# 目标

- 能够利用 `ajaxPrefilter` 统一设置请求头
- 完成获取用户信息功能
- 能够利用 `ajaxPrefilter` 统一控制用户访问权限
- 能够完成基本资料的功能
- 能够完成重置密码功能
- 能够完成更换头像功能
- 参照文档使用 `cropper` 图片裁剪功能
- 知道 `base64` 格式图片的优缺点
- 利用 `layer.open` 自定义弹出层效果

# 1. 后台主页

## 1.1 获取用户基本信息(☆☆☆)

在用户登录成功，进入后台页面后，我们应该立刻获取用户的信息，然后把对应的信息展示在页面上

- 在后台的 `index.html` 页面中引入需要的 `js` 文件

```
<!-- 导入 jQuery -->
<script src="/assets/lib/jquery.js"></script>
<!-- 导入自己封装的 baseAPI -->
<script src="/assets/js/baseAPI.js"></script>
<!-- 导入自己的 js 文件 -->
<script src="/assets/js/index.js"></script>
```

- 定义 `getUserInfo` 函数，当页面加载完毕之后调用这个函数
- 利用 `$.ajax()` 进行网络请求，查阅文档，获取关键信息

**说明**

1. 项目的请求根路径为 `http://ajax.frontend.itheima.net`
2. 以 `/api` 开头的请求路径，不需要访问权限
3. 以 `/my` 开头的请求路径，需要在请求头中携带 `Authorization` 身份认证字段，才能正常访问成功

- 我们请求的时候就需要设置请求头信息，把我们获取到的 `token` 传递给后台
- 数据获取失败提示用户

**示例代码**

```
// 获取用户的基本信息
function getUserInfo() {
  $.ajax({
```

```
    method: 'GET',
    url: '/my/userinfo',
    // headers 就是请求头配置对象
    headers: {
      Authorization: localStorage.getItem('token') || ''
    },
    success: function(res) {
      if (res.status !== 0) {
        return layui.layer.msg('获取用户信息失败! ')
      }
      // 调用 renderAvatar 渲染用户的头像
      renderAvatar(res.data)
    }
  })
}
```

## 1.2 渲染用户头像和名称

如果请求成功，我们根据服务器返回的数据来渲染页面

- 定义 `renderAvatar` 函数，接收服务器返回的用户数据
- 获取用户的名称
- 设置欢迎的文本，找到关键元素进行设置
- 按需渲染用户的头像，如果用户有头像，那么就直接设置图片头像，如果没有设置文本头像

**示例代码**

```
// 渲染用户的头像
function renderAvatar(user) {
  // 1. 获取用户的名称
  var name = user.nickname || user.username
  // 2. 设置欢迎的文本
  $('#welcome').html('欢迎  ' + name)
  // 3. 按需渲染用户的头像
  if (user.user_pic !== null) {
    // 3.1 渲染图片头像
    $('.layui-nav-img')
      .attr('src', user.user_pic)
      .show()
    $('.text-avatar').hide()
  } else {
    // 3.2 渲染文本头像
    $('.layui-nav-img').hide()
    var first = name[0].toUpperCase()
    $('.text-avatar')
      .html(first)
      .show()
  }
}
```

## 1.3 统一为有权限的接口设置`headers`请求头(☆☆☆)

还记得之前的 `ajaxPrefilter` 函数吗，在我们调用 `$.ajax()` 后，并且在请求服务器之前调用的过滤器，那么我们能统一设置根路径，那么我们就可以去统一设置请求头

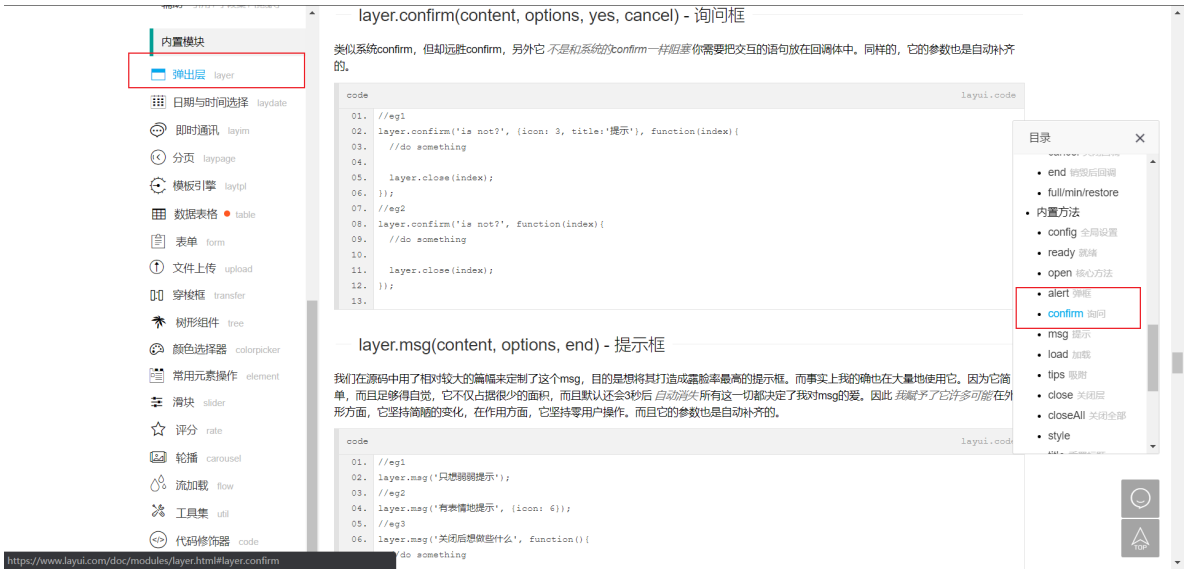
- 在 `baseAPI` 的 `ajaxPrefilter` 中添加如下代码

- 判断 `url` 里面是否携带 `/my/`
- 如果携带，那么我们就设置 `options.headers`

```
// 统一为有权限的接口，设置 headers 请求头
if (options.url.indexOf('/my/') !== -1) {
  options.headers = {
    Authorization: localStorage.getItem('token') || ''
  }
}
```

# 1.4 实现退出功能

- 给退出按钮绑定点击事件，取消 a 标签的默认行为
- 用户点击后，弹出提示框（`layui` 中有弹出层的相关代码），如果用户点击确认




- 移除本地缓存的 token，并且跳转到登录页面

**示例代码**

```
var layer = layui.layer

// 点击按钮，实现退出功能
$('#btnLogout').on('click', function() {
  // 提示用户是否确认退出
  layer.confirm('确定退出登录?', { icon: 3, title: '提示' }, function(index) {
    //do something
    // 1. 清空本地存储中的 token
    localStorage.removeItem('token')
    // 2. 重新跳转到登录页面
    location.href = '/login.html'

    // 关闭 confirm 询问框
    layer.close(index)
  })
})
```

# 1.5 控制用户的访问权限

用户如果没有登录，是否能够允许用户访问后台主页？肯定是不能的，所以我们需要进行权限的校验，可以利用请求后服务器返回的状态来决定

- 在调用有权限接口的时候，指定`complete`回调函数，这个回调函数不管成功还是失败都会调用
- 在回调里面判断 服务器返回的的状态是否等于 1，并且错误的信息是 "身份认证失败"，如果成立，那么就强制用户跳转到登录页

```
// 不论成功还是失败，最终都会调用 complete 回调函数
complete: function(res) {
  // console.log('执行了 complete 回调：')
  // console.log(res)
  // 在 complete 回调函数中，可以使用 res.responseJSON 拿到服务器响应回来的数据
  if (res.responseJSON.status === 1 && res.responseJSON.message === '身份认证失
败！') {
    // 1. 强制清空 token
    localStorage.removeItem('token')
    // 2. 强制跳转到登录页面
    location.href = '/login.html'
  }
}
```

## 1.6 优化权限控制的代码(☆☆☆)

目前有一个问题，每一个权限的请求难道都需要去判断一次 complete的逻辑代码吗?借助统一设置请求根路径和请求头的思路，我们也可以统一来设置 complete 函数

- 将权限控制的代码，从每个请求中，抽离到`ajaxPrefilter`中
- 利用`options`来挂在统一的`complete`函数

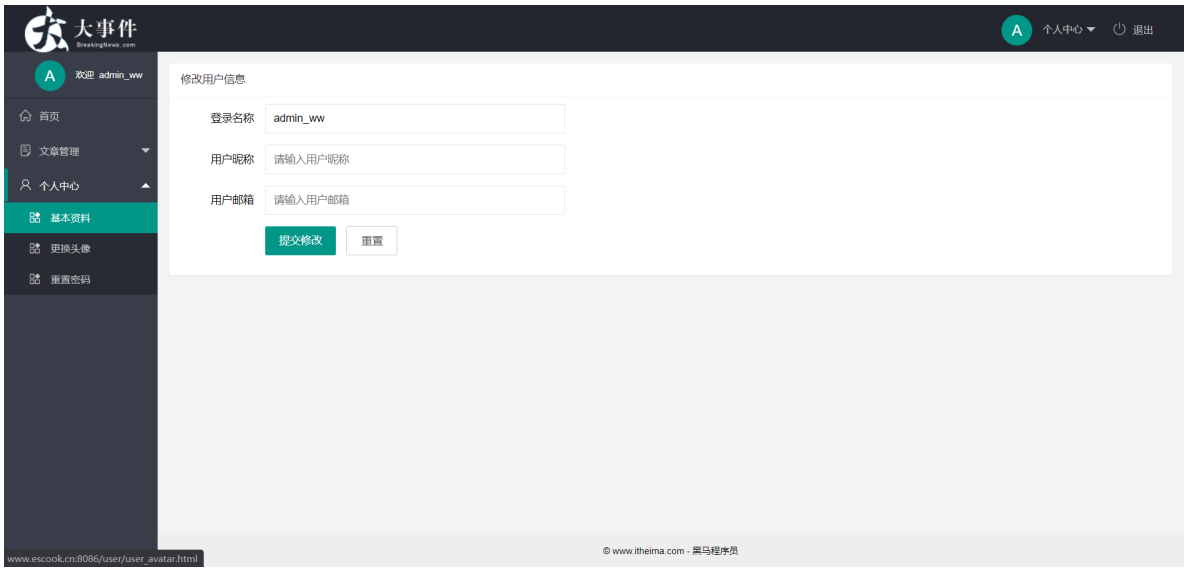
```
// 注意：每次调用 $.get() 或 $.post() 或 $.ajax() 的时候，
// 会先调用 ajaxPrefilter 这个函数
// 在这个函数中，可以拿到我们给Ajax提供的配置对象
$.ajaxPrefilter(function(options) {
  ...
  // 全局统一挂载 complete 回调函数
  options.complete = function(res) {
    // console.log('执行了 complete 回调：')
    // console.log(res)
    // 在 complete 回调函数中，可以使用 res.responseJSON 拿到服务器响应回来的数据
    if (res.responseJSON.status === 1 && res.responseJSON.message === '身份认证失
败！') {
      // 1. 强制清空 token
      localStorage.removeItem('token')
      // 2. 强制跳转到登录页面
      location.href = '/login.html'
    }
  }
})
```

## 1.7 将本地代码同步到云端

- `git status`查看文件状态
- `git branch`查看分支
- `git add .`添加到暂存区

- `git commit -m 提交消息`
- `git push -u origin index` 推送到远程分支
- `git checkout master` 切换到主分支
- `git merge index` 合并分支
- `git push` 推送到云端

# 2. 基本资料(☆☆☆)




## 2.1 创建基本资料对应的页面

- 新建一个 `user` 文件夹
- 新建 `/user/user_info.html` 页面
- 在 `index.html` 里面指定，点击对应的 `a` 标签要打开的页面，并且指定在哪里进行打开

```
<a href="/user/user_info.html" target="fm"><i class="layui-icon layui-icon-app">
</i>基本资料</a>
```

- 搭建页面结构，利用 `layui` 的卡片的组件来实现


**结构示例代码**

```
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <!-- 导入 layui 的样式 -->
    <link rel="stylesheet" href="/assets/lib/layui/css/layui.css" />
    <!-- 导入自己的样式 -->
    <link rel="stylesheet" href="/assets/css/user/user_info.css" />
  </head>
  <body>
    <!-- 卡片区域 -->
    <div class="layui-card">
      <div class="layui-card-header">修改用户信息</div>
      <div class="layui-card-body">
        卡片式面板面板通常用于非白色背景色的主体内<br />
        从而映衬出边框投影
      </div>
    </div>
  </body>
</html>
```

- 新建 `/assets/css/user/user_info.css`

```
html,
body {
  margin: 0;
  padding: 0;
}

body {
  background-color: #f2f3f5;
  padding: 15px;
}
```

## 2.2 绘制基本资料对应的表单

- 利用 `layui` 中表单元素来搭建
- 一共使用 三个 input 输入框的结构
- 还需要一个 提交按钮 和 重置按钮

```
<!-- form 表单区域 -->
<form class="layui-form" action="">
  <div class="layui-form-item">
    <label class="layui-form-label">登录名称</label>
    <div class="layui-input-block">
      <input type="text" name="username" required lay-verify="required"
placeholder="请输入登录名称" autocomplete="off" class="layui-input" readonly />
    </div>
  </div>
  <div class="layui-form-item">
    <label class="layui-form-label">用户昵称</label>
    <div class="layui-input-block">
```

```
      <input type="text" name="nickname" required lay-verify="required|nickname"
placeholder="请输入用户昵称" autocomplete="off" class="layui-input" />
    </div>
  </div>
  <div class="layui-form-item">
    <label class="layui-form-label">用户邮箱</label>
    <div class="layui-input-block">
      <input type="text" name="email" required lay-verify="required|email"
placeholder="请输入用户邮箱" autocomplete="off" class="layui-input" />
    </div>
  </div>
  <div class="layui-form-item">
    <div class="layui-input-block">
      <button class="layui-btn" lay-submit lay-filter="formDemo">提交修改
</button>
      <button type="reset" class="layui-btn layui-btn-primary">重置</button>
    </div>
  </div>
</form>
```

- 在页面底部导入如下的脚本

```
<!-- 导入 layui 的 js -->
<script src="/assets/lib/layui/layui.all.js"></script>
<!-- 导入 jquery -->
<script src="/assets/lib/jquery.js"></script>
<!-- 导入自己的 js -->
<script src="/assets/js/user/user_info.js"></script>
```

- 在 `user_info.js` 中校验表单数据

```
$(function() {
  var form = layui.form
  form.verify({
    nickname: function(value) {
      if (value.length > 6) {
        return '昵称长度必须在 1 ~ 6 个字符之间！'
      }
    }
  })
})
```

## 2.3 获取用户的基本信息

- 导入 `baseAPI`
- 在 `user_info.js` 中定义并调用 `initUserInfo` 函数
- 在 `initUserInfo` 函数中 调用 `$.ajax()` 获取用户基本信息
- 如果返回的 status 为0，那么代表成功，如果不等于，代表失败，利用 `layer`进行提示

```
  initUserInfo()

  // 初始化用户的基本信息
  function initUserInfo() {
    $.ajax({
```

```
      method: 'GET',
      url: '/my/userinfo',
      success: function(res) {
        if (res.status !== 0) {
          return layer.msg('获取用户信息失败! ')
        }
        console.log(res)
      }
    })
  }
```

## 2.4 为表单快速赋值

利用 `form.val()` 进行快速赋值，赋值之前我们需要给 `form` 表单添加 一个 `lay-filter` 的属性

- 为表单指定 `lay-filter` 属性

```
<form class="layui-form" lay-filter="formUserInfo"></form>
```

- 调用 `form.val()` 方法为表单赋值

```
form.val('formUserInfo', res.data)
```

- 使用隐藏域保存用户的 `id` 值

```
<!-- form 表单区域 -->
<form class="layui-form" lay-filter="formUserInfo">
  <!-- 这是隐藏域 -->
  <input type="hidden" name="id" value="" />

  <!-- 省略其他代码 -->
</form>
```

## 2.5 实现表单的重置效果

用户点击了重置按钮后，表单里面的内容应该被重置

- 阻止表单的默认重置行为
- 重新调用 `initUserInfo()` 函数，重新获取用户信息

```
// 重置表单的数据
$('#btnReset').on('click', function(e) {
  // 阻止表单的默认重置行为
  e.preventDefault()
  initUserInfo()
})
```

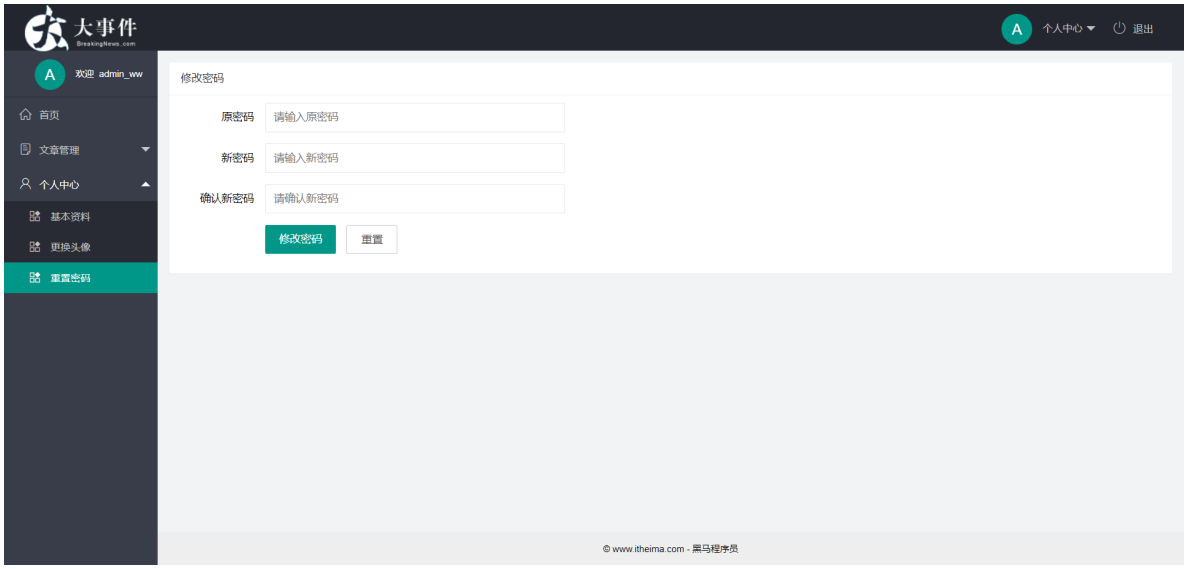
## 2.6 发起请求更新用户的信息

- 监听表单提交事件，在事件处理函数里面取消默认行为
- 查阅接口文档，利用 `$.ajax()` 发起 `post` 请求
- 利用 `$(this).serialize()` 获取表单数据

- 如果返回的 status 不为0，说明更新失败，及逆行提示
- 更新成功之后，调用父页面中的方法，重新渲染用户的头像和用户信息

```
// 监听表单的提交事件
$('.layui-form').on('submit', function(e) {
  // 阻止表单的默认提交行为
  e.preventDefault()
  // 发起 ajax 数据请求
  $.ajax({
    method: 'POST',
    url: '/my/userinfo',
    data: $(this).serialize(),
    success: function(res) {
      if (res.status !== 0) {
        return layer.msg('更新用户信息失败！')
      }
      layer.msg('更新用户信息成功！')
      // 调用父页面中的方法，重新渲染用户的头像和用户的信息
      window.parent.getUserInfo()
    }
  })
})
```

# 3. 重置密码(☆☆☆)




## 3.1 渲染重置密码的页面结构

- 新建重置密码的页面，`user_pwd.html`，给侧边栏对应a标签设置`href`的路径，以及`target`属性
- 利用`layui`的卡片（页面元素 > 面板 > 卡片布局） 搭建结构
- 利用`layui`的表单元素（页面元素 > 表单）来搭建里面内容
- 修改下里面的文字内容

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```
    <title>Document</title>
    <link rel="stylesheet" href="/assets/lib/layui/css/layui.css" />
    <link rel="stylesheet" href="/assets/css/user/user_pwd.css" />
  </head>
  <body>
    <!-- 卡片区域 -->
    <div class="layui-card">
      <div class="layui-card-header">修改密码</div>
      <div class="layui-card-body">
        <form class="layui-form">
          <div class="layui-form-item">
            <label class="layui-form-label">原密码</label>
            <div class="layui-input-block">
              <input type="password" name="oldPwd" required lay-
verify="required" placeholder="请输入原密码" autocomplete="off" class="layui-
input" />
            </div>
          </div>
          <div class="layui-form-item">
            <label class="layui-form-label">新密码</label>
            <div class="layui-input-block">
              <input type="password" name="newPwd" required lay-
verify="required" placeholder="请输入新密码" autocomplete="off" class="layui-
input" />
            </div>
          </div>
          <div class="layui-form-item">
            <label class="layui-form-label">确认新密码</label>
            <div class="layui-input-block">
              <input type="password" name="rePwd" required lay-verify="required"
placeholder="请再次确认密码" autocomplete="off" class="layui-input" />
            </div>
          </div>
          <div class="layui-form-item">
            <div class="layui-input-block">
              <button class="layui-btn" lay-submit lay-filter="formDemo">修改密码
</button>
              <button type="reset" class="layui-btn layui-btn-primary">重置
</button>
            </div>
          </div>
        </form>
      </div>
    </div>
  </body>
</html>
```

- 在 `/assets/css/user/user_pwd.css` 中编写相关样式

```
html,
body {
  margin: 0;
  padding: 0;
}

body {
  padding: 15px;
```

```
  background-color: #f2f3f5;
}

.layui-form {
  width: 500px;
}
```

## 3.2 为密码框定义校验规则

- 导入相关 `js` 脚本 （`layui的js > jQuery的js > baseAPI.js > 导入自己的js`）

```
<!-- 导入 layui 的 js -->
<script src="/assets/lib/layui/layui.all.js"></script>
<!-- 导入 jQuery -->
<script src="/assets/lib/jquery.js"></script>
<!-- 导入 baseAPI -->
<script src="/assets/js/baseAPI.js"></script>
<!-- 导入自己的 js -->
<script src="/assets/js/user/user_pwd.js"></script>
```

- 导入 form 模块
- 利用 `form.verify()` 来定义规则
  - 长度必须是6到12位
  - 不能与旧密码一致
  - 两次密码是否相同
- 在对应的表单元素中，利用 `lay-verify` 来设置自定义校验规则

```
$(function() {
  var form = layui.form

  form.verify({
    pwd: [/^[\S]{6,12}$/, '密码必须6到12位，且不能出现空格'],
    samePwd: function(value) {
      if (value === $('[name=oldPwd]').val()) {
        return '新旧密码不能相同！'
      }
    },
    rePwd: function(value) {
      if (value !== $('[name=newPwd]').val()) {
        return '两次密码不一致！'
      }
    }
  })
})
```

- 为密码框分别添加对应的校验规则

```
<form class="layui-form">
  <div class="layui-form-item">
    <label class="layui-form-label">原密码</label>
    <div class="layui-input-block">
      <input type="password" name="oldPwd" required lay-verify="required|pwd"
placeholder="请输入原密码" autocomplete="off" class="layui-input" />
    </div>
```

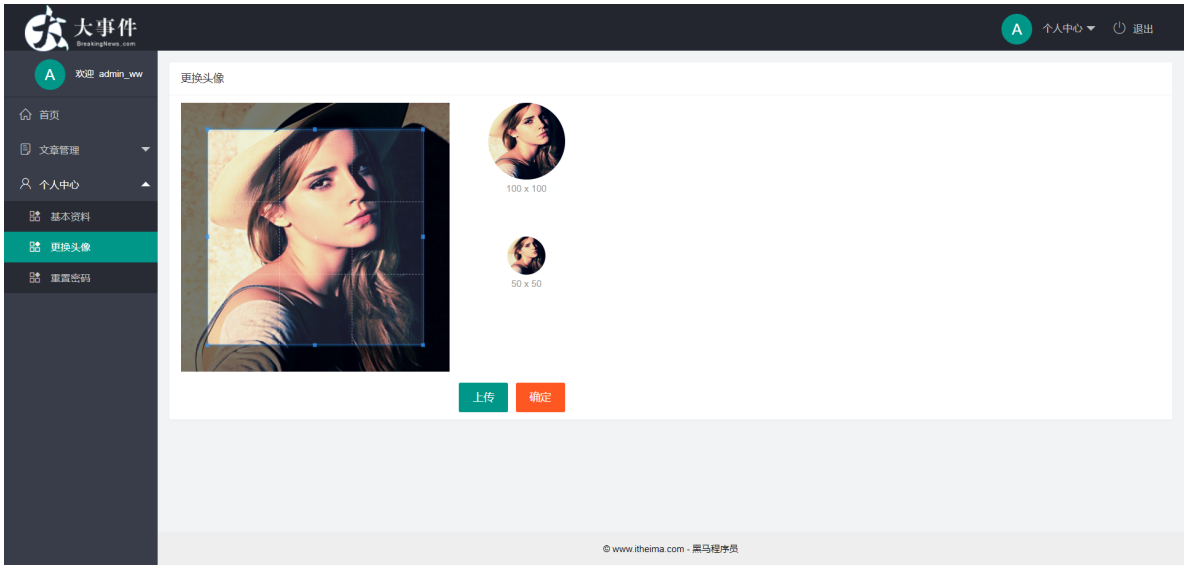
```
  </div>
  <div class="layui-form-item">
    <label class="layui-form-label">新密码</label>
    <div class="layui-input-block">
      <input type="password" name="newPwd" required lay-
verify="required|pwd|samePwd" placeholder="请输入新密码" autocomplete="off"
class="layui-input" />
    </div>
  </div>
  <div class="layui-form-item">
    <label class="layui-form-label">确认新密码</label>
    <div class="layui-input-block">
      <input type="password" name="rePwd" required lay-
verify="required|pwd|rePwd" placeholder="请再次确认密码" autocomplete="off"
class="layui-input" />
    </div>
  </div>
  <div class="layui-form-item">
    <div class="layui-input-block">
      <button class="layui-btn" lay-submit lay-filter="formDemo">修改密码
</button>
      <button type="reset" class="layui-btn layui-btn-primary">重置</button>
    </div>
  </div>
</form>
```

## 3.3 发起请求实现重置密码的功能

- 给form表单绑定 submit 事件，在事件处理函数里面取消默认行为
- 查阅接口文档，利用 `$.ajax()` 来发送 `post` 请求
- 利用 `$(this).serialize()` 来设置提交的数据
- 如果 服务器返回的 `status` 不等于0，说明失败，利用 l`ayui.layer.msg` 来进行提示
- 如果更新成功，重置一下表单内容

```
$('.layui-form').on('submit', function(e) {
  e.preventDefault()
  $.ajax({
    method: 'POST',
    url: '/my/updatepwd',
    data: $(this).serialize(),
    success: function(res) {
      if (res.status !== 0) {
        return layui.layer.msg('更新密码失败！')
      }
      layui.layer.msg('更新密码成功！')
      // 重置表单
      $('.layui-form')[0].reset()
    }
  })
})
```

# 4. 更换头像




# 4.1 初步渲染更换头像页面的结构

- 创建 `/user/user_avatar.html` 页面
- 放入卡片区域 (页面元素 > 面板 > 卡片)，修改对应文字信息
- 在侧边栏对应的 `a` 标签设置 `href` 属性 和 `target` 属性

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <link rel="stylesheet" href="/assets/lib/layui/css/layui.css" />
    <link rel="stylesheet" href="/assets/css/user/user_avatar.css" />
  </head>
  <body>
    <!-- 卡片区域 -->
    <div class="layui-card">
      <div class="layui-card-header">更换头像</div>
      <div class="layui-card-body">
        卡片式面板面板通常用于非白色背景色的主体内<br />
        从而映衬出边框投影
      </div>
    </div>
  </body>
</html>
```

- 修改一下对应样式

```
html,
body {
  margin: 0;
  padding: 0;
}

body {
  padding: 15px;
  background-color: #f2f3f5;
}
```

- 修改 `index.html` 中对应链接的属性

```
<a href="/user/user_avatar.html" target="fm"><i class="layui-icon layui-icon-app"></i>更换头像</a>
```

# 4.2 实现裁剪区域图片的替换(☆☆☆)

## 4.2.1 cropper 图片裁剪

- 在 `<head>` 中导入 `cropper.css` 样式表：

```
<link rel="stylesheet" href="/assets/lib/cropper/cropper.css" />
```

- 在`<body>`的结束标签之前，按顺序导入如下的 `js` 脚本

```
<script src="/assets/lib/jquery.js"></script>
<script src="/assets/lib/cropper/Cropper.js"></script>
<script src="/assets/lib/cropper/jquery-cropper.js"></script>
```

- 在卡片的 `layui-card-body` 主体区域中，定义如下的 `HTML` 结构

```
<!-- 第一行的图片裁剪和预览区域 -->
<div class="row1">
  <!-- 图片裁剪区域 -->
  <div class="cropper-box">
    <!-- 这个 img 标签很重要，将来会把它初始化为裁剪区域 -->
    <img id="image" src="/assets/images/sample.jpg" />
  </div>
  <!-- 图片的预览区域 -->
  <div class="preview-box">
    <div>
      <!-- 宽高为 100px 的预览区域 -->
      <div class="img-preview w100"></div>
      <p class="size">100 x 100</p>
    </div>
    <div>
      <!-- 宽高为 50px 的预览区域 -->
      <div class="img-preview w50"></div>
      <p class="size">50 x 50</p>
    </div>
  </div>
</div
<!-- 第二行的按钮区域 -->
<div class="row2">
  <button type="button" class="layui-btn">上传</button>
  <button type="button" class="layui-btn layui-btn-danger">确定</button>
</div>
```

- 美化的样式

```
/* 设置卡片主体区域的宽度 */
.layui-card-body {
  width: 500px;
```

```
}

/* 设置按钮行的样式 */
.row2 {
  display: flex;
  justify-content: flex-end;
  margin-top: 20px;
}

/* 设置裁剪区域的样式 */
.cropper-box {
  width: 350px;
  height: 350px;
  background-color: cyan;
  overflow: hidden;
}

/* 设置第一个预览区域的样式 */
.w100 {
  width: 100px;
  height: 100px;
  background-color: gray;
}

/* 设置第二个预览区域的样式 */
.w50 {
  width: 50px;
  height: 50px;
  background-color: gray;
  margin-top: 50px;
}

/* 设置预览区域下方文本的样式 */
.size {
  font-size: 12px;
  color: gray;
  text-align: center;
}

/* 设置图片行的样式 */
.row1 {
  display: flex;
}

/* 设置 preview-box 区域的的样式 */
.preview-box {
  display: flex;
  flex-direction: column;
  flex: 1;
  align-items: center;
}

/* 设置 img-preview 区域的样式 */
.img-preview {
  overflow: hidden;
  border-radius: 50%;
}
```

- 实现基本裁剪效果
  - 获取裁剪区域的 DOM 元素
  - 配置选项: 纵横比、指定预览区域
  - 创建裁剪区域

```
// 1.1 获取裁剪区域的 DOM 元素
var $image = $('#image')
// 1.2 配置选项
const options = {
  // 纵横比
  aspectRatio: 1,
  // 指定预览区域
  preview: '.img-preview'
}

// 1.3 创建裁剪区域
$image.cropper(options)
```

## 4.2.2 点击弹出文件选择框

- 默认的文件选择框样式比较丑，所以我们定义这个结构，让其隐藏，给文件选择框指定可以上传的文件类型

```
<input type="file" id="file" accept="image/png,image/jpeg" />
```

- 下面定义一个按钮，文本是 上传，一旦用户点击按钮，我们手动触发 文件选择框的点击事件

```
$('#btnChooseImage').on('click', function() {
  $('#file').click()
})
```

## 4.2.3 更换裁剪区域的图片

- 给文件选择框绑定 change 事件
- 用户选择了文件就会触发这个事件，通过 `e.target.files` 获取用户选择文件列表
- 通过索引0拿到用户选择的文件
- 将文件转化为路径
- 利用 `$image` 重新初始化裁剪区域

```
// 为文件选择框绑定 change 事件
$('#file').on('change', function(e) {
  // 获取用户选择的文件
  var filelist = e.target.files
  if (filelist.length === 0) {
    return layer.msg('请选择照片! ')
  }

  // 1. 拿到用户选择的文件
  var file = e.target.files[0]
  // 2. 将文件，转化为路径
  var imgURL = URL.createObjectURL(file)
  // 3. 重新初始化裁剪区域
  $image
```

```
    .cropper('destroy') // 销毁旧的裁剪区域
    .attr('src', imgURL) // 重新设置图片路径
    .cropper(options) // 重新初始化裁剪区域
})
```

# 4.3 将裁剪后的头像上传到服务器

- 为确定按钮，绑定点击事件
- 要拿到用户裁剪之后的头像
    - 创建一个 Canvas 画布，将 Canvas 画布上的内容，转化为 `base64` 格式的字符串

```
var dataURL = $image
      .cropper('getCroppedCanvas', {
        // 创建一个 Canvas 画布
        width: 100,
        height: 100
      })
      .toDataURL('image/png')
```

- 调用接口，把头像上传到服务器

```
$.ajax({
      method: 'POST',
      url: '/my/update/avatar',
      data: {
        avatar: dataURL
      },
      success: function(res) {
        if (res.status !== 0) {
          return layer.msg('更换头像失败！')
        }
        layer.msg('更换头像成功！')
        window.parent.getUserInfo()
      }
    })
```

## 4.3.1 了解 `base64` 格式的图片

`base64` 格式你会发现是一段字符串，其实 `base64` 格式的图片，就是利用一段字符串来描述这张图片

**好处：**能够避免一些额外的图片请求

**缺点：**体积会比原来图片要大 30% 左右

**使用场景：**不适用大图片，一些小图片比较适合使用

# 4.4 设置头部区域的快捷方式

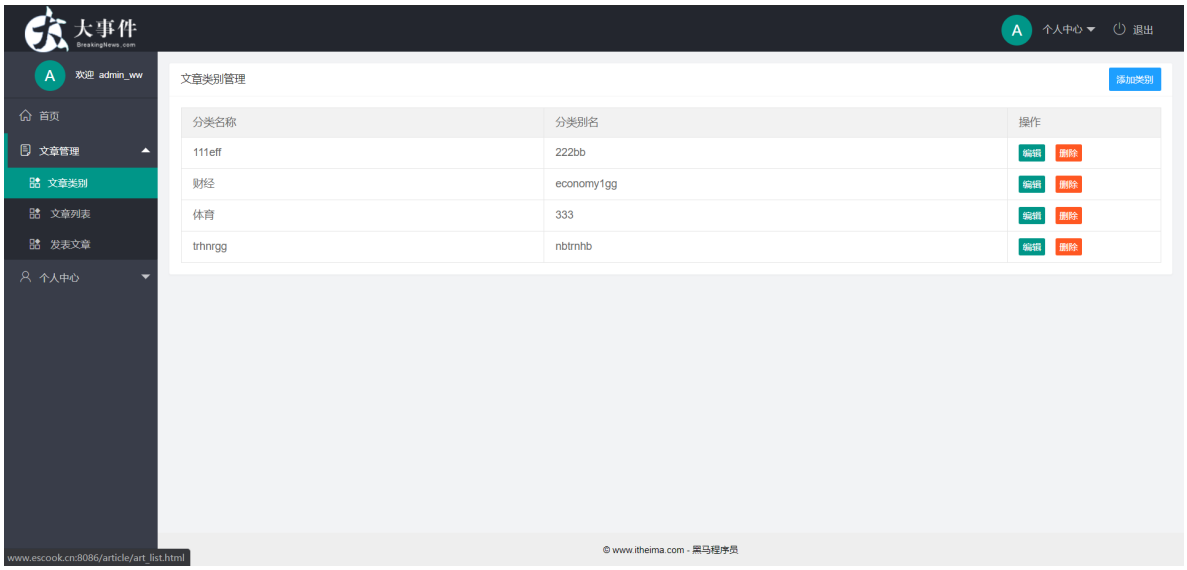
- 打开 `index.html`，修改头部 `个人中心` 下的三个快捷方式如下

```
<dl class="layui-nav-child">
  <dd><a href="/user/user_info.html" target="fm">基本资料</a></dd>
  <dd><a href="/user/user_avatar.html" target="fm">更换头像</a></dd>
  <dd><a href="/user/user_pwd.html" target="fm">重置密码</a></dd>
</dl>
```

## 4.5 本地代码推送到 `github`

- `git branch` 查看一下分支
- `git add .` 添加到暂存区
- `git commit -m` 进行提交到本地仓库
- `git push -u origin user` 推送到远程仓库的user分支
- `git checkout master` 切换到主分支
- `git merge user` 合并分支
- `git push` 推送到远程仓库的主分支

# 5. 文章分类




## 5.1 创建并显示文章分类页面

- 创建 `/article/art_cate.html` 页面，并初始化如下的`UI`结构

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <link rel="stylesheet" href="/assets/lib/layui/css/layui.css" />
    <link rel="stylesheet" href="/assets/css/article/art_cate.css" />
  </head>
  <body>
    <!-- 卡片区域 -->
    <div class="layui-card">
      <div class="layui-card-header">
        <span>文章类别管理</span>
```

```
        <button type="button" class="layui-btn layui-btn-normal layui-btn-sm">添加类别</button>
      </div>
      <div class="layui-card-body">
        <table class="layui-table">
          <colgroup>
            <col />
            <col />
            <col width="200" />
          </colgroup>
          <thead>
            <tr>
              <th>分类名称</th>
              <th>分类别名</th>
              <th>操作</th>
            </tr>
          </thead>
          <tbody>
            <tr>
              <td>贤心</td>
              <td>2016-11-29</td>
              <td>人生就像是一场修行</td>
            </tr>
            <tr>
              <td>许闲心</td>
              <td>2016-11-28</td>
              <td>于千万人之中遇见你所遇见的人，于千万年之中，时间的无涯的荒野里…</td>
            </tr>
          </tbody>
        </table>
      </div>
    </div>
  </body>
</html>
```

- 定义 `/assets/css/article/art_cate.css` 美化样式

```
html,
body {
  margin: 0;
  padding: 0;
}

body {
  padding: 15px;
  background-color: #f2f3f5;
}

.layui-card-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

- 修改 `index.html` 中对应的 `<a>` 链接

```
<a href="/article/art_cate.html" target="fm"><i class="layui-icon layui-icon-app"></i>文章类别</a>
```

## 5.2 快速绘制文章类别页面的基本结构

- 在 `layui-card-header` 里面添加右侧的按钮
- 给 `layui-card-header` 设置成flex 布局，让里面内容两侧对齐 `justify-content: space-between`

## 5.3 获取并使用模板引擎渲染表格的数据

- 利用模板引擎来进行渲染
- 在页面底部导入模板引擎

```
<script src="/assets/lib/template-web.js"></script>
```

- 定义模板

```
<!-- 表格数据的模板 -->
<script type="text/html" id="tpl-table">
  {{each data}}
  <tr>
    <td>{{$value.name}}</td>
    <td>{{$value.alias}}</td>
    <td>
      <button type="button" class="layui-btn layui-btn-xs">编辑</button>
      <button type="button" class="layui-btn layui-btn-danger layui-btn-xs">删除
</button>
    </td>
  </tr>
  {{/each}}
</script>
```

- 发起请求获取数据

```
initArtCateList()

// 获取文章分类的列表
function initArtCateList() {
  $.ajax({
    method: 'GET',
    url: '/my/article/cates',
    success: function(res) {
      var htmlStr = template('tpl-table', res)
      $('tbody').html(htmlStr)
    }
  })
}
```

## 5.4 使用`layer.open`实现弹出层效果(☆☆☆)

- 导入 `layer`

```
var layer = layui.layer
```

- 为按钮添加 `id` 属性

```
<button type="button" class="layui-btn layui-btn-normal layui-btn-sm"
id="btnAddCate">添加类别</button>
```

- 在按钮的点击事件中，通过 `layer.open()` 展示弹出层：

```
// 为添加类别按钮绑定点击事件
$('#btnAddCate').on('click', function() {
  layer.open({
    type: 1,
    area: ['500px', '250px'],
    title: '添加文章分类',
    content: 'ABC'
  })
})
```

## 5.5 在弹出层中渲染`form`表单结构

- 利用模板引擎的思路，在页面中定义如下的 `script` 标签

```
<script type="text/html" id="dialog-add">
  <form class="layui-form" id="form-add">
    <div class="layui-form-item">
      <label class="layui-form-label">分类名称</label>
      <div class="layui-input-block">
        <input type="text" name="name" required  lay-verify="required"
placeholder="请输入分类名称" autocomplete="off" class="layui-input">
      </div>
    </div>
    <div class="layui-form-item">
      <label class="layui-form-label">分类别名</label>
      <div class="layui-input-block">
        <input type="text" name="alias" required  lay-verify="required"
placeholder="请输入分类别名" autocomplete="off" class="layui-input">
      </div>
    </div>
    <div class="layui-form-item">
      <div class="layui-input-block">
        <button class="layui-btn" lay-submit lay-filter="formDemo">确认添加
</button>
        <button type="reset" class="layui-btn layui-btn-primary">重置</button>
      </div>
    </div>
  </form>
</script>
```

- 通过 `content` 属性指定内容

```
layer.open({
  type: 1,
  area: ['500px', '250px'],
  title: '添加文章分类',
  content: $('#dialog-add').html()
})
```

## 5.6 实现添加文章分类的功能

- 发起Ajax请求，注意，我们这个按钮不是写死的，是弹框出来的时候动态生成的，所以我们通过事件委派方式给表单绑定 `submit` 事件

```
// 通过代理的形式，为 form-add 表单绑定 submit 事件
$('body').on('submit', '#form-add', function(e) {
  e.preventDefault()
  $.ajax({
    method: 'POST',
    url: '/my/article/addcates',
    data: $(this).serialize(),
    success: function(res) {
      if (res.status !== 0) {
        return layer.msg('新增分类失败！')
      }
      initArtCateList()
      layer.msg('新增分类成功！')
      // 根据索引，关闭对应的弹出层
      layer.close(indexAdd)
    }
  })
})
```

- 预先保存弹出层的索引，方便进行关闭

```
// 为添加类别按钮绑定点击事件
var indexAdd = null
$('#btnAddCate').on('click', function() {
  indexAdd = layer.open({
    type: 1,
    area: ['500px', '250px'],
    title: '添加文章分类',
    content: $('#dialog-add').html()
  })
})
```