# 五 后台界面

当在前台使用admin登录后，便可以通过连接或者直接输入对应网址进入后台界面。

修改后台渲染的界面：

router.get('/',(req,res,next)=>{  
 res.render('admin/index',{  
 userInfo: req.userInfo  
 });  
});

此时我们给views文件夹中添加上述页面，并使用bootstrap来布局。但是我们要考虑的是，很多时候，页面的内容是重复的，我们需要一个父页面来放置共同信息，子页面来展示具体的页面信息。

父页面 layout.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>后台管理</title>  
 <link rel="stylesheet" href="/public/css/bootstrap.min.css" type="text/css" />  
 <link rel="stylesheet" href="/public/css/bootstrap-theme.min.css" type="text/css" />  
 <script src="/public/lib/jquery-1.12.4.min.js"></script>  
 <script src="/public/lib/bootstrap.min.js"></script>  
</head>  
<body>

<!--bootstrap布局的顶部导航-->

<nav></nav>

{% block main %} {% endblock %}  
  
</body>  
</html>

在这里要注意：{% block main %} {% endblock %}

即是子页面中的具体信息。

子页面admain/index.html的引入：

{% extends 'layout.html' %}  
{% block main %}  
<!--巨幕-->  
<div class="jumbotron">  
 <div class="container">  
 <h1>Hello {{userInfo.username}}</h1>  
 <p>欢迎来到后台管理！</p>  
 </div>  
</div>  
{% endblock %}

# 六 用户管理

*//用户管理*router.get('/user',(req,res,next)=>{  
 /\*  
 limit(number) 限制获取的数据条数  
 skip(number) 忽略数据的条数  
 如果每页展示2条：  
 第1页：1-2， skip：0 -----（当前页-1）\*limit  
 第2页：3-4， skip：2 -----（当前页-1）\*limit  
 第3页：5-6， skip：4 -----（当前页-1）\*limit  
 \*/  
 *//我们通过用户的get来获取在第几页* let page = Number(req.query.page || 1);  
 let limitN = 5;  
 let skipN = (page - 1) \* limitN;  
 let pages = 0; *//总页数* UserM.count().then(function (count) {  
 pages = Math.ceil(count / limitN);  
 page = Math.min(page, pages); *//取值不能超过pages* page = Math.max(page, 1); *//取值不能小于1  
 //count find都是异步的，所以不能让find直接写在count后，应该写在count内部* UserM.find().limit(limitN).skip(skipN).then(function(users){  
 res.render('admin/user',{  
 userInfo: req.userInfo,  
 users: users,  
 page: page,  
 count: count,  
 pages: pages,  
 limitN: limitN  
 });  
 });  
 });  
  
});

前台：

{% extends 'layout.html' %}  
{% block main %}  
<ol class="breadcrumb">  
 <li><a href="#">用户列表</a></li>  
</ol>  
<table class="table table-hover table-striped">  
 <tr>  
 <th>ID</th>  
 <th>用户名</th>  
 <th>密码</th>  
 <th>是否是管理</th>  
 </tr>  
 {% for user in users %}  
 <tr>  
 <td>{{user.\_id.toString()}}</td>  
 <td>{{user.username}}</td>  
 <td>{{user.password}}</td>  
 <td>{% if user.isAdmin %}是{% endif %}</td>  
 </tr>  
 {% endfor %}  
</table>  
  
{% include 'page-layout.html' %}  
  
  
{% endblock %}

其中 page-layout.html 是抽取出的新模板：

<nav aria-label="...">  
 <ul class="pager">  
 <li><a href="/admin/user?page={{page-1}}">上一页</a></li>  
 <li><a href="/admin/user?page={{page+1}}">下一页</a></li>  
 <div>  
 一共有{{count}}条数据，每页显示{{limitN}}条数据，一共有{{pages}}页，当前在第{{page}}页  
 </div>  
 </ul>  
</nav>

# 七 导航

导航的页面布局以及数据提交与 用户管理是一致的，这里不再多说。

值得注意的是，用户数据的增删改使用了前台ajax提交的方式，为了区分，导航数据的提交这里我们使用表单方式：

增加导航前台：

{% extends 'layout.html' %}  
{% block main %}  
<ol class="breadcrumb">  
 <li><a href="#">添加分类</a></li>  
</ol>  
<div class="jumbotron">  
 <div class="container">  
 <form role="form" method="post">  
 <div class="form-group">  
 <label for="name">分类名称：</label>  
 <input type="text" class="form-control" id="exampleInputEmail1" placeholder="请输入分类" name="name">  
 </div>  
 <button type="submit" class="btn btn-default">提交</button>  
 </form>  
 </div>  
</div>  
  
{% endblock %}

后台处理提交：

*//在分类添加页面post提交数据*router.post('/category\_add',function (req,res,next) {  
 console.log(req.body);  
 if(req.body.name == ''){  
 res.render('admin/submitMSG',{  
 userInfo: req.userInfo,  
 errorMsg: '栏目名不能为空'  
 });  
 return;  
 }  
 CategoryM.findOne({  
 categoryname: req.body.name  
 }).then(function (result) {  
 if(result){  
 res.render('admin/submitMSG',{  
 userInfo: req.userInfo,  
 errorMsg: '栏目已存在'  
 });  
 return Promise.reject();  
 }else {  
 return new CategoryM({  
 categoryname:req.body.name  
 }).save();  
 }  
 }).then(function (newCategory) {  
 res.render('admin/submitMSG',{  
 userInfo: req.userInfo,  
 errorMsg: '提交成功',  
 url: '/admin/category' *//跳转页面* });  
 });  
});

注意这里：提交成功有跳转页面，使用了



而对于导航的修改删除，我们则使用这种方式提交：

<table class="table table-hover table-striped">  
 <tr>  
 <th>分类ID</th>  
 <th>分类名</th>  
 <th>操作</th>  
 </tr>  
 {% for category in categories %}  
 <tr>  
 <td>{{category.\_id.toString()}}</td>  
 <td>{{category.categoryname}}</td>  
 <td>  
 <a href="/admin/category\_edit?id={{category.\_id.toString()}}">修改</a>  
 <a href="/admin/category\_delet?id={{category.\_id.toString()}}">删除</a>  
 </td>  
 </tr>  
 {% endfor %}  
</table>

分类修改页面：

*//分类修改页面*router.get('/category\_edit',function (req,res,next) {  
 let id = req.query.id || '';  
 CategoryM.findOne({  
 \_id: id  
 }).then(function (category) {  
 if(!category){  
 res.render('admin/submitMSG',{  
 userInfo: req.userInfo,  
 msg: '分类信息不存在'  
 });  
 } else {  
 res.render('admin/category\_edit',{  
 userInfo:req.userInfo,  
 category:category  
 });  
 }  
 });  
});

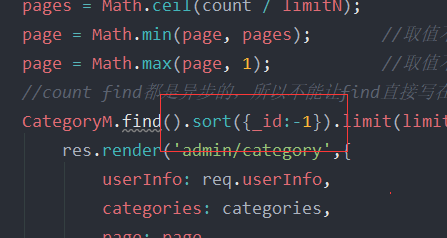
分类修改提交的两种异步写法

router.post('/category\_edit',function (req,res,next) {  
  
 let id = req.query.id || '';  
 let categoryname = req.body.name || '';  
  
 *//提交了空选项* if(categoryname == ''){  
 res.render('admin/submitMSG',{  
 userInfo:req.userInfo,  
 msg:'非法提交'  
 });  
 }else {  
 (async ()=>{  
  
 let category = await CategoryM.findOne({  
 \_id: id  
 });  
  
 if(categoryname == category.categoryname){  
 res.render('admin/submitMSG',{  
 userInfo: req.userInfo,  
 msg: '没有进行修改',  
 url: '/admin/category'  
 });  
 } else {  
 let isSame = await CategoryM.findOne({  
 \_id: {$ne:id},  
 categoryname:categoryname  
 });  
  
 if(isSame){  
 res.render('admin/submitMSG', {  
 userInfo: req.userInfo,  
 msg: '数据库中已经存在同名分类'  
 });  
 } else {  
  
 await CategoryM.update({  
 \_id: id  
 }, {  
 categoryname: categoryname  
 });  
  
 res.render('admin/submitMSG', {  
 userInfo: req.userInfo,  
 msg: '修改成功',  
 url: '/admin/category'  
 });  
  
 }  
 }  
  
 })();  
 *// CategoryM.findOne({  
 // \_id: id  
 // }).then(function (category) {  
 // //如果没有修改直接提交  
 // if(categoryname == category.categoryname){  
 // res.render('admin/submitMSG',{  
 // userInfo: req.userInfo,  
 // msg: '没有进行修改',  
 // url: '/admin/category'  
 // });  
 // return Promise.reject();  
 // } else {  
 // return CategoryM.findOne({  
 // \_id: {$ne:id},  
 // categoryname:categoryname  
 // });  
 // }  
 // }).then(function(sameCategory) {  
 // if (sameCategory) {  
 // res.render('admin/submitMSG', {  
 // userInfo: req.userInfo,  
 // msg: '数据库中已经存在同名分类'  
 // });  
 // return Promise.reject();  
 // } else {  
 // return CategoryM.update({  
 // \_id: id  
 // }, {  
 // categoryname: categoryname  
 // });  
 // }  
 // }).then(function() {  
 // res.render('admin/submitMSG', {  
 // userInfo: req.userInfo,  
 // msg: '修改成功',  
 // url: '/admin/category'  
 // });  
 // })* }  
});

删除分类：

*//分类删除*router.get('/category\_delet',function (req,res,next) {  
 let id = req.query.id || '';  
 CategoryM.remove({  
 \_id:id  
 }).then(function () {  
 res.render('admin/submitMSG', {  
 userInfo: req.userInfo,  
 msg: '删除成功',  
 url: '/admin/category'  
 });  
  
 });  
});

分类排序：



升序 1

降序 -1