逻辑功能实现和路由限制

一、逻辑功能实现

接下来将会一步步完成之前指定的功能：

1. 注册

1.1 建立表单

修改 /louNote/views/register.ejs 文件：

```
<%- include header %>
       <div class="container">
          <div class="row">
             <div class="col-xs-4 col-xs-offset-4">
                <form method="post">
                   <div class="form-group">
                      <input type="text" class="form-control" name="username" placeholder="用户名">
                   </div>

                   <div class="form-group">
                      <input type="password" class="form-control" name="password" placeholder="密码">
                   </div>

                   <div class="form-group">
                      <input type="password" class="form-control" name="passwordRepeat" placeholder="确认密
码">
                   </div>

                   <button type="submit" class="btn btn-default center-block">注册</button>
                </form>
             </div>
          </div>
       <div>
    </body>
</html>
```

1.2 编写逻辑代码

修改 /louNote/app.js 文件：

```
// get 请求
app.get('/reg', function(req, res) {
    res.render('register', {
        title: '注册',
        user: req.session.user,
        page: 'reg'
    });
});


// post 请求
app.post('/reg', function(req, res) {
    var username = req.body.username,
        password = req.body.password,
        passwordRepeat = req.body.passwordRepeat;

    //检查两次输入的密码是否一致
    if(password != passwordRepeat) {
        console.log('两次输入的密码不一致! ');
        return res.redirect('/reg');
    }

    //检查用户名是否已经存在
    User.findOne({username:username}, function(err, user) {
        if(err) {
            console.log(err);
            return res.redirect('/reg');
        }

        if(user) {
            console.log('用户名已经存在');
            return res.redirect('/reg');
        }

        //对密码进行md5加密
        var md5 = crypto.createHash('md5'),
            md5password = md5.update(password).digest('hex');

        var newUser = new User({
            username: username,
            password: md5password
        });

        newUser.save(function(err, doc) {
            if(err) {
                console.log(err);
                return res.redirect('/reg');
            }
            console.log('注册成功! ');
            newUser.password = null;
            delete newUser.password;
            req.session.user = newUser;
            return res.redirect('/');
        });
    });
});
```

2. 登录

2.1 建立表单

修改 /louNote/views/login.ejs 文件:

```
<%- include header %>
        <div class="container">
             <div class="row">
                 <div class="col-xs-4 col-xs-offset-4">
                     <form method="post">
                         <div class="form-group">
                             <input type="text" class="form-control" name="username" placeholder="用户名">
                         </div>

                         <div class="form-group">
                             <input type="password" class="form-control" name="password" placeholder="密码">
                         </div>

                         <button type="submit" class="btn btn-default center-block">登录</button>
                     </form>
                 </div>
             </div>
         </div>
    </body>
</html>
```

2.2 编写逻辑代码

修改 `/louNote/app.js` 文件：

```
app.get('/login', function(req, res) {
    res.render('login', {
       title: '登录',
       user: req.session.user,
       page: 'login'
    });
});

app.post('/login', function(req, res) {
    var username = req.body.username,
        password = req.body.password;

    User.findOne({username:username}, function(err, user) {
        if(err) {
            console.log(err);
            return next(err);
        }
        if(!user) {
            console.log('用户不存在! ');
            return res.redirect('/login');
        }
        //对密码进行md5加密
        var md5 = crypto.createHash('md5'),
            md5password = md5.update(password).digest('hex');
        if(user.password !== md5password) {
            console.log('密码错误! ');
            return res.redirect('/login');
        }
        console.log('登录成功! ');
        user.password = null;
        delete user.password;
        req.session.user = user;
        return res.redirect('/');
    });
});
```

3. 会话机制

完成以上两步就实现了项目的注册和登录功能，但运行时会报错，原因有二：

- 上一个实验的末尾创建了两个数据模型 `userSchema` 和 `noteSchema`，我们还没有引入模型就开始数据操作，当然会报错，需要修改 `app.js` 文件：

```
// 引入模型并实例化
var models = require('./models/models');
var User = models.User;
var Note = models.Note;
```

- 是为了方便讲解，提前运用 session 机制，但还没有引入相关的模块。接着就开始学习运用 web 开发里很常见的 session 机制。

3.1 建立 session 模型

在 Express 框架中需要安装两个模块 `express-session` 和 `connect-mongo`，执行命令 - `npm install --save express-session connect-mongo`，并修改 `app.js` 文件

```
// 引入建立 session 必备的模块
var session = require('express-session');
var MongoStore = require('connect-mongo')(session);

// 建立 session 模型
app.use(session({
    key: 'session',
    secret: 'Keboard cat',
    cookie: {maxAge: 1000 * 60 * 60 * 24},
    store: new MongoStore({
        db: 'notes',
        mongooseConnection: mongoose.connection
    }),
    resave: false,
    saveUninitialized: true
}));
```

3.2 保存会话

项目需要通过 seesion 判断用户的登录状态，所以只需要保存用户的登录后的基本信息，注册和登录的操作就需要保存这一信息，也就是上面代码中的 `req.session.user = user` 一句。

有了保存会话功能，我们就能通过判断用户的登录状态来显示不同的信息，修改 `/louNote/views/header.ejs` 文件：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title><%= title %></title>
        <link rel="stylesheet" href="/bower_components/bootstrap/dist/css/bootstrap.min.css">
        <link rel="stylesheet" href="/css/common.css">
        <script src="/bower_components/jquery/dist/jquery.min.js"></script>
        <script src="/bower_components/bootstrap/dist/js/bootstrap.min.js"></script>
    </head>
<body>
    <div class="clearfix page-nav">
    <% if(user) { %>
        <div class="pull-left nav-username">Hello! <a href="/"><%= user.username %></a></div>
        <div class="pull-left"><a href="/post">发表</a></div>
        <div class="pull-right nav-quit"><a href="/quit">退出</a></div>
    <% } else { %>
        <% if(page == 'login') { %>
            <div class="pull-right nav-register"><a href="/reg">注册</a></div>
        <% } else { %>
            <div class="pull-right nav-login"><a href="/login">登录</a></div>
        <% } %>
    <% } %>

    </div>
    <div class="text-center page-title"><%= title %></div>
```

当用户登录后，就能看到发布和退出按钮，未登录时则看到注册和登录按钮，这也很符合逻辑操作。

4. 发布笔记

4.1 建立表单

修改 `/louNote/views/post.ejs` 文件：

```
<%- include header %>
        <div class="container">
             <div class="row">
                 <div class="col-xs-4 col-xs-offset-4">
                     <form method="post">

                         <div class="form-group">
                             <input type="text" class="form-control" name="title" placeholder="标题">
                         </div>

                         <div class="form-group">
                             <input type="text" class="form-control" name="tag" placeholder="标签">
                         </div>
                         <div class="form-group">
                             <textarea class="form-control" name="content" placeholder="文章内容"></textarea>
                         </div>

                         <button type="submit" class="btn btn-default center-block">发表</button>
                     </form>
                 </div>
             </div>
         </div>
    </body>
</html>
```

4.2 编写逻辑代码

修改 /louNote/app.js 文件：

```
app.get('/post', function(req, res) {
    res.render('post', {
        title: '发布',
        user: req.session.user
    })
});

app.post('/post', function(req, res) {
    var note = new Note({
        title: req.body.title,
        author: req.session.user.username,
        tag: req.body.tag,
        content: req.body.content
    });

    note.save(function(err, doc) {
        if(err) {
        console.log(err);
            return res.redirect('/post');
        }
        console.log('文章发表成功！')
        return res.redirect('/');
    });
});
```

5. 笔记列表和笔记详情

笔记列表和笔记详情除了查询条件不同，其余并没有很大的区别，所以这里放到一起讲解：

修改 /louNote/views/index.ejs 文件：

```
<%- include header %>
        <div class="container">
            <div class="row">
                <% arts.forEach(function(art) { %>
                    <div class="col-xs-8 col-xs-offset-2">
                        <div class="article-box">
                            <a href="/detail/<%= art._id %>">
                                <h3 class="text-center"><%= art.title %></h3>
                            </a>
                            <p class="text-center">
                                <span class="label label-info"><%= art.tag %></span> |
                                <span class="create-time">
                                    <span class="text-bold">时间:</span>
                                    <%= art.createTime %>
                                </span>
                            </p>
                            <p><%= art.content %></p>
                        </div>
                    </div>
                <% }); %>
            </div>
        </div>
    </body>
</html>
```

这里还运用到了 ejs 模板另一标签 `<% %>`，可以直接执行 javascript 语句，这里我们遍历了由模板传来的 `arts` 数据，就生成了所有的笔记列表。

笔记详情只展示其中一个，所以就不需要这个遍历。

编辑笔记列表和笔记详情的逻辑代码，修改 `/louNote/app.js` 文件：

```
// 笔记列表
app.get('/', function(req, res) {
    Note.find({author: req.session.user.username})
        .exec(function(err, arts) {
            if(err) {
                console.log(err);
                return res.redirect('/');
            }
            res.render('index', {
                title: '笔记列表',
                user: req.session.user,
                arts: arts,
                moment: moment
            });
        })
});

// 笔记详情
app.get('/detail/:_id', function(req, res) {
    Note.findOne({_id: req.params._id})
        .exec(function(err, art) {
            if(err) {
                console.log(err);
                return res.redirect('/');
            }
            if(art) {
                res.render('detail', {
                    title: '笔记详情',
                    user: req.session.user,
                    art: art,
                    moment: moment
                });
            }
        });
});
```

其中我们可以看到两者的区别就是查询条件，一个是 `find()` 方法，另一个是 `findOne()` 方法，由字面意义也能看出前者返回所有数据，后者只返回一个数据，这样就分别是实现了列表和详情的不需求展示。

二、路由限制

1. 逻辑设计

完成以上步骤，项目的基本功能已经初步实现，但还存在一些 bug 需要修复，比如当我们未登录时，访问 `localhost:3000` 会因为 session 未保存用户登录信息，会报出错误。这就需要我们对访问做出限制。

1.1 登录

当用户已登录，就不能通过输入 url 访问登录和注册页面。

1.2 未登录

当用户未登录时，就不能通过输入 url 访问笔记列表页、笔记详情和发布笔记页面。

2. 检测登录状态

新建 `/louNote/checkLogin.js` 文件，并敲入代码：

```
// 已登录
function login(req, res, next) {
    if(req.session.user) {
        console.log('您已经登录！');
        return res.redirect('back');//返回之前的页面
    }
    next();
}

// 未登录
function noLogin(req, res, next) {
    if(!req.session.user) {
        console.log('抱歉，您还没有登录！');
        return res.redirect('/login');//返回登录页面
    }
    next();
}

exports.login = login;
exports.noLogin = noLogin;
```

修改 `app.js` 文件：

```
// 引入检测登录文件
var checkLogin = require('./checkLogin.js');

// 在注册和登录的 get 请求前加入已登录判断
app.get('/', checkLogin.login);
app.get('/reg', function(req, res) {
    ...
});

app.get('/', checkLogin.login);
app.get('/login', function(req, res) {
    ...
});

// 在笔记列表、发布笔记和笔记详情前加入未登录判断
app.get('/', checkLogin.noLogin);
app.get('/', function(req, res) {
    ...
});

app.get('/', checkLogin.noLogin);
app.get('/post', function(req, res) {
    ...
});

app.get('/', checkLogin.noLogin);
app.get('/detail/:_id', function(req, res) {
    ...
});
```

这样就基本完成了项目功能，当然还有许多可扩展的功能和可以优化的细节，比如，允许修改、删除和查询笔记，在显示笔记的创建时间可以通过 `moment.js` 进行优化。希望大家能在已有的基础上自由发挥。

上一节 (/courses/446/labs/1420/document)

课程教师

Tryltry

共发布过**7**门课程

查看老师的所有课程 > (/teacher/9061)

前置课程

Node.js 教程 (/courses/44)

Node.js包教不包会 (/courses/493)

动手做实验，轻松学IT

(http://weibo.com/shiyanlou2013)