2] || "休息",

wednesdayClass: threeClass[i + 2] || "休息",

thursdayClass: fourClass[i + 2] || "休息",

fridayClass: fiveClass[i + 2] || "休息"

});

}

$scope.columns = [];

$scope.columns.push({

title: "时间",

field: 'timePeriod',

visible: true,

showFilter: false

});

$scope.columns.push({

title: "星期一",

field: 'mondayClass',

visible: true,

showFilter: false

});

$scope.columns.push({

title: "星期二",

field: 'tuesdayClass',

visible: true,

showFilter: false

});

$scope.columns.push({

title: "星期三",

field: 'wednesdayClass',

visible: true,

showFilter: false

});

$scope.columns.push({

title: "星期四",

field: 'thursdayClass',

visible: true,

showFilter: false

});

$scope.columns.push({

title: "星期五",

field: 'fridayClass',

visible: true,

showFilter: false

});

$scope.tableParams.reload();

if (res.successful == true && res.data && res.data.length == 0)

noty.show("信息获取失败", 'error');

$loading.finish("loadingMask");

}, function (error) {

$loading.finish("loadingMask");

});

//$scope.tableParams.reload();

};

后台：

//后台获取课程表方法

@RequestMapping(value = "/queryAllSchedule", method = RequestMethod.GET, produces = "application/json")

public

@ResponseBody

ResponsePackage queryAllSchedule() {

try {

List<Student> students = ManagerBuilder.build("Schedule").read();

return UnifiedFunctions.buildQuickSuccess(students, "成功");

} catch (ClassNotFoundException e) {

e.printStackTrace();

return UnifiedFunctions.buildQuickError("搜索失败:" + e.getMessage());

}

}

5.选修课实现

前台：

//选修课方法前台调用

$scope.addOuterCourse = function () {

if ($scope.selOuterCourse.length == 4) {

noty.show("最多只能选择4门课外实践课程!", 'error');

return;

}

var Outer = $("input[name='inlineRadioOptions']:checked").val();

if (Outer && $scope.selOuterCourse.indexOf(Outer) < 0) {

var backCourse = angular.copy($scope.selOuterCourse);

$scope.selOuterCourse.push(Outer);

var saveUser = {};

//选修课获取以后，对数据库中已有数据进行比对和业务逻辑校准。

for(var i=0;i<$rootScope.domainRootInfo.length;i++){

if($rootScope.domainRootInfo[i].id==$scope.selectedCourseNId){

saveUser = $rootScope.domainRootInfo[i];

$rootScope.domainRootInfo[i].shijianCourse = $scope.selOuterCourse;

break;

}

}

saveUser.shijianCourse = $scope.selOuterCourse;

QueryToolService.addPerson(saveUser,function(res){

noty.show("选课成功!","success");

},function(err){

$scope.selOuterCourse = backCourse;

noty.show("选课失败!","error");

});

}

};

//前台对选修课选择的清空操作，清空操作会同时在前台数据缓存中清除和在数据库中清除

$scope.clearOuterCourse = function () {

var backCourse = angular.copy($scope.selOuterCourse);

$scope.selOuterCourse = [];

var saveUser = {};

for(var i=0;i<$rootScope.domainRootInfo.length;i++){

if($rootScope.domainRootInfo[i].id==$scope.selectedCourseNId){

saveUser = $rootScope.domainRootInfo[i];

$rootScope.domainRootInfo[i].shijianCourse = [];

break;

}

}

saveUser.shijianCourse = $scope.selOuterCourse;

QueryToolService.addPerson(saveUser,function(res){

noty.show("清空成功!","success");

},function(err){

$scope.selOuterCourse = backCourse;

noty.show("清空失败!","error");

});

};

//选修课外弹窗口的控制句柄代码

$scope.openOuter = function (id,event) {

event.preventDefault();

$scope.selectedCourseNId = id;

console.log(id);

for(var i=0;i<$rootScope.domainRootInfo.length;i++){

if($rootScope.domainRootInfo[i].id==$scope.selectedCourseNId){

$scope.selOuterCourse = $rootScope.domainRootInfo[i].shijianCourse || [];

break;

}

}

outerPad = $uibModal.open({

animation: true,

size: "md",

//前台显示页面的模板

templateUrl: 'app/partials/modals/exCourse.html',

//controller: 'EDController',

scope: $scope

});

};

后台：

//因为选修课信息是基于用户信息之上的，所以每次保存都调用了addPerson方法进行存储刷新

@RequestMapping(value = "/addPerson", method = RequestMethod.POST, produces = "application/json")

public

@ResponseBody

ResponsePackage addPerson(@RequestBody Student student) {

try {

List<Student> students = ManagerBuilder.build("Student").read();

int nextNumber = 100000;

if (students.size() > 0) {

for (Student student1 : students) {

if (student1.getNumber() > nextNumber) {

nextNumber = student1.getNumber();

}

}

nextNumber = nextNumber + 1;

}

long exist = ManagerBuilder.build("Student").findExist(student.getId());

if (exist == 0) {

student.setNumber(nextNumber);

}

ManagerBuilder.build("Student").insert(student);

List<Student> returnList = new ArrayList<Student>();

returnList.add(student);

return UnifiedFunctions.buildQuickSuccess(returnList, "成功");

} catch (ClassNotFoundException e) {

e.printStackTrace();

return UnifiedFunctions.buildQuickError("插入失败:" + e.getMessage());

}

}