前端开发

```
目的：开发一个平台
    - 前端开发：HTML CSS JavaScript
    - 接收请求并处理
    - Mysql数据库：存储数据

快速上手：
    基于Flask Web框架快速搭建一个网站
深入学习：
    基于Django框架
```

1.快速开发网站

python 安装 Flask web 框架

```
pip install flask
```

> 创建Flask的python目录

```
[root@hecs-33592 ~]# mkdir -p /root/python/FlaskWeb
[root@hecs-33592 ~]# cd /root/python/FlaskWeb
[root@hecs-33592 FlaskWeb]# pwd
/root/python/FlaskWeb
1234
```

创建一个名为 `web.py` 的python文件

```
from flask import Flask

app = Flask(__name__)

# 创建了网址 /show/info 和 函数index 的对应关系
# 以后用户在浏览器上访问 /show/info, 网站自动执行
@app.route("/show/info")
def index():
    return "中国联通"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5100, debug=False)
123456789101112
```

运行

```
[root@hecs-33592 ~]# /usr/bin/python3 /root/python/FlaskWeb/web.py
1
```

```
[root@hecs-33592 ~]# /usr/bin/python3 /root/python/FlaskWeb/web.py
 * Serving Flask app 'web'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a
 production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5100
 * Running on http://192.168.0.123:5100
```


浏览器进行访问: http://[你的ip]:5100/show/info

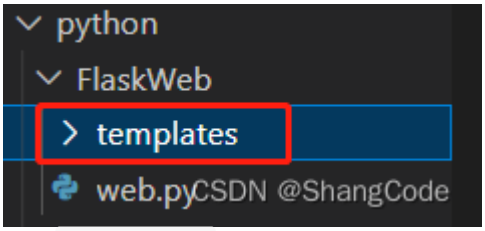
中国联通

> 这种 return 方式返回 HTML 内容的方式不方便进行管理,因此我们会引入 `templates` 模板

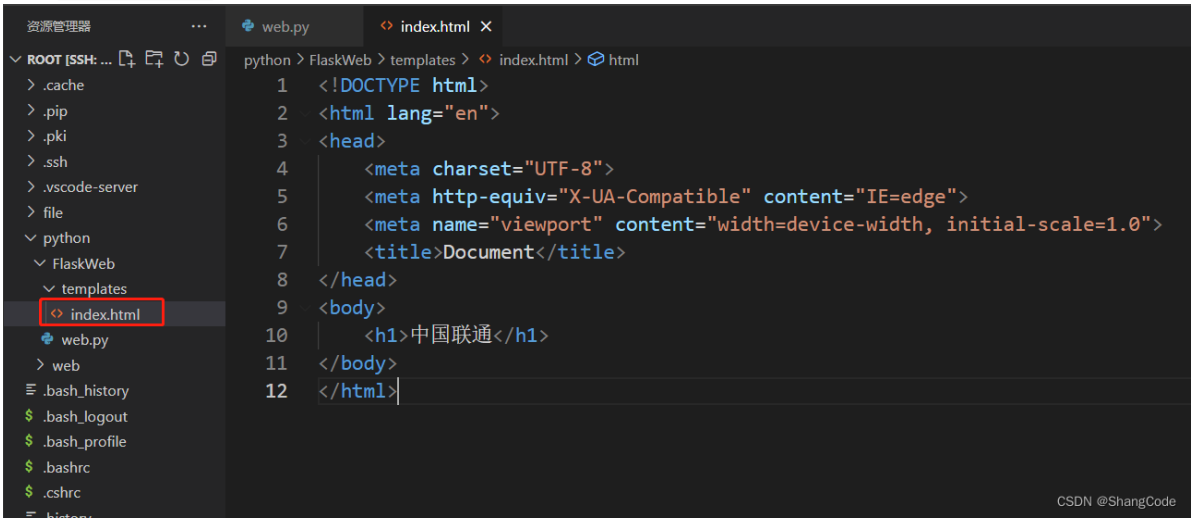
```python
from flask import Flask, render_template

app = Flask(__name__)

# 创建了网址 /show/info 和 函数index 的对应关系
# 以后用户在浏览器上访问 /show/info, 网站自动执行
@app.route("/show/info")
def index():
    # 默认去当前目录的 templates 文件夹中找
    return render_template("index.html")

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5100, debug=False)
12345678910111213
```

创建 `templates` 目录

```
mkdir /root/python/FlaskWeb/templates/
1
```

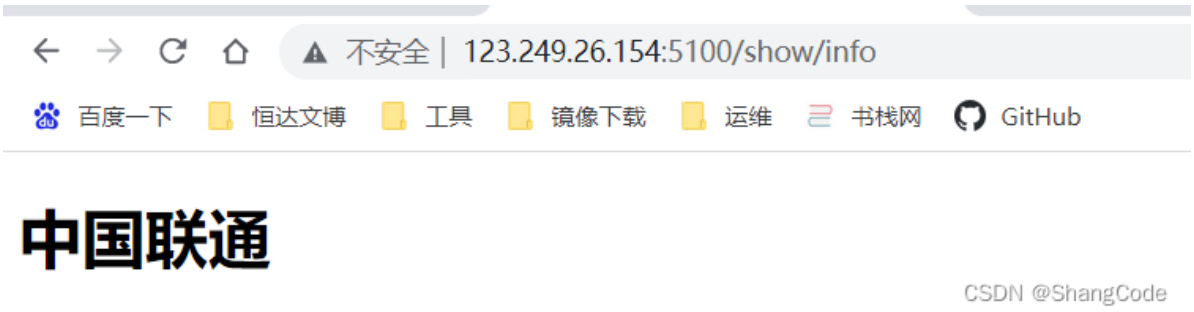


编写 `index.html` 文件




```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>中国联通</h1>
</body>
</html>
123456789101112
```

重新运行Flask,浏览器刷新访问



> 当然这个 `templates` 目录也可以自定义名称

```
# 例如目录名称为"xxx"
app = Flask(__name__, template_folder="xxx")
```

2.标签

2.1 编码

```
<meta charset="UTF-8">
```

2.2 title

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
123456
```

2.3 标题

```
<body>
    <h1>一级标题</h1>
    <h2>二级标题</h1>
    <h3>三级标题</h1>
    <h4>四级标题</h1>
    <h5>五级标题</h1>
</body>
1234567
```

2.4 div和span

```
<div>内容</div>

<span>asd</span>
123
```

- div: 占一整行(块级标签)
- span: 用多少占多少(行内标签/内联标签)
    - 两个 span 标签不在同一行,页面显示时会在同一行,中间以一个空格分隔
    - 两个 span 标签在同一行,页面显示时会在同一行,中间没有空格,连着

2.5 超链接

> 这里就很有意思了,你可以选择跳转自己网站的地址,或者跳转外部的网站

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>我的联通</title>
</head>
<body>
    <a href="/get/news">点击跳转自己的网站</a></br>
    <a href="http://www.baidu.com">点击跳转别人的网站百度</a>
</body>
</html>
```

然后需要修改 `web.py` 文件

```python
from flask import Flask, render_template

app = Flask(__name__)

# 创建了网址 /show/info 和 函数index 的对应关系
# 以后用户在浏览器上访问 /show/info, 网站自动执行
@app.route("/show/info")
def index():
    # 默认去当前目录的 templates 文件夹中找
    return render_template("index.html")

# 新添加如下配置
@app.route("/get/news")
def get_news():
    # 默认去当前目录的 templates 文件夹中找
    return render_template("get_news.html")

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5100, debug=False)
```

在 `templates` 目录下新添加一个 `get_news.html` 文件

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>我是内部链接</h1>
</body>
</html>
```

重新运行Flask,刷新页面

点击跳转自己的网站
点击跳转别人的网站百度

点击第一行后,跳转到如下页面

我是内部链接

点击点击第二行后,跳转到百度

> 自行脑补百度页面哈

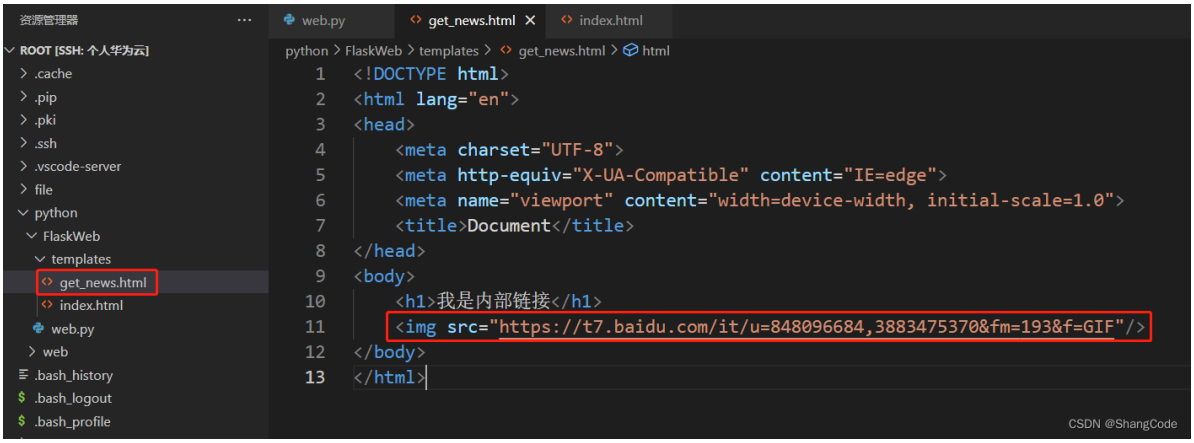
在新的 Tab 标签页打开链接

> 添加 target="_blank"

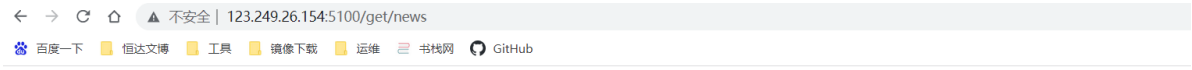
```
<body>
    <a href="https://www.mi.com/shop/buy/detail?product_id=16642" target="_blank">
        <img src="https://cdn.cnbj1.fds.api.mi-img.com/nr-pub/202210262033_ef39fca0e37395d07682124770fd3ad9.png" style="width: 150px;"/>
    </a>
</body>
```

2.6 图片

```
<body>
    <h1>我是内部链接</h1>
    <img src="https://t7.baidu.com/it/u=848096684,3883475370&fm=193&f=GIF"/>
</body>
1234
```



刷新浏览器

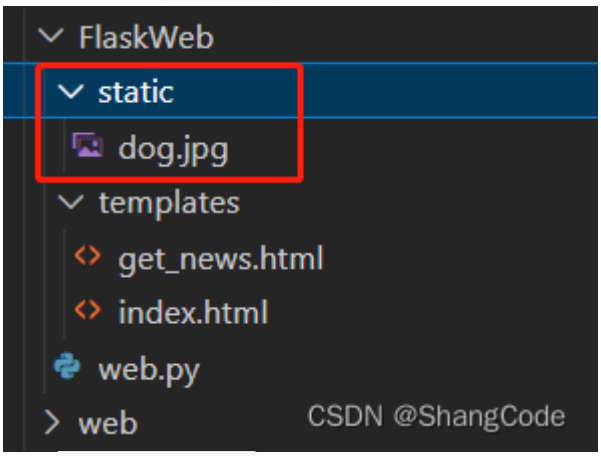


我是内部链接

> 尝试访问服务器本地图片

在 `/root/python/FlaskWeb/` 下新建目录 `static`

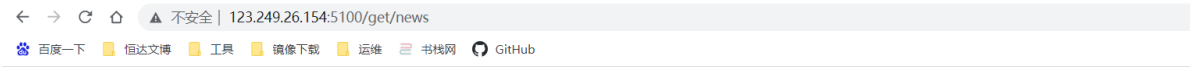
放入一张图片 `dog.jpg`



修改 `get_news.html`

```
<body>
    <h1>我是内部链接</h1>
    <img src="/static/dog.jpg"/>
</body>
```

刷新浏览器



我是内部链接

> 跟刚才一样

然后可以调整一下图片的高度与宽度

```
<body>
    <h1>我是内部链接</h1>
    <img style="height: 100px; width: 200px;" src="/static/dog.jpg"/>
</body>
```

我是内部链接

小结

```
- 块级标签
    - <h1></h1>
    - <div></div>

- 行内标签
    - <span></span>
    - <a></a>
    - <img />
```

标签的嵌套

> 实现: 点击图片,跳转至指定页面

修改 `web.py` ,增加 `get_product`

```python
from flask import Flask, render_template

app = Flask(__name__)

# 创建了网址 /show/info 和 函数index 的对应关系
# 以后用户在浏览器上访问 /show/info, 网站自动执行
@app.route("/show/info")
def index():
    # 默认去当前目录的 templates 文件夹中找
    return render_template("index.html")

@app.route("/get/news")
def get_news():
    return render_template("get_news.html")

@app.route("/get/product")
def get_product():
    return render_template("get_product.html")

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5100, debug=False)
123456789101112131415161718192021
```

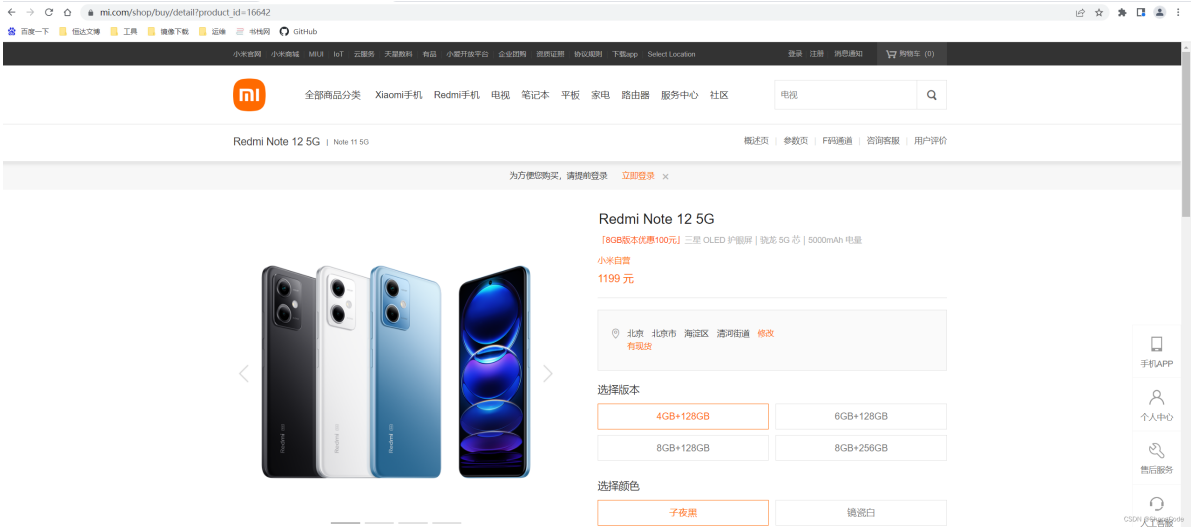
在 `templates` 下新增一个 `get_product.html`

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <a href="https://www.mi.com/shop/buy/detail?product_id=16642">
        <img src="https://cdn.cnbj1.fds.api.mi-img.com/nr-pub/202210262033_ef39fca0e37395d07682124770fd3ad9.png" style="width: 150px;"/>
    </a>
```

```
</body>
</html>
1234567891011121314
```

访问页面

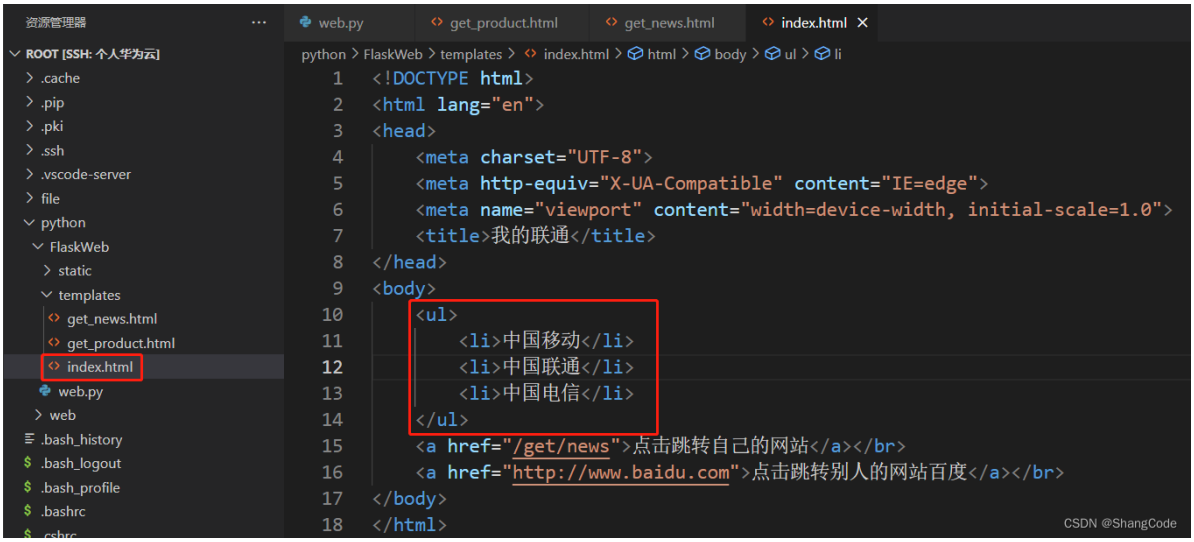
点击图片进行url跳转



2.7 列表

> 无序列表

```
<ul>
    <li>中国移动</li>
    <li>中国联通</li>
    <li>中国电信</li>
</ul>
12345
```



- 中国移动
- 中国联通
- 中国电信

点击跳转自己的网站
点击跳转别人的网站百度

> 有序列表

```
<ol>
    <li>中国移动</li>
    <li>中国联通</li>
    <li>中国电信</li>
</ol>
12345
```

1. 中国移动
2. 中国联通
3. 中国电信

点击跳转自己的网站
点击跳转别人的网站百度

2.8 表格

修改`web.py`新增一个访问路径

```
@app.route("/get/table")
def get_table():
    return render_template("get_table.html")
123
```

在`templates`页面下新建`get_table.html`文件

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <table>
        <thead>
            <tr><th>ID</th><th>姓名</th><th>年龄</th></tr>
        </thead>
        <tbody>
            <tr><td>10</td><td>张三</td><td>20</td></tr>
            <tr><td>11</td><td>李四</td><td>20</td></tr>
            <tr><td>12</td><td>王五</td><td>20</td></tr>
            <tr><td>13</td><td>赵六</td><td>20</td></tr>
        </tbody>
    </table>
</body>
</html>
123456789101112131415161718192021 22
```

重新运行并访问页面

> 为表格增加边框

```
<table border="1">
1
```

2.9 input系列

```
<!-- 文本与密码 -->
<input type="text" />
<input type="password" />

<!-- 选择文件 -->
<input type="file" />

<!-- 单选框 -->
<input type="radio" name="n1" />男
<input type="radio" name="n1" />女

<!-- 复选框 -->
<input type="checkbox" />唱
<input type="checkbox" />跳
<input type="checkbox" />Rap
<input type="checkbox" />篮球

<!-- 按钮 -->
<input type="button" value="提交"/>    普通按钮
<input type="submit" value="提交"/>    提交表单
12345678910111213141516171819 20
```

2.10 下拉框

```
<select>
    <option>北京</option>
    <option>上海</option>
    <option>深圳</option>
</select>
12345
```

2.11 多行文本

```
<textarea></textarea>
1
```

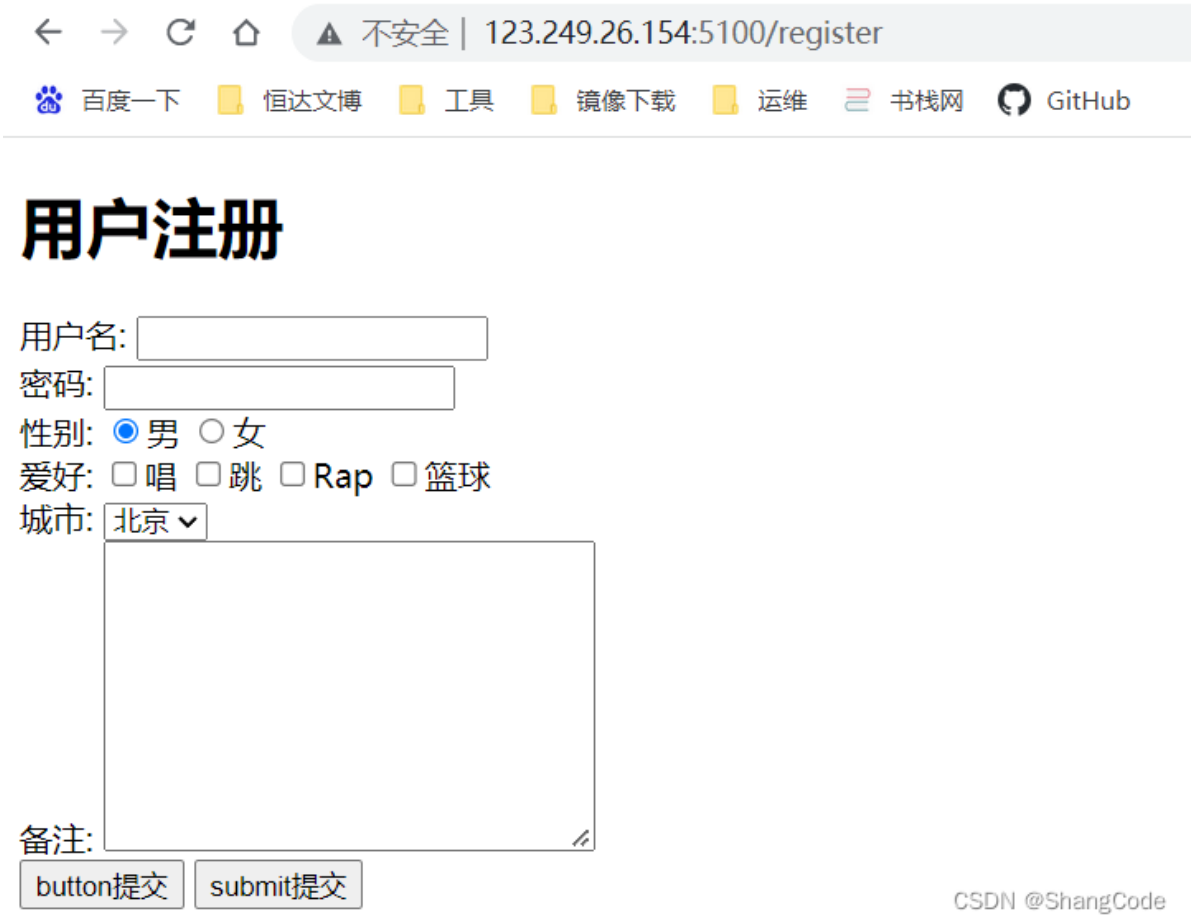
用户注册

修改`web.py`

```
@app.route("/register")
def register():
    return render_template("login.html")
123
```

在`templates`下新建`register.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>用户注册</h1>
    <div>
        用户名: <input type="text" />
    </div>
    <div>
        密码: <input type="password" />
    </div>
    <div>
        性别: <input type="radio" name="sex"/>男 <input type="radio" name="sex"/>女
    </div>
    <div>
        爱好:
        <input type="checkbox">唱
        <input type="checkbox">跳
        <input type="checkbox">Rap
        <input type="checkbox">篮球
    </div>
    <div>
        城市:
        <select>
            <option>北京</option>
            <option>上海</option>
            <option>深圳</option>
        </select>
    </div>
    <div>
        备注: <textarea cols="30" rows="10"></textarea>
    </div>
    <div>
        <input type="button" value="button提交">
        <input type="submit" value="submit提交">
    </div>
</body>
</html>
12345678910111213141516171819202122232425262728293031323334353637383940414243
```



> 顺便说一下 GET 方法与 POST 方法的区别
> GET: 可通过 `URL/表单` 提交
> POST: 只能通过 `表单` 提交,提交数据不在URL而是在 `请求体` 中

案例: 用户注册

新建项目
在 `/root/python` 下新建目录:

- `account`
- `template`

在 `account` 下新建 `app.py` 文件

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/register')
def register():
    return render_template('login.html')

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5200, debug=False)
12345678910
```

在 `templates` 下新建 `register.html` 文件

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>用户注册</h1>
</body>
</html>
123456789101112
```

运行,浏览器进行访问

表单可以提交的前提条件:

- 提交方式: method="get"
- 提交地址: action="/xxx/xxx/xxx"
- 在 `form` 标签里面必须有一个 `submit` 标签
- 每个标签有 `name` 属性

接下来尝试接收用户提交的表单数据

GET 方式

修改 `app.py` ,导入 `request` 方法,使用 `/do/register` 接收用户数据并展示

```
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/register', methods=['GET'])
def register():
    return render_template('login.html')

@app.route("/do/register", methods=['GET'])
def do_register():
    get_info = request.args
    return get_info

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5200, debug=True)
123456789101112131415
```

修改 `templates` 下的 `register.html`

> 点击注册后跳转至路由 `/do/register`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>用户注册</h1>
    <form action="/do/register" method="get">
        <div>
            用户名: <input type="text" name="username">
        </div>
        <div>
            密码: <input type="password" name="passwd">
        </div>

        <input type="submit" value="提交">
    </form>
</body>
</html>
12345678910111213141516171819202122
```


```
{
  "passwd": "123456",
  "username": "zhangsan"
}
```


POST 方式

修改 `app.py`

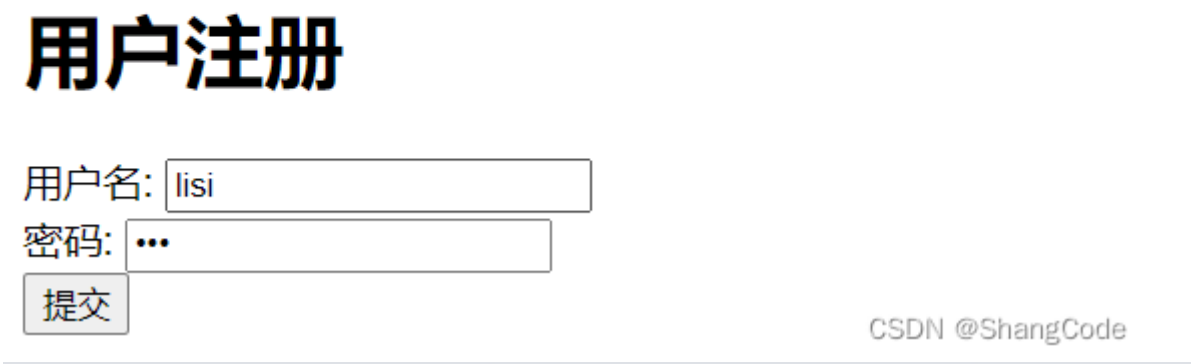
```
@app.route("/post/register", methods=['POST'])
def post_register():
    get_info = request.form
    return get_info
1234
```

修改`register.html`

```
1   <form action="/post/register" method="post">
```

```
1   <body>
2       <h1>用户注册</h1>
3       <form action="/post/register" method="post">
4           <div>
5               用户名: <input type="text" name="username">
6           </div>
7           <div>
8               密码: <input type="password" name="passwd">
9           </div>
10          <input type="submit" value="提交">
11      </form>
12  </body>
13  123456789101112
```

浏览器访问

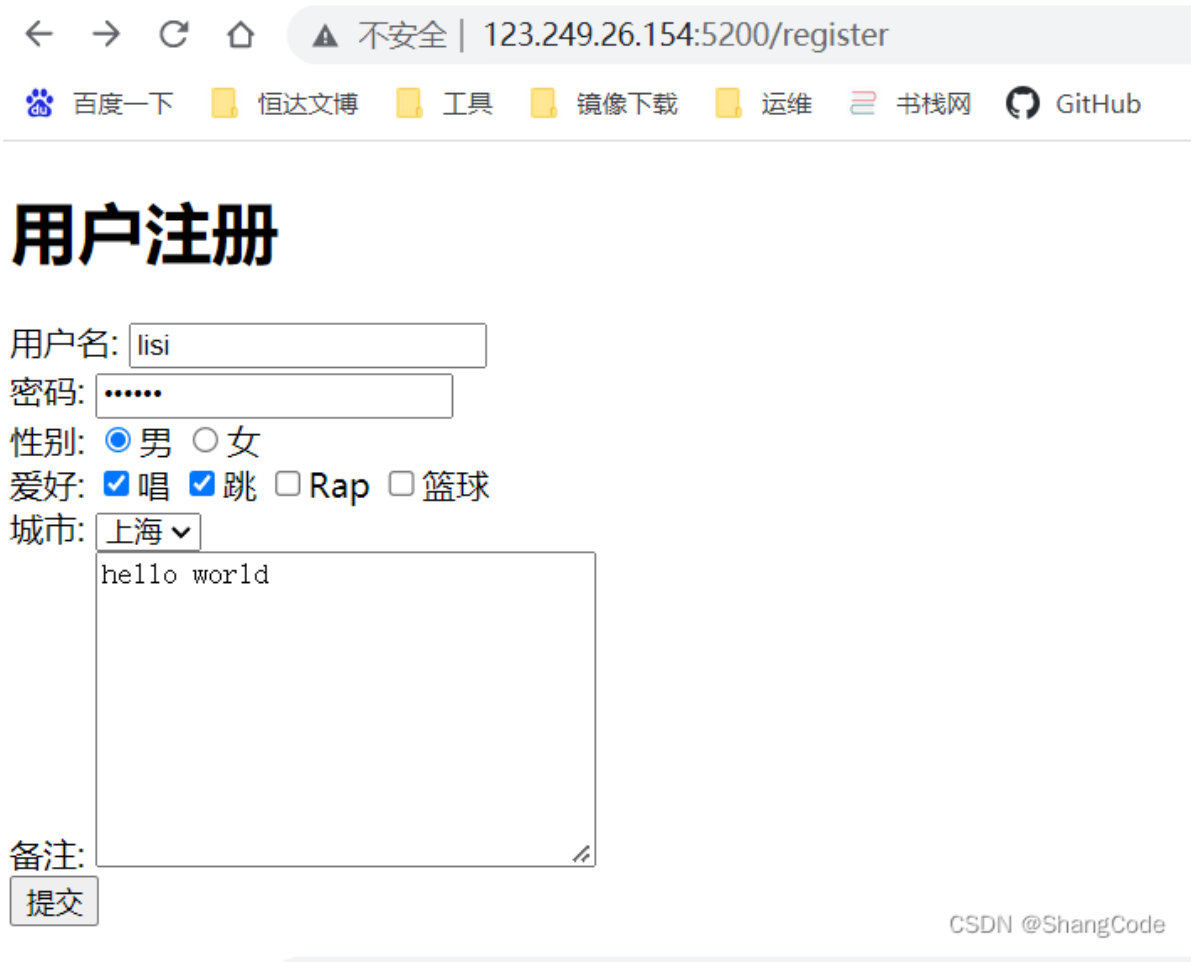


可以发现,跟上面的`GET`方法不同的是, 提交后跳转的页面的`URL`后并没有我们提交的参数,而是提交到了后台

表单数据提交优化

修改`register.html`

> 添加`name`与`value`属性



> 在控制台输出数据

修改 `app.py`

```python
@app.route("/post/register", methods=['POST'])
def post_register():
    get_info = request.form

    username = request.form.get("username")
    passwd = request.form.get("passwd")
    sex = request.form.get("sex")
    hobby_list = request.form.getlist("hobby")
    city = request.form.get("city")
    more = request.form.getlist("textarea")

    print(username, passwd, sex, hobby_list, city, more)

    return get_info
```

```
15    1234567891011121314
```

用户注册

用户名: lisi
密码: ••••••
性别: 男 女
爱好: 唱 跳 Rap 篮球
城市: 北京
备注: hello
提交


```
问题  输出  调试控制台  终端  端口 3  JUPYTER

    print(username, passwd, sex, hobby_list, city_list, more)
NameError: name 'city_list' is not defined
106.47.30.60 - - [30/Nov/2022 16:32:39] "GET /post/register?__debugger__=yes&cmd=resource&f=style.css HTTP/1.1" 200 -
106.47.30.60 - - [30/Nov/2022 16:32:39] "GET /post/register?__debugger__=yes&cmd=resource&f=debugger.js HTTP/1.1" 200 -
106.47.30.60 - - [30/Nov/2022 16:32:39] "GET /post/register?__debugger__=yes&cmd=resource&f=console.png HTTP/1.1" 200 -
106.47.30.60 - - [30/Nov/2022 16:32:39] "GET /post/register?__debugger__=yes&cmd=resource&f=console.png HTTP/1.1" 304 -
 * Detected change in '/root/python/account/app.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 217-362-781
lisi 123456 man ['sing', 'jump'] beijing ['hello']
106.47.30.60 - - [30/Nov/2022 16:32:48] "POST /post/register HTTP/1.1" 200 -
```


> 整合 `GET` 与 `POST` 方法

```python
@app.route("/do/register", methods=['GET'])
def do_register():
    get_info = request.args
    return get_info

@app.route("/post/register", methods=['POST'])
def post_register():
    get_info = request.form

    username = request.form.get("username")
    passwd = request.form.get("passwd")
    sex = request.form.get("sex")
    hobby_list = request.form.getlist("hobby")
    city = request.form.get("city")
    more = request.form.getlist("textarea")

    print(username, passwd, sex, hobby_list, city, more)

    return get_info
```


将上面图片中的内容整合

```python
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == "GET":
        return render_template('register.html')
    else:
        username = request.form.get("username")
        passwd = request.form.get("passwd")
        sex = request.form.get("sex")
        hobby_list = request.form.getlist("hobby")
        city = request.form.get("city")
        more = request.form.getlist("textarea")

        print(username, passwd, sex, hobby_list, city, more)

        get_info = request.args
        return get_info
```



```python
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == "GET":
        return render_template('login.html')
    else:
        username = request.form.get("username")
        passwd = request.form.get("passwd")
        sex = request.form.get("sex")
        hobby_list = request.form.getlist("hobby")
        city = request.form.get("city")
        more = request.form.getlist("textarea")

```

```
        print(username, passwd, sex, hobby_list, city, more)

        get_info = request.args
        return get_info
```

3.CSS样式

css , 专门用来美化标签

3.1 快速上手

```
<img src="..." stype="height: 100px">
```

3.2 CSS应用方式

1. 在标签上

```
<img src="..." stype="height: 100px">
```

2. 在 head 标签的 style 上

```
...
<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <style>
        .c1 {
            color: red;
        }
    </style>

</head>
<body>
    <h1 class="c1">用户注册</h1>
    ...
```

3. 写到文件中

- common.css

```
.c1 {
    color: red;
}
 .c2{
    color:blue;
}

```

调用 `common.css`

```
...
<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <link rel="stylesheet" href="common.css" />

</head>
<body>
    <h1 class="c1">用户注册</h1>
    ...
1234567891011
```

3.3 选择器

1. ID选择器

id 唯一

```
#c1 {
    color: red;
}

<div id='c1'></div>
```

2. 类选择器

```
.c1 {
    color: red;
}

<div class='c1'></div>
```

3. 标签选择器

```
div{
    color: red;
}

<div>xxx</div>
```

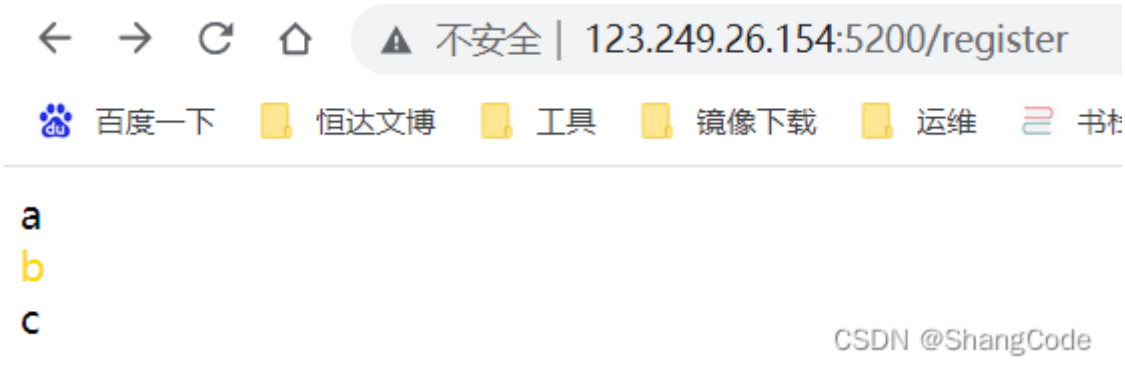
4. 属性选择器

> 下面的例子中,所有的 `text` 类型的 `input` 都会生效

```
<head>
    <title>Document</title>
    <link rel="stylesheet" href="/static/commons.css">
    <style>
        input[type="text"]{
        border: 1px solid red;
    }
    </style>
</head>
```

> 还有另一种方式,看下面的例子

```
<style>
    .v1[xx="456"]{
        color: gold;    <!-- 橙色 -->
    }
</style>

...

<body>
...
    <div class="v1" xx="123">a</div>
    <div class="v1" xx="456">b</div>
    <div class="v1" xx="789">c</div>
...
</body>
12345678910111213141 5
```



5. 后代选择器

> 这个选择器很有意思,你可以指定标签让它下面对应的标签全部生效,也可以指定标签让他下面的n级标签生效,具体看例子

```
<style>
    .zz h2{
        color:chartreuse;
    }
</style>
</head>

<body>

    <div class="zz" >
        <div>
            <h2>我是div里面的h2</h2>
        </div>
        <h2>我是div外面的h2</h2>
...
123456789101112131415
```

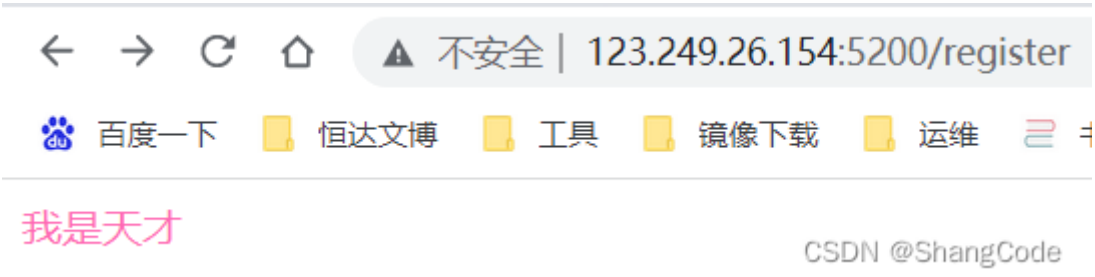
> 如果只想让第一层的 `h1` 生效,可以添加 `>` 号

```
<style>
    .zz > h2{
        color:chartreuse;
    }
</style>
12345
```

关于样式的覆盖问题

当一个标签引用了多个 css 样式时,可能会遇到样式属性重复的问题

```
<style>
    .c2 {
        color: darkgoldenrod;
    }

    .c3 {
        color:hotpink;
    }
</style>

<body>
    <div class="c2 c3">我是天才</div>
</body>
12345678910111213
```



> 观察到,c3生效,而c2没有生效,这是因为c3在c2的下面,会将上面的c2属性覆盖掉
> 如果不想让上面的被覆盖掉怎么办呢?
> 可以在对应的属性后面添加 `!important`

```
<style>
    .c2 {
        color: darkgoldenrod !important;
    }

    .c3 {
        color:hotpink;
    }
</style>
123456789
```

3.4 样式

1. 高度和宽度

```
.c4 {
        height: 300px;
        width: 500px;
    }
```

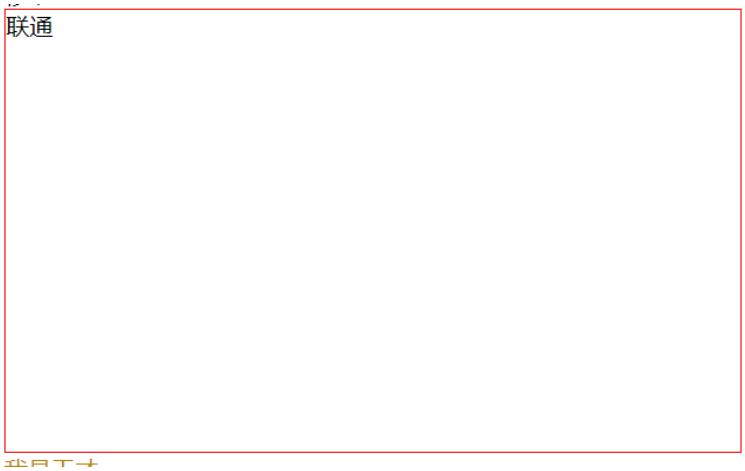
注意事项:

- 支持百分比
- 行内标签: 默认无效
- 块级标签: 默认有效(右边的剩余空白区域也会被占用)

2. 块级和行内标签

> `display:inline-block` 使行内标签对 **height** 和 **width** 生效

```
<style>
.c4 {
        display: inline-block;
        height: 300px;
        width: 500px;
        border: 1px solid red;
    }
</style>

...

<body>
    <span class="c4">联通</span>
</body>

```



> 块级与行内标签的转换

```
# 转化为 行内标签
<div style="display: inline;">移动</div>
# 转化为 块级标签
<span style="display: block;">联通</span>

```

注意:

- 块级标签 + 块级&行内标签

3. 字体和对齐方式

> 设置字体颜色/大小/粗细/字体样式

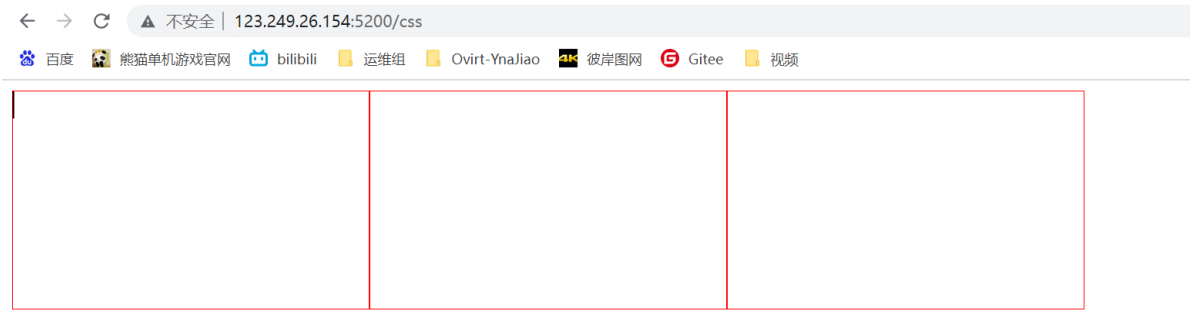
```
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        .c1 {
            color: #00FF7F;                 /* 字体颜色 */
            font-size: 20px;                /* 字体大小 */
            font-weight: 600;               /* 字体粗细 */
            font-family: Microsoft Yahei;   /* 字体样式 */
            text-align: center;             /* 水平方向居中 */
            line-height: 50px;              /* 垂直方向居中 */
            border: 1px solid red;          /* 边框 */
        }
    </style>
</head>
```

4. 浮动

> 如果在块级标签中，加入了 `float` 属性，那么这个块级标签奖不会再占用一整行，而是自己有多大就占用多大

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        .item {
            float: left;
            width: 280px;
            height: 170px;
            border: 1px solid red;
        }

    </style>
</head>
<body>
    <div>
        <div class="item"></div>
        <div class="item"></div>
        <div class="item"></div>
    </div>
</body>
</html>
12345678910111213141516171819202122 23
```

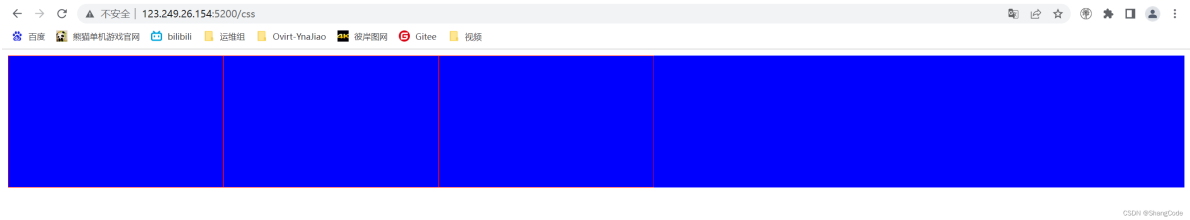


> 如果你让标签浮动起来之后，就会脱离文档流。
> 例如下面的例子中，我们给div的父标签赋予了一个蓝色的背景，但是你不会看到蓝色背景。因为他被浮动的div字标签挡住了。

```
<body>
    <div style="background-color: blue;">
        <div class="item"></div>
        <div class="item"></div>
        <div class="item"></div>
    </div>
</body>
```

> 解决办法: 在同级子标签的最下面添加 `style="clear: both;"`

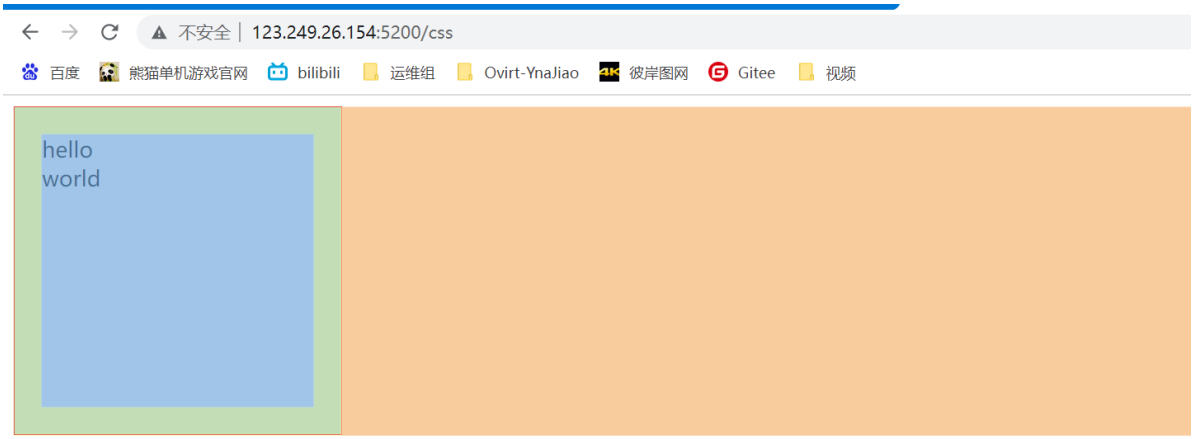
```
<body>
    <div style="background-color: blue;">
        <div class="item"></div>
        <div class="item"></div>
        <div class="item"></div>
        <div style="clear: both;"></div>
    </div>
</body>
```

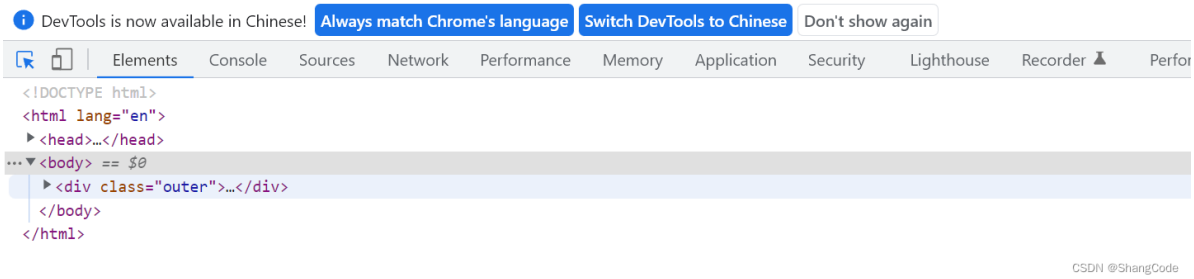


5. 内边距

> padding-top | padding-left | padding-right | padding-botton

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        .outer {
            border: 1px solid red;
            height: 200px;
            width: 200px;

            padding-top: 20px;
            padding-left: 20px;
            padding-right: 20px;
            padding-bottom: 20px;
        }

    </style>
</head>
<body>
    <div class="outer">
        <div>hello</div>
        <div>world</div>

    </div>
</body>
</html>
```



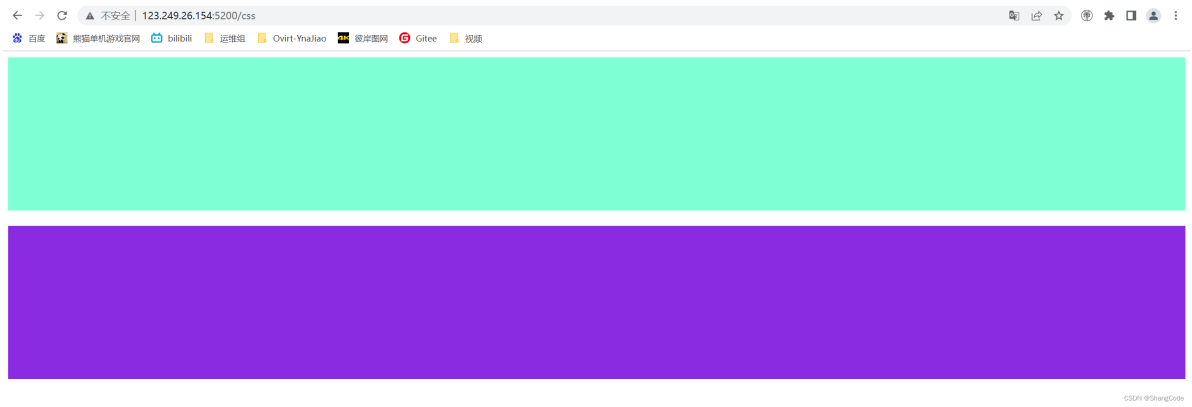


其实上面的四个上下左右的padding可以简写为 `padding: 20px 20px 20px 20px` ,顺序为上右下左(顺时针方向)

6. 外边距

> margin

```
<body>
    <div style="height: 200px; background-color: aquamarine;"></div>
    <div style="height: 200px; background-color:blueviolet; margin-top: 20px;">
</div>
</body>
1234
```



7. hover(伪类)

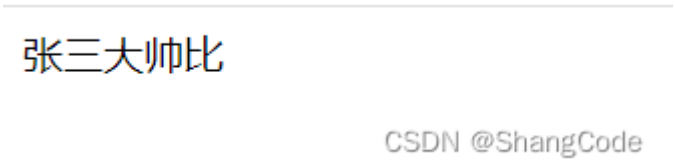
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
```

```
        .c1 {
            color:brown;
        }
        .c1:hover {
            color: green;
            font-size: 20px;
        }

        .c2 {
            width: 300px;
            height: 300px;
            border: 3px solid red;
        }
        .c2:hover {
            border: 3px solid green;
        }

        .download {
            display: none;
        }

        .app:hover .download {
            display: block;
        }

    </style>
</head>
<body>
    <div class="c1">字体碰到鼠标变成绿色</div>
    <div class="c2">边框碰到鼠标变成绿色</div>
    <div class="app">
        <div>鼠标放我这里触发显示二维码</div>
        <div class="download">
            <img src="https://cdn.cnbj1.fds.api.mi-img.com/mi-mall/78c30d4f259ed43ab20e810a522a6249.png" alt="">
        </div>
    </div>
</body>
</html>
```

8. after (伪类)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        .c1:after {
            content: "大帅比"
        }
    </style>
</head>
<body>
    <div class="c1">张三</div>
```

```
</body>
</html>
```



> after一般像下面这样用,不用每次都写 `stype="clear: both;"`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        .clearfix:after {
            content: "";
            display: block;
            clear: both;
        }

        .item {
            float: left;
        }

    </style>
</head>
<body>
    <div class="clearfix">
        <div class="item">1</div>
        <div class="item">2</div>
        <div class="item">3</div>
    </div>
</body>
</html>
```

9. position

- fixed
- relative
- absolute

9.1 fixed

固定在窗口的某个位置。

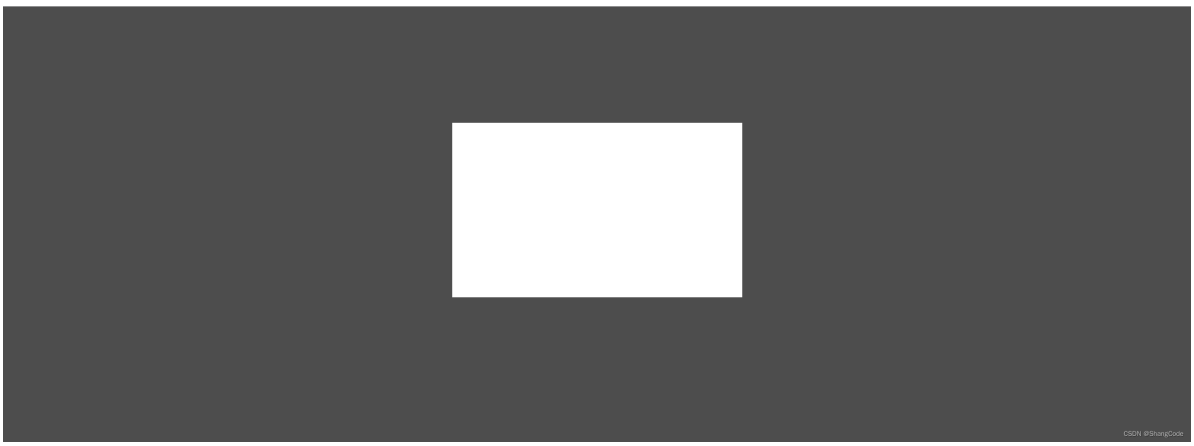
> 返回顶部

```css
.back {
    position: fixed;
    width: 60px;
    height: 60px;
    border: 1px solid red;

    right: 50px;
    bottom: 50px;
}
```

> 对话框

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>

        body {
            margin: 0;
        }

        .dialog {
            position: fixed;
            height: 300px;
            width: 500px;
            background-color: white;

            /* 居中 */

            left: 0;
            right: 0;
            margin: 0 auto;

            top: 200px;

            z-index: 1000;  /* 防止对话框被mask遮住 */
        }

        .mask {
            background-color: black;
            position: fixed;
            left: 0;
            right: 0;
            top: 0;
            bottom: 0;
            opacity: 0.7;

            z-index: 999;   /* 防止对话框被mask遮住  z-index 越大 优先级越高*/
        }

    </style>
</head>
<body>
```

```
    <div style="height: 1000px;"></div>
    <!-- 如果css中不加 z-index 设置优先级的话 -->
    <!-- 那么下面的 dialog 如果在 mask 的上面,对话框就显示不出来了 -->
    <!-- 设置优先级可以解决此问题 -->


    <div class="dialog"></div>
    <div class="mask"></div>
</body>
</html>
```

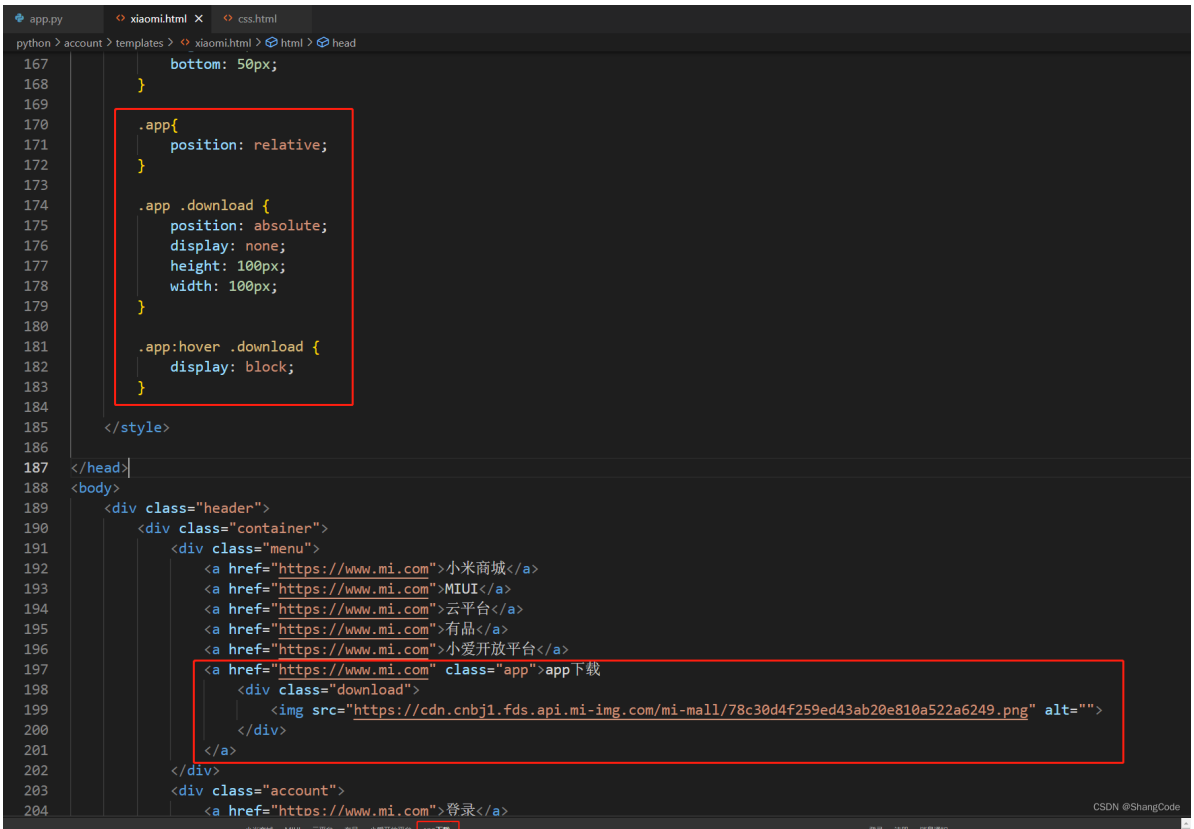


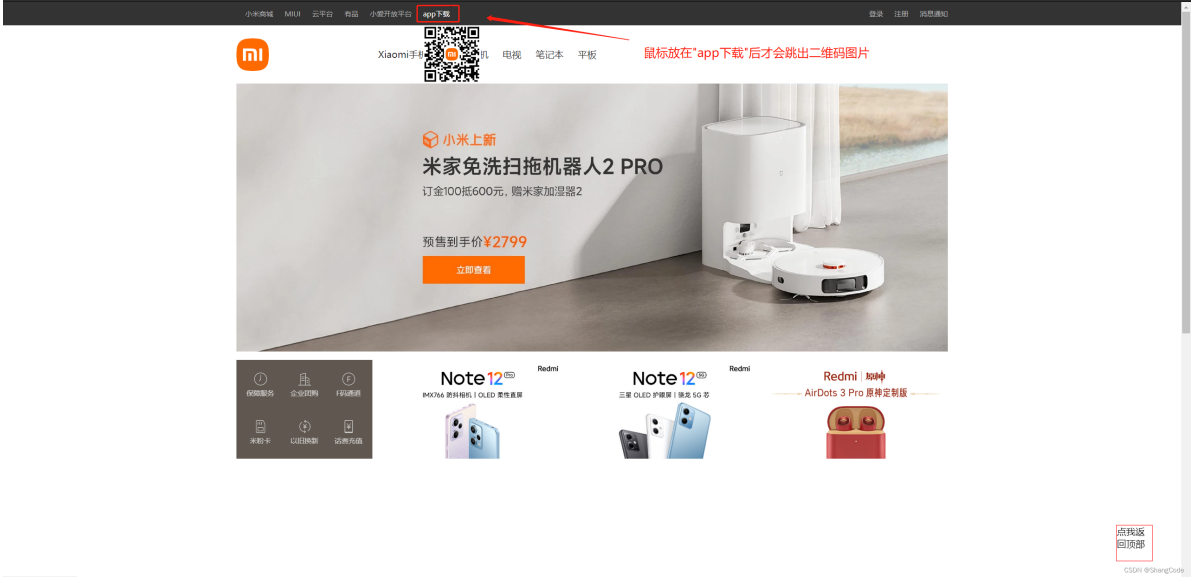
9.2 relative和absolute

> 在 小米商城 案例的基础上进行测试

```
...
        .app{
            position: relative;
        }

        .app .download {
            position: absolute;
            display: none;
            height: 100px;
            width: 100px;
        }

        .app:hover .download {
            display: block;
        }

    </style>

</head>
<body>
    <div class="header">
        <div class="container">
            <div class="menu">
                <a href="https://www.mi.com">小米商城</a>
                <a href="https://www.mi.com">MIUI</a>
                <a href="https://www.mi.com">云平台</a>
```

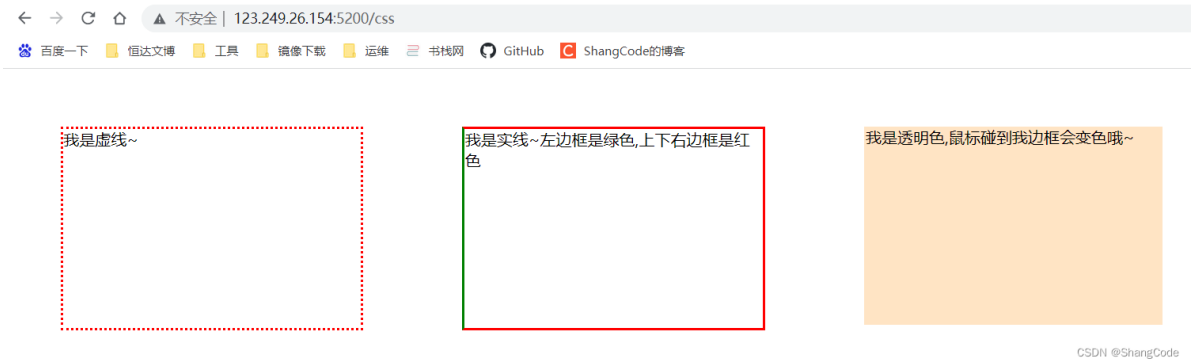
```
                <a href="https://www.mi.com">有品</a>
                <a href="https://www.mi.com">小爱开放平台</a>
                <a href="https://www.mi.com" class="app">app下载
                    <div class="download">
                        <img src="https://cdn.cnbj1.fds.api.mi-img.com/mi-
mall/78c30d4f259ed43ab20e810a522a6249.png" alt="">
                    </div>
                </a>
            </div>
            <div class="account">
                <a href="https://www.mi.com">登录</a>
                <a href="https://www.mi.com">注册</a>
                <a href="https://www.mi.com">消息通知</a>
            </div>'
            <div style="clear: both;"></div>
        </div>
    </div>
...
```





10. border

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>

        .left {
            float: left;
        }

        .c1 {
            height: 200px;
            width: 300px;
            border: 3px dotted red;
            margin: 50px;
        }

        .c2 {
            height: 200px;
            width: 300px;
            border: 3px solid red;
            border-left: 3px solid green;
            margin: 50px;
        }

        .c3 {
            height: 200px;
            width: 300px;
            margin: 50px;
            background-color: bisque;
            border-left: 2px solid transparent;  /* 透明色 */
        }

        .c3:hover {
            border-left: 2px solid rgb(35, 211, 19);
        }

    </style>
</head>
<body>
    <div class="c1 left">我是虚线~</div>
    <div class="c2 left">我是实线~左边框是绿色,上下右边框是红色</div>
    <div class="c3 left">我是透明色,鼠标碰到我边框会变色哦~</div>
    <div style="clear: both;"></div>
</body>
</html>
```



11. 背景色

> background-color: bisque;
> 无需多言☺

> 注意: 以上不是所有的CSS样式,这些是最常用的标签

总结

- body 标签，默认有一个边距，造成页面四边都有白色间隙，如何去除呢?

```
body{
    margin:0;
}
```

- 内容居中
  - 文本居中

    ```
    <div style="width:200px; text-align:center;">学习Django</div>
    ```

  - 区域居中，自己要有宽度 + margin -left:auto ; margin-right:auto

    ```
    .container{
        width:980px;
        margin:0 auto;
    }

    <div class="container">
        adsfaf
    </div>
    ```

- 父亲没有宽度或者高度，被孩子支撑起来
- 关于布局 不知道如何下手

4.案例: 小米商城

4.1 小米顶部

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>小米商城</title>
    <style>
        /* 去掉body的边距 */
        body {
            margin: 0;
        }

        .header {
            background-color: #333;
        }

        /* 让中间内容居中 */
        .container {
            width: 1226px;
            margin: 0 auto;     /* 上下为0, 左右为auto */
        }

        /* header class 下的标签 a 自动应用这个样式 */
        .header a {
            color: #b0b0b0;
            line-height: 40px;
            display: inline-block;
            font-size: 12px;
        }

        .header .menu {
            float: left;
            color: white;
        }

        .header .account {
            float: right;
            color: white;
        }
    </style>
</head>
<body>
    <div class="header">
        <div class="container">
            <div class="menu">
                <a>小米商城</a>
                <a>MIUI</a>
                <a>云平台</a>
                <a>有品</a>
                <a>小爱开放平台</a>
            </div>
```

```
            <div class="account">
                <a>登录</a>
                <a>注册</a>
                <a>消息通知</a>
            </div>'
            <div style="clear: both;"></div>
        </div>
    </div>
</body>
</html>
```

4.2 二级菜单

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>小米商城</title>
    <style>
        /* 去掉body的边距 */
        body {
            margin: 0;
        }

        .header {
            background-color: #333;
        }

        /* 让中间内容居中 */
        .container {
            width: 1226px;
            margin: 0 auto;     /* 上下为0, 左右为auto */
        }

        /* header class 下的标签 a 自动应用这个样式 */
        .header a {
            color: #b0b0b0;
            line-height: 40px;
            display: inline-block;
            font-size: 12px;
        }

        .header .menu {
            float: left;
            color: white;
        }

        .header a {
            text-decoration: none;
        }

        .header a:hover {
            color: white;
```

```
        }

        .header .account {
            float: right;
            color: white;
        }

        .sub-header {
            height: 100px;
        }

        .sub-header .hw {
            width: 234px;
            height: 100px;
        }

        .sub-header .logo {
            float: left;
        }

        /* a标签是行内标签,默认不支持设置高度与边距 因此设置padding是不起作用的,因此可以加上inline-block */
        .sub-header .logo a {
            padding-top: 22px;
            padding-bottom: 22px;
            display: inline-block;
        }

        /* 设置logo的图片像素大小 */
        .sub-header .logo img {
            height: 56px;
            width: 56px;
        }

        .sub-header .menu {
            width: 400px;
            float:left;
            line-height: 100px;     /* 与行高度保持一致 */
        }

        .sub-header .menu a {
            text-decoration: none;  /* 去掉 a 标签的下划线 */
            color: #333;
            font-size: 16px;
            padding: 0 10px;        /* 设置字体的左右外边距 */
            display: inline-block;
        }

        /* 鼠标放到字体时,使字体变红 */
        .sub-header .menu a:hover {
            color: #ff6700;
        }

        .sub-header .search {
            float: right;
        }
```

```
96
97      </style>
98
99  </head>
100 <body>
101     <div class="header">
102         <div class="container">
103             <div class="menu">
104                 <a href="https://www.mi.com">小米商城</a>
105                 <a href="https://www.mi.com">MIUI</a>
106                 <a href="https://www.mi.com">云平台</a>
107                 <a href="https://www.mi.com">有品</a>
108                 <a href="https://www.mi.com">小爱开放平台</a>
109             </div>
110             <div class="account">
111                 <a href="https://www.mi.com">登录</a>
112                 <a href="https://www.mi.com">注册</a>
113                 <a href="https://www.mi.com">消息通知</a>
114             </div>'
115             <div style="clear: both;"></div>
116         </div>
117     </div>
118     <div class="sub-header">
119         <div class="container">
120             <div class="hw logo">
121                 <a href="https://www.mi.com">
122                     <img src="https://s02.mifile.cn/assets/static/image/logo-
    mi2.png" alt="小米官网">
123                 </a>
124             </div>
125             <div class="hw menu">
126                 <a href="https://www.mi.com">Xiaomi手机</a>
127                 <a href="https://www.mi.com">Redmi手机</a>
128                 <a href="https://www.mi.com">电视</a>
129                 <a href="https://www.mi.com">笔记本</a>
130                 <a href="https://www.mi.com">平板</a>
131             </div>
132             <div class="hw search"></div>
133             <div style="clear: both;"></div>
134         </div>
135     </div>
136 </body>
137 </html>
138
```

4.3 推荐区域

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
```

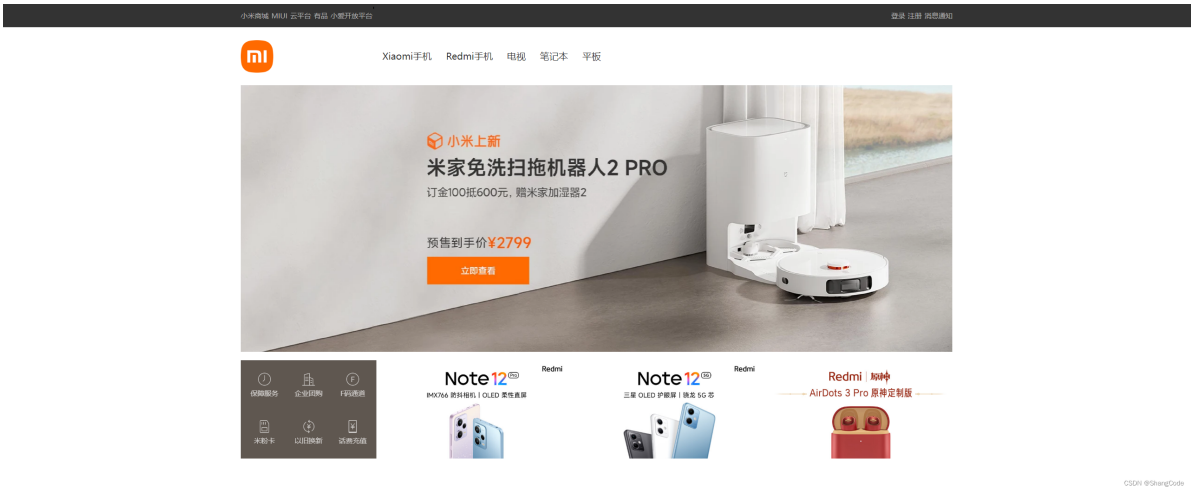
```
    <title>小米商城</title>
    <style>
        /* 去掉body的边距 */
        body {
            margin: 0;
        }

        img {
            width: 100%;
            height: 100%;
        }

        .left {
            float: left;
        }

        .margin_left {
            margin-left: 14.5px;
        }

        .header {
            background-color: #333;
        }

        /* 让中间内容居中 */
        .container {
            width: 1226px;
            margin: 0 auto;     /* 上下为0, 左右为auto */
        }

        /* header class 下的标签 a 自动应用这个样式 */
        .header a {
            color: #b0b0b0;
            line-height: 40px;
            display: inline-block;
            font-size: 12px;
        }

        .header .menu {
            float: left;
            color: white;
        }

        .header a {
            text-decoration: none;
        }

        .header a:hover {
            color: white;
        }

        .header .account {
            float: right;
            color: white;
        }

```

```
        .sub-header {
            height: 100px;
        }

        .sub-header .hw {
            width: 234px;
            height: 100px;
        }

        .sub-header .logo {
            float: left;
        }

        /* a标签是行内标签,默认不支持设置高度与边距 因此设置padding是不起作用的,因此可以加上inline-block */
        .sub-header .logo a {
            padding-top: 22px;
            padding-bottom: 22px;
            display: inline-block;
        }

        /* 设置logo的图片像素大小 */
        .sub-header .logo img {
            height: 56px;
            width: 56px;
        }

        .sub-header .menu {
            width: 400px;
            float:left;
            line-height: 100px;      /* 与行高度保持一致 */
        }

        .sub-header .menu a {
            text-decoration: none;  /* 去掉 a 标签的下划线 */
            color: #333;
            font-size: 16px;
            padding: 0 10px;        /* 设置字体的左右外边距 */
            display: inline-block;
        }

        /* 鼠标放到字体时,使字体变红 */
        .sub-header .menu a:hover {
            color: #ff6700;
        }

        .sub-header .search {
            float: right;
        }

        .slider {
            height: 460px;
        }

        .news{
            margin-top: 14px;
```

```
        }

        .news .channel {
            width: 228px;
            height: 164px;
            background-color: #5f5750;
            padding: 3px;
        }

        .news .channel .item {
            width: 76px;
            height: 82px;
            float: left;
            text-align: center;
        }

        .news .channel .item img {
            width: 24px;
            height: 24px;
            display: block;         /* 让图片自己占一整行 */
            margin: 0 auto;         /* 让图片垂直居中 */
            margin-bottom: 4px;     /* 设置图片与下方字体的间距 */
        }

        .news .channel .item a {
            display: inline-block;
            font-size: 12px;        /* 设置字体大小 */
            text-decoration: none;  /* a标签去掉下划线 */
            padding-top: 18px;
            color: #fff;          /* 设置字体为白色 */
            opacity: 0.7;           /* 设置透明度 */
        }

        .news .channel .item a:hover {
            opacity: 1;           /* 设置透明度 */
        }


        .news .list-item {
            width: 316px;
            height: 170px;
        }



    </style>

</head>
<body>
    <div class="header">
        <div class="container">
            <div class="menu">
                <a href="https://www.mi.com">小米商城</a>
                <a href="https://www.mi.com">MIUI</a>
                <a href="https://www.mi.com">云平台</a>
                <a href="https://www.mi.com">有品</a>
```

```
                <a href="https://www.mi.com">小爱开放平台</a>
            </div>
            <div class="account">
                <a href="https://www.mi.com">登录</a>
                <a href="https://www.mi.com">注册</a>
                <a href="https://www.mi.com">消息通知</a>
            </div>'
            <div style="clear: both;"></div>
        </div>
    </div>
    <div class="sub-header">
        <div class="container">
            <div class="hw logo">
                <a href="https://www.mi.com">
                    <img src="https://s02.mifile.cn/assets/static/image/logo-mi2.png" alt="小米官网">
                </a>
            </div>
            <div class="hw menu">
                <a href="https://www.mi.com">Xiaomi手机</a>
                <a href="https://www.mi.com">Redmi手机</a>
                <a href="https://www.mi.com">电视</a>
                <a href="https://www.mi.com">笔记本</a>
                <a href="https://www.mi.com">平板</a>
            </div>
            <div class="hw search"></div>
            <div style="clear: both;"></div>
        </div>
    </div>
    <div class="slider">
        <div class="container">
            <div>
                <img src="https://cdn.cnbj1.fds.api.mi-img.com/mi-mall/454c1da2c5b64a3f2c07c5a4c01aa9c4.jpg?thumb=1&w=1533&h=575&f=webp&q=90" alt="推荐商品">
            </div>
        </div>
    </div>
    <div class="news">
        <div class="container">
            <div class="channel left">
                <div class="item">
                    <a href="https://www.mi.com">
                        <img src="https://cdn.cnbj1.fds.api.mi-img.com/mi-mall/82abdba456e8caaea5848a0cddce03db.png?w=48&h=48" alt="">
                        <div>保障服务</div>
                    </a>
                </div>
                <div class="item">
                    <a href="https://www.mi.com">
                        <img src="https://cdn.cnbj1.fds.api.mi-img.com/mi-mall/806f2dfb2d27978e33fe3815d3851fa3.png?w=48&h=48" alt="">
                        <div>企业团购</div>
                    </a>
                </div>
                <div class="item">
```

```
                    <a href="https://www.mi.com">
                        <img src="https://cdn.cnbj1.fds.api.mi-img.com/mi-mall/eded6fa3b897a058163e2485532c4f10.png?w=48&h=48" alt="">
                        <div>F码通道</div>
                    </a>
                </div>
                <div class="item">
                    <a href="https://www.mi.com">
                        <img src="https://cdn.cnbj1.fds.api.mi-img.com/mi-mall/43a3195efa6a3cc7662efed8e7abe8bf.png?w=48&h=48" alt="">
                        <div>米粉卡</div>
                    </a>
                </div>
                <div class="item">
                    <a href="https://www.mi.com">
                        <img src="https://cdn.cnbj1.fds.api.mi-img.com/mi-mall/f4846bca6010a0deb9f85464409862af.png?w=48&h=48" alt="">
                        <div>以旧换新</div>
                    </a>
                </div>
                <div class="item">
                    <a href="https://www.mi.com">
                        <img src="https://cdn.cnbj1.fds.api.mi-img.com/mi-mall/9a76d7636b08e0988efb4fc384ae497b.png?w=48&h=48" alt="">
                        <div>话费充值</div>
                    </a>
                </div>
                <div style="clear: both;"></div>
            </div>
            <div class="list-item left margin_left">
                <img src="https://cdn.cnbj1.fds.api.mi-img.com/mi-mall/d0c515086acb3c3a3e976ad20901aac5.jpg?w=632&h=340" alt="">
            </div>
            <div class="list-item left margin_left">
                <img src="https://cdn.cnbj1.fds.api.mi-img.com/mi-mall/254c711cc71facf156ac955b8719dffa.jpg?w=632&h=340" alt="">
            </div>
            <div class="list-item left margin_left">
                <img src="https://cdn.cnbj1.fds.api.mi-img.com/mi-mall/2b120c0dddc056dcb36e847269fb92cd.jpg?w=632&h=340" alt="">
            </div>
            <div style="clear: both;"></div>
        </div>
    </div>
</body>
</html>

```



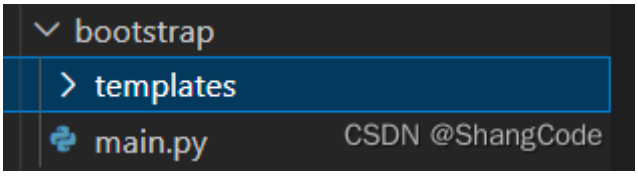
5. Bootstrap

> 别人已经帮忙写好的CSS样式

使用方式:

- 下载Bootstrap
- 使用:
    - 在页面上引入 Bootstrap
    - 编写HTML时,按照Bootstrap的规定来编写或者自定制

> 由于我没有下载Pycharm,无法本地实时测试,我使用的VSCode进行的编辑,所以我继续使用Flaskweb进行页面的访问测试

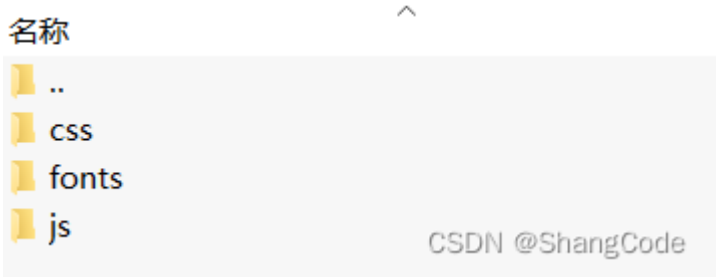


5.1 初识Bootstrap

下载地址: https://v3.bootcss.com/


```
BaiDuNetDisk Download:
链接：https://pan.baidu.com/s/1rcZldkNHrpC11f2plUv-rg?pwd=mh5b
提取码：mh5b
123
```

下载完成后解压,目录如下:



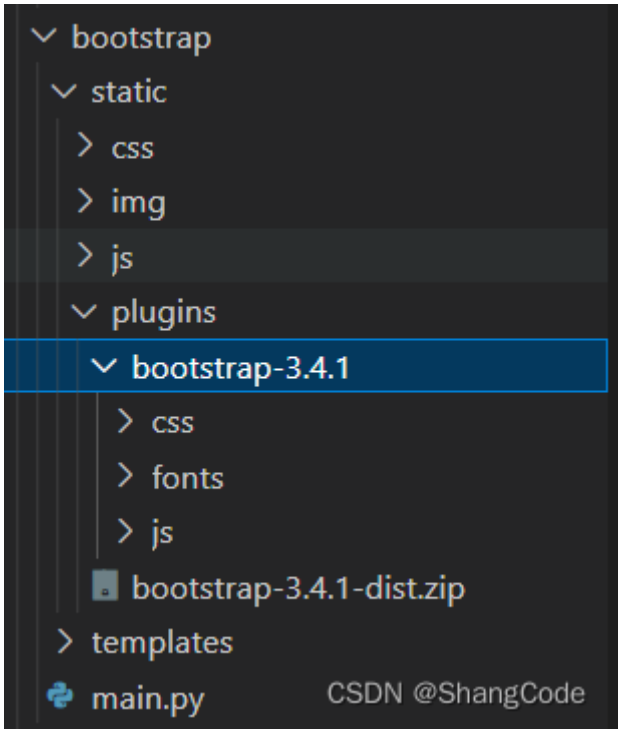
在服务器中创建必要的目录

```
[root@hecs-33592 python]# cd bootstrap/
[root@hecs-33592 bootstrap]# ls
main.py  templates
[root@hecs-33592 bootstrap]# mkdir static
[root@hecs-33592 bootstrap]# cd static/
[root@hecs-33592 static]# ls
[root@hecs-33592 static]# mkdir css
[root@hecs-33592 static]# mkdir js
```

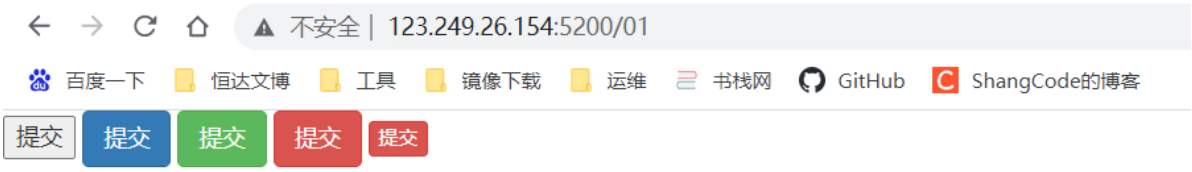
```
[root@hecs-33592 static]# mkdir img
[root@hecs-33592 static]# mkdir plugins



[root@hecs-33592 static]# tree /root/python/bootstrap/
/root/python/bootstrap/
├── main.py
├── static
│   ├── css
│   ├── img
│   ├── js
│   └── plugins
└── templates

12345678910111213141516171819202122 23
```

我会把刚刚下载好的`bootstrap-3.4.1-dist.zip`解压放到`plugins`下

```
[root@hecs-33592 plugins]# ls
bootstrap-3.4.1-dist.zip
[root@hecs-33592 plugins]# unzip bootstrap-3.4.1-dist.zip
[root@hecs-33592 plugins]# mv bootstrap-3.4.1-dist bootstrap-3.4.1
[root@hecs-33592 plugins]# tree bootstrap-3.4.1
bootstrap-3.4.1
├── css
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap.min.css.map
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   ├── bootstrap-theme.min.css
│   └── bootstrap-theme.min.css.map
├── fonts
│   ├── glyphicons-halflings-regular.eot
│   ├── glyphicons-halflings-regular.svg
│   ├── glyphicons-halflings-regular.ttf
│   ├── glyphicons-halflings-regular.woff
│   └── glyphicons-halflings-regular.woff2
└── js
    ├── bootstrap.js
    ├── bootstrap.min.js
    └── npm.js

12345678910111213141516171819202122232425 26
```

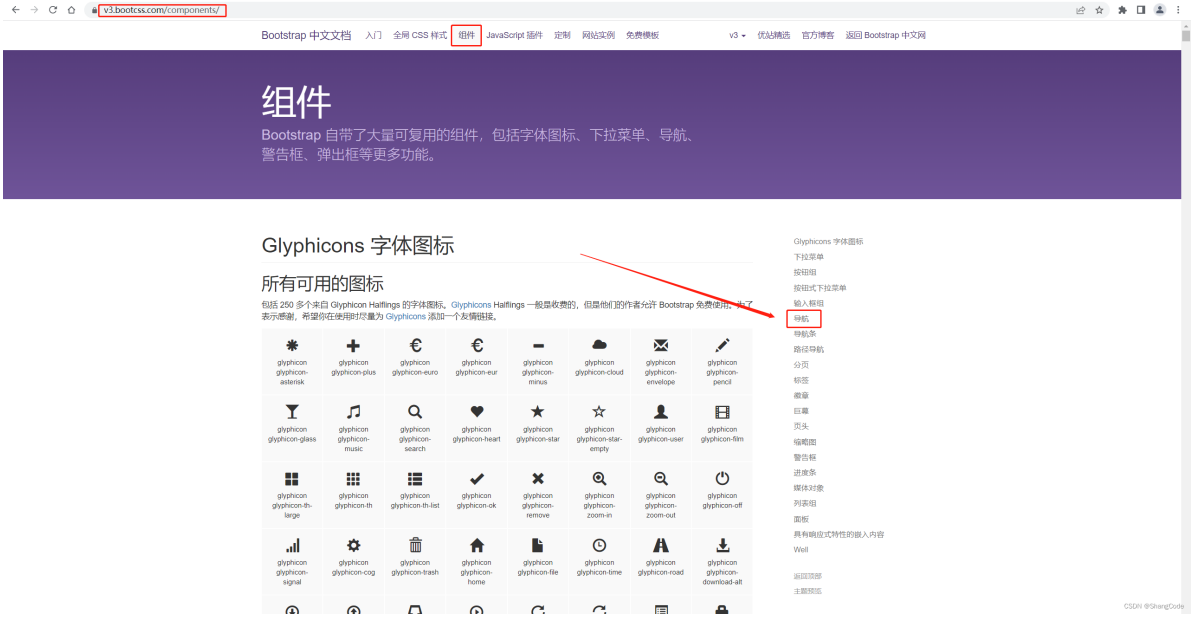



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <!-- 开发版本 -->
    <link rel="stylesheet" href="static/plugins/bootstrap-3.4.1/css/bootstrap.css">

    <!-- 生产版本 -->
    <link rel="stylesheet" href="static/plugins/bootstrap-3.4.1/css/bootstrap.min.css">

</head>
<body>
    <input type="button" value="提交">
    <input type="button" value="提交" class="btn btn-primary">
    <input type="button" value="提交" class="btn btn-success">
    <input type="button" value="提交" class="btn btn-danger">
    <input type="button" value="提交" class="btn btn-danger btn-xs">
</body>
</html>
```



接下来使用已经写好的导航栏

链接地址: https://v3.bootcss.com/components/



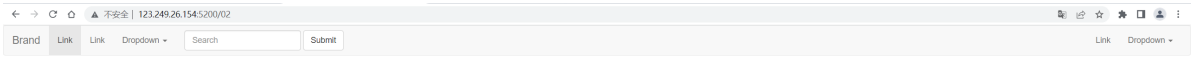
复制上面的代码

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <!-- 开发版本 -->
    <link rel="stylesheet" href="static/plugins/bootstrap-3.4.1/css/bootstrap.css">

    <!-- 生产版本 -->
    <link rel="stylesheet" href="static/plugins/bootstrap-3.4.1/css/bootstrap.min.css">

</head>

<body>
    <nav class="navbar navbar-default">
        <div class="container-fluid">
            <!-- Brand and toggle get grouped for better mobile display -->
            <div class="navbar-header">
```

```
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
                    data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="#">Brand</a>
            </div>

            <!-- Collect the nav links, forms, and other content for toggling -->
            <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                <ul class="nav navbar-nav">
                    <li class="active"><a href="#">Link <span class="sr-only">(current)</span></a></li>
                    <li><a href="#">Link</a></li>
                    <li class="dropdown">
                        <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true"
                            aria-expanded="false">Dropdown <span class="caret"></span></a>
                        <ul class="dropdown-menu">
                            <li><a href="#">Action</a></li>
                            <li><a href="#">Another action</a></li>
                            <li><a href="#">Something else here</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">Separated link</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">One more separated link</a></li>
                        </ul>
                    </li>
                </ul>
                <form class="navbar-form navbar-left">
                    <div class="form-group">
                        <input type="text" class="form-control" placeholder="Search">
                    </div>
                    <button type="submit" class="btn btn-default">Submit</button>
                </form>
                <ul class="nav navbar-nav navbar-right">
                    <li><a href="#">Link</a></li>
                    <li class="dropdown">
                        <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true"
                            aria-expanded="false">Dropdown <span class="caret"></span></a>
                        <ul class="dropdown-menu">
                            <li><a href="#">Action</a></li>
                            <li><a href="#">Another action</a></li>
                            <li><a href="#">Something else here</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">Separated link</a></li>
                        </ul>
```

```
                    </li>
                </ul>
            </div><!-- /.navbar-collapse -->
        </div><!-- /.container-fluid -->
    </nav>
</body>

</html>
```

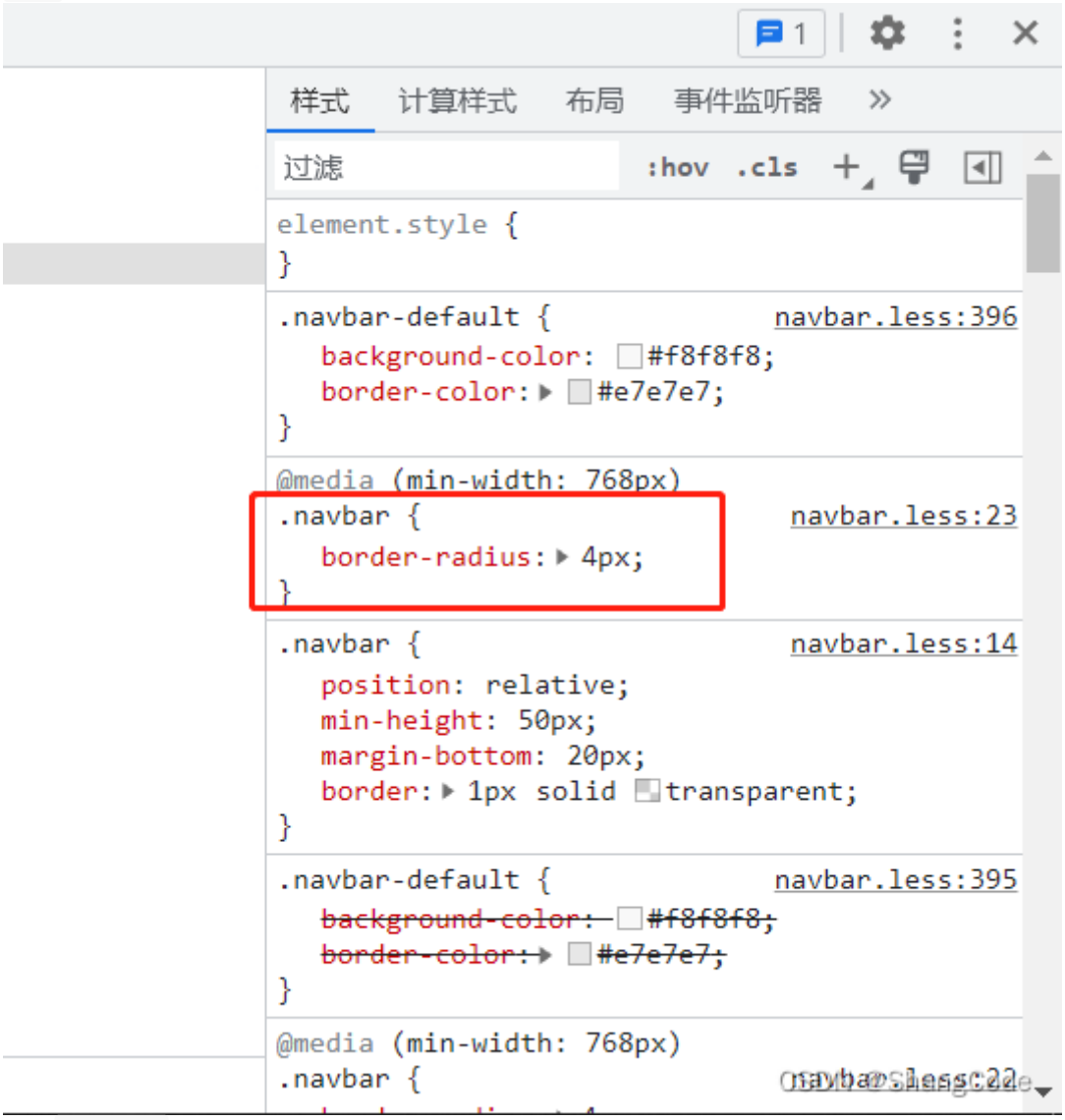
访问效果如下:



其实你仔细看会发现这个导航栏是有 圆角 的

接下来我们去掉 圆角

F12 调试页面

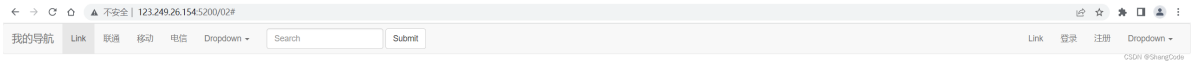


覆盖 .navbar 样式

```
<style>
    .navbar {
        border-radius: 0;
    }
</style>
12345
```

再次访问就没有圆角了

> 可以在相应的位置进行修改,代码部分自己测试修改哈



5.2 栅格系统

栅格系统介绍

> 整体划分为了 12 格

大致分为四种风格

> .col-xs-
> .col-sm-
> .col-md-
> .col-lg-

	超小屏幕 手机 (<768px)	小屏幕 平板 (≥768px)	中等屏幕 桌面显示器 (≥992px)	大屏幕 大桌面显示器 (≥1200px)
栅格系统行为	总是水平排列	开始是堆叠在一起的，当大于这些阈值时将变为水平排列C		
`.container` 最大宽度	None （自动）	750px	970px	1170px
类前缀	`.col-xs-`	`.col-sm-`	`.col-md-`	`.col-lg-`
列（column）数	12			
最大列（column）宽	自动	~62px	~81px	~97px
槽（gutter）宽	30px （每列左右均有 15px）			
可嵌套	是			
偏移（Offsets）	是			
列排序	是			

响应式:根据页面的宽度,动态的改变布局

- .col-sm- : 750px
- .col-md- : 970px
- .col-lg- : 1170px

非响应式:

- .col-xs-

```
<div class="col-xs-2" style="background-color: brown; height: 20px;"></div>
<div class="col-xs-10" style="background-color: green;height: 20px;"></div>
12
```


列偏移

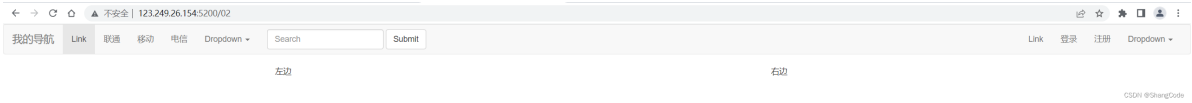
> col-sm-offset-

```
<div class="col-sm-offset-3 col-sm-2" style="background-color: brown; height: 20px;"></div>
<div class="col-sm-7" style="background-color: green;height: 20px;"></div>
```

5.3 container

- container

```
<div class="container clearfix">
    <div class="col-sm-9">左边</div>
    <div class="col-sm-3">右边</div>
</div>
```

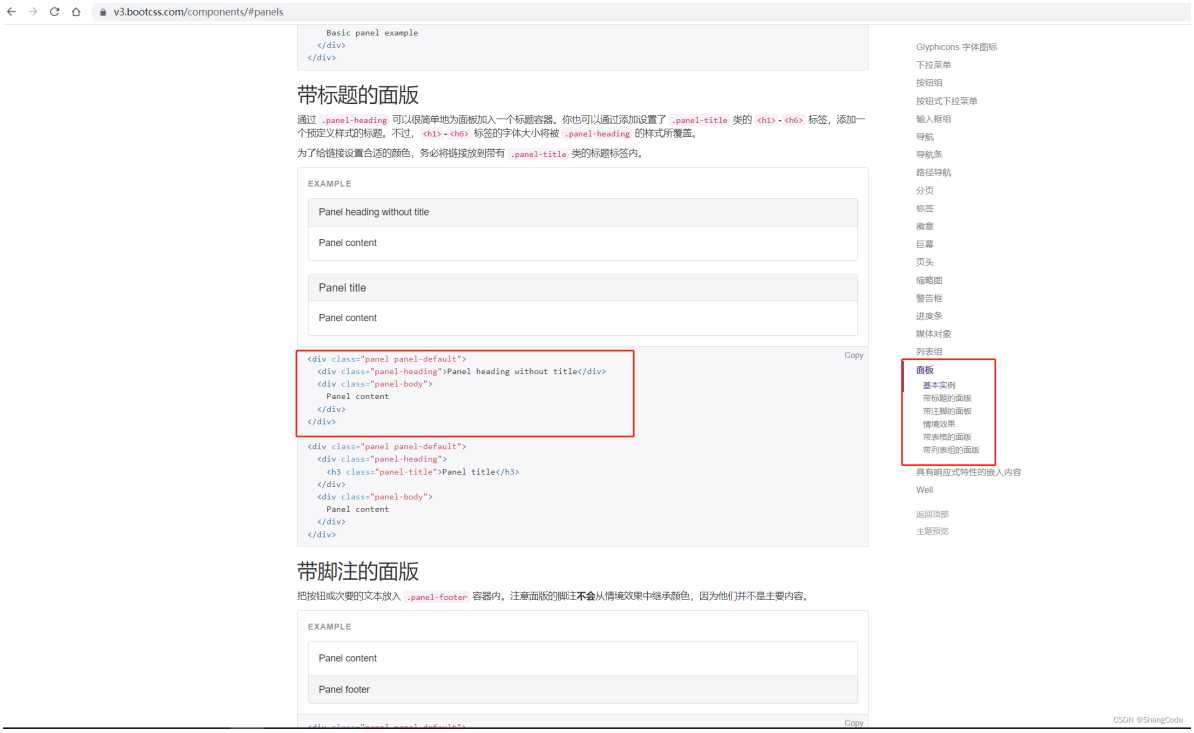


- container-fluid

```
<div class="container-fluid clearfix">
    <div class="col-sm-9">左边</div>
    <div class="col-sm-3">右边</div>
</div>
```

5.4 面板

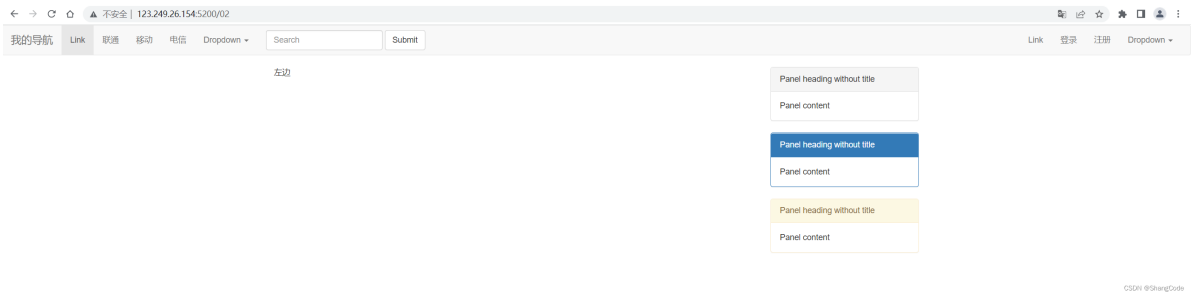
地址: https://v3.bootcss.com/components/#panels




```
<body>
    <nav class="navbar navbar-default">
        <div class="container-fluid">
```

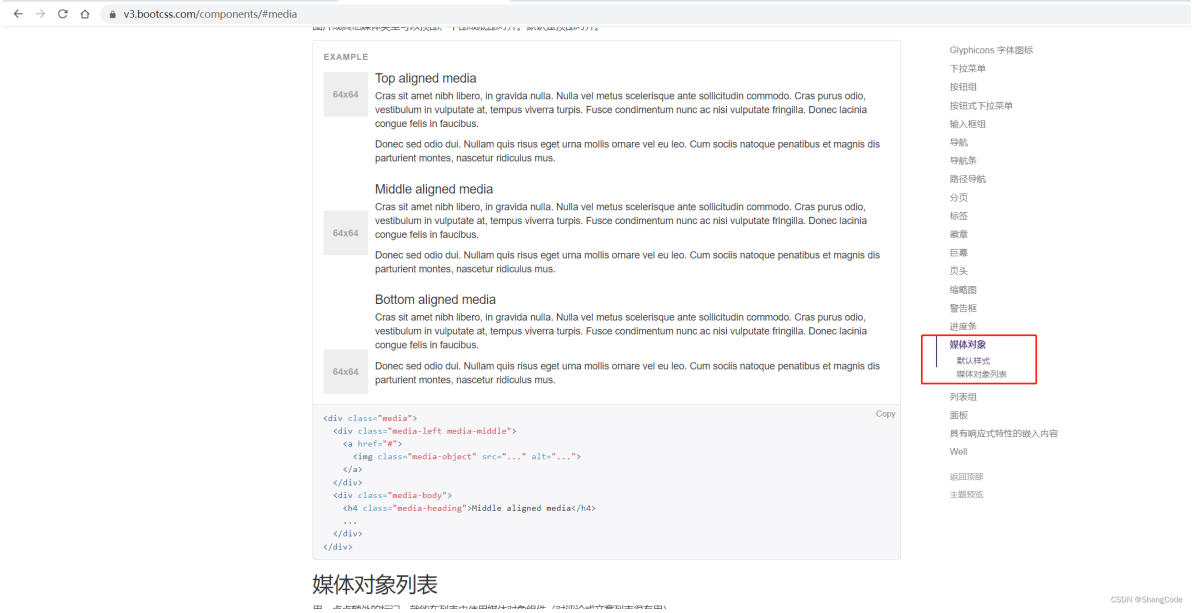
```
            <!-- Brand and toggle get grouped for better mobile display -->
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
                    data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="#">我的导航</a>
            </div>

            <!-- Collect the nav links, forms, and other content for toggling -->
            <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                <ul class="nav navbar-nav">
                    <li class="active"><a href="#">Link <span class="sr-only">(current)</span></a></li>
                    <li><a href="#">联通</a></li>
                    <li><a href="#">移动</a></li>
                    <li><a href="#">电信</a></li>
                    <li class="dropdown">
                        <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true"
                            aria-expanded="false">Dropdown <span class="caret"></span></a>
                        <ul class="dropdown-menu">
                            <li><a href="#">Action</a></li>
                            <li><a href="#">Another action</a></li>
                            <li><a href="#">Something else here</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">Separated link</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">One more separated link</a></li>
                        </ul>
                    </li>
                </ul>
                <form class="navbar-form navbar-left">
                    <div class="form-group">
                        <input type="text" class="form-control" placeholder="Search">
                    </div>
                    <button type="submit" class="btn btn-default">Submit</button>
                </form>
                <ul class="nav navbar-nav navbar-right">
                    <li><a href="#">Link</a></li>
                    <li><a href="#">登录</a></li>
                    <li><a href="#">注册</a></li>
                    <li class="dropdown">
                        <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true"
                            aria-expanded="false">Dropdown <span class="caret"></span></a>
                        <ul class="dropdown-menu">
```

```
                            <li><a href="#">Action</a></li>
                            <li><a href="#">Another action</a></li>
                            <li><a href="#">Something else here</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">Separated link</a></li>
                        </ul>
                    </li>
                </ul>
            </div><!-- /.navbar-collapse -->
        </div><!-- /.container-fluid -->
    </nav>
    <div class="container clearfix">
        <div class="col-sm-9">左边</div>
        <div class="col-sm-3">
            <div class="panel panel-default">
                <div class="panel-heading">Panel heading without title</div>
                <div class="panel-body">
                    Panel content
                </div>
            </div>
            <div class="panel panel-primary">
                <div class="panel-heading">Panel heading without title</div>
                <div class="panel-body">
                    Panel content
                </div>
            </div>
            <div class="panel panel-warning">
                <div class="panel-heading">Panel heading without title</div>
                <div class="panel-body">
                    Panel content
                </div>
            </div>
        </div>
    </div>

</body>
```

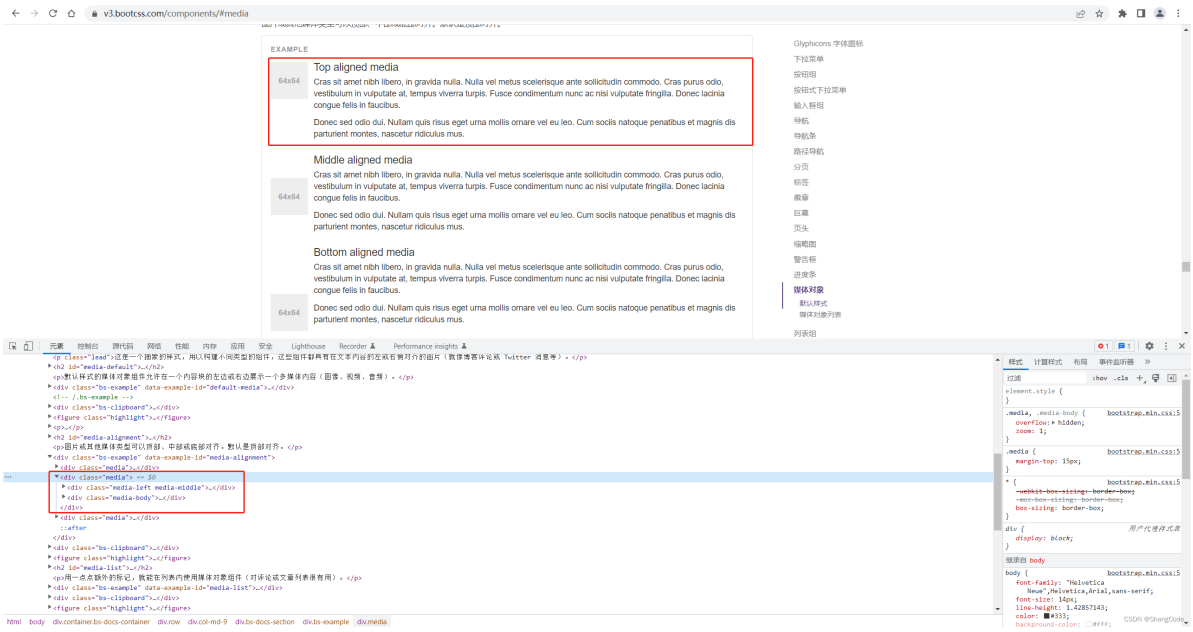


5.5 媒体对象

添加媒体对象



由于官方文档给的示例代码不全,所以可以F12查看源码,复制页面中的样式

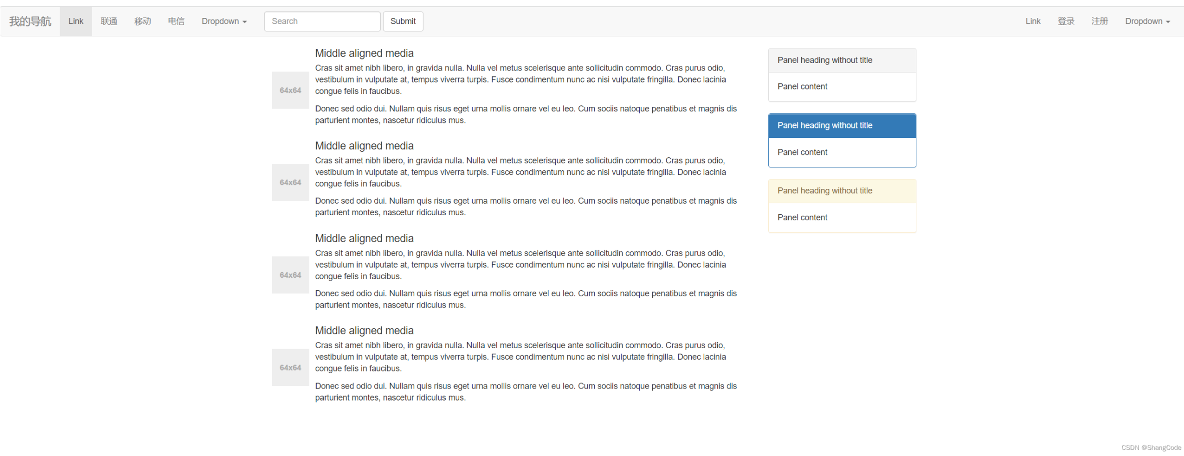



```
<div class="col-sm-9">
    <div class="media">
        <div class="media-left media-middle">
            <a href="#">
                <img class="media-object" data-src="holder.js/64x64" alt="64x64"
                    style="width: 64px; height: 64px;"
```

```
 src="
HN0YW5kYWxvbmU9InllcyI/PjxzdmcgeG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB3aW
R0aD0iNjQiIGhlaWdodD0iNjQiIHZpZXdCb3g9IjAgMCA2NCA2NCIgcHJlc2VydmVBc3BlY3RSYXRpbz0
ibm9uZSI+PCEtLQpTb3VyY2UgVVJMOiBob2xkZXIuanMvNjR4NjQKQ3JlYXRlZCB3aXRoIEhvbGRlci5q
cyAyLjYuMC4KTGVhcm4gbW9yZSBhdCBodHRwOi8vaG9sZGVyanMuY29tCihjKSAyMDEyLTIwMTUgSXZhb
iBNYWxvcGluc2t5IC0gaHR0cDovL2ltc2t5LmNvCi0tPjxkZWZzPjxzdHlsZSB0eXBlPSJ0ZXh0L2Nzcy
I+PCFbQ0RBVEFbI2hvbGRlcl8xODRlYTE3NjE2OSB0ZXh0IHsgZmlsbDojQUFBQUFBO2ZvbnQtd2VpZ2h
0OmJvbGQ7Zm9udC1mYW1pbHk6QXJpYWwsIEhlbHZldGljYSwgT3BlbiBTYW5zLCBzYW5zLXNlcmlmLCBt
b25vc3BhY2U7Zm9udC1zaXplOjEwcHQgfSBdXT48L3N0eWxlPjwvZGVmcz48ZyBpZD0iaG9sZGVyXzE4N
GVhMTc2MTY5Ij48cmVjdCB3aWR0aD0iNjQiIGhlaWdodD0iNjQiIGZpbGw9IiNFRUVFRUUiLz48Zz48dG
V4dCB4PSIxMy4xNzUyMTg1ODIxNTMzMiIgeT0iMzYuNTU1OTk5OTQyNzc5NTQiPjY0eDY0PC90ZXh0Pjw
vZz48L2c+PC9zdmc+"
                    data-holder-rendered="true">
            </a>
        </div>
        <div class="media-body">
            <h4 class="media-heading">Middle aligned media</h4>
            <p>Cras sit amet nibh libero, in gravida nulla. Nulla vel metus
scelerisque ante sollicitudin
                commodo. Cras purus odio, vestibulum in vulputate at, tempus
viverra turpis. Fusce condimentum
                nunc ac nisi vulputate fringilla. Donec lacinia congue felis in
faucibus.</p>
            <p>Donec sed odio dui. Nullam quis risus eget urna mollis ornare vel
eu leo. Cum sociis natoque
                penatibus et magnis dis parturient montes, nascetur ridiculus
mus.</p>
        </div>
    </div>
    <div class="media">
        <div class="media-left media-middle">
            <a href="#">
                <img class="media-object" data-src="holder.js/64x64" alt="64x64"
                    style="width: 64px; height: 64px;"

 src="
HN0YW5kYWxvbmU9InllcyI/PjxzdmcgeG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB3aW
R0aD0iNjQiIGhlaWdodD0iNjQiIHZpZXdCb3g9IjAgMCA2NCA2NCIgcHJlc2VydmVBc3BlY3RSYXRpbz0
ibm9uZSI+PCEtLQpTb3VyY2UgVVJMOiBob2xkZXIuanMvNjR4NjQKQ3JlYXRlZCB3aXRoIEhvbGRlci5q
cyAyLjYuMC4KTGVhcm4gbW9yZSBhdCBodHRwOi8vaG9sZGVyanMuY29tCihjKSAyMDEyLTIwMTUgSXZhb
iBNYWxvcGluc2t5IC0gaHR0cDovL2ltc2t5LmNvCi0tPjxkZWZzPjxzdHlsZSB0eXBlPSJ0ZXh0L2Nzcy
I+PCFbQ0RBVEFbI2hvbGRlcl8xODRlYTE3NjE2OSB0ZXh0IHsgZmlsbDojQUFBQUFBO2ZvbnQtd2VpZ2h
0OmJvbGQ7Zm9udC1mYW1pbHk6QXJpYWwsIEhlbHZldGljYSwgT3BlbiBTYW5zLCBzYW5zLXNlcmlmLCBt
b25vc3BhY2U7Zm9udC1zaXplOjEwcHQgfSBdXT48L3N0eWxlPjwvZGVmcz48ZyBpZD0iaG9sZGVyXzE4N
GVhMTc2MTY5Ij48cmVjdCB3aWR0aD0iNjQiIGhlaWdodD0iNjQiIGZpbGw9IiNFRUVFRUUiLz48Zz48dG
V4dCB4PSIxMy4xNzUyMTg1ODIxNTMzMiIgeT0iMzYuNTU1OTk5OTQyNzc5NTQiPjY0eDY0PC90ZXh0Pjw
vZz48L2c+PC9zdmc+"
                    data-holder-rendered="true">
            </a>
        </div>
        <div class="media-body">
            <h4 class="media-heading">Middle aligned media</h4>
            <p>Cras sit amet nibh libero, in gravida nulla. Nulla vel metus
scelerisque ante sollicitudin
```

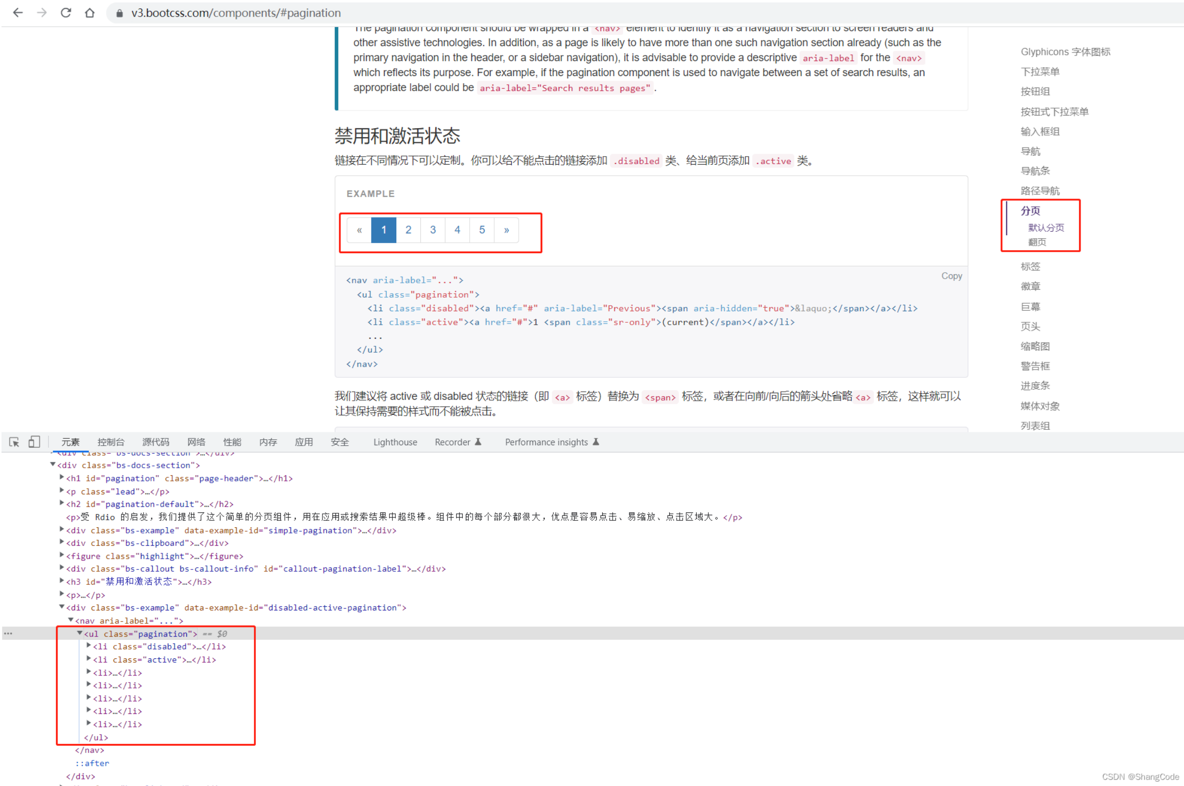
```
                commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum
                nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.</p>
            <p>Donec sed odio dui. Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque
                penatibus et magnis dis parturient montes, nascetur ridiculus mus.</p>
        </div>
    </div>
    <div class="media">
        <div class="media-left media-middle">
            <a href="#">
                <img class="media-object" data-src="holder.js/64x64" alt="64x64"
                    style="width: 64px; height: 64px;"

 src=""
                    data-holder-rendered="true">
            </a>
        </div>
        <div class="media-body">
            <h4 class="media-heading">Middle aligned media</h4>
            <p>Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin
                commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum
                nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.</p>
            <p>Donec sed odio dui. Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque
                penatibus et magnis dis parturient montes, nascetur ridiculus mus.</p>
        </div>
    </div>
    <div class="media">
        <div class="media-left media-middle">
            <a href="#">
                <img class="media-object" data-src="holder.js/64x64" alt="64x64"
                    style="width: 64px; height: 64px;"
```

```
src=""
                    data-holder-rendered="true">
            </a>
        </div>
        <div class="media-body">
            <h4 class="media-heading">Middle aligned media</h4>
            <p>Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin
                commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum
                nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.</p>
            <p>Donec sed odio dui. Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque
                penatibus et magnis dis parturient montes, nascetur ridiculus mus.</p>
        </div>
    </div>

```

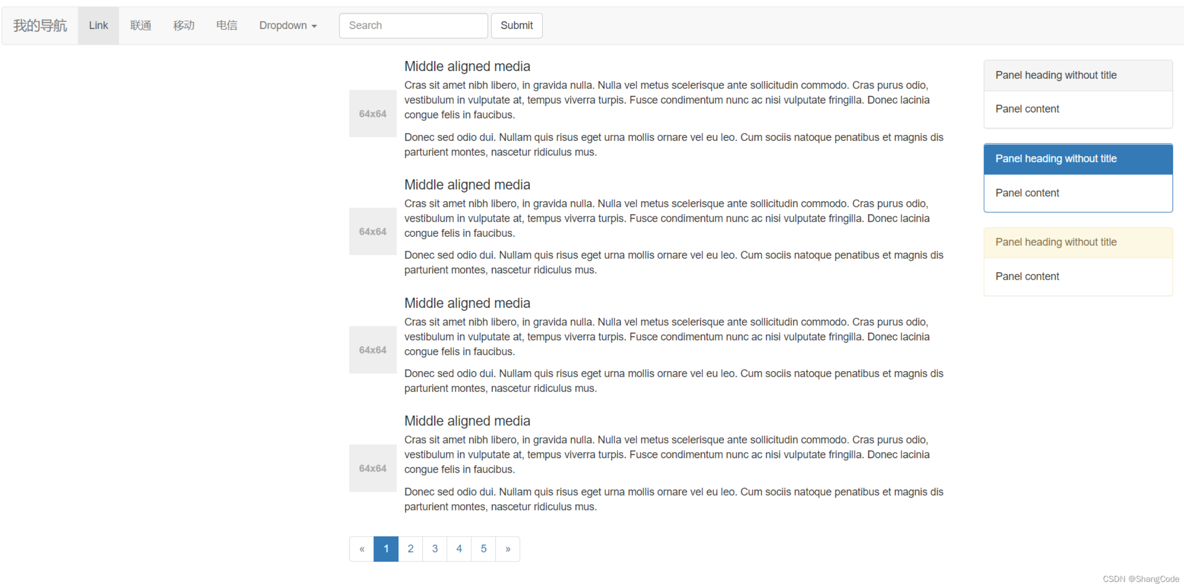


5.6 分页

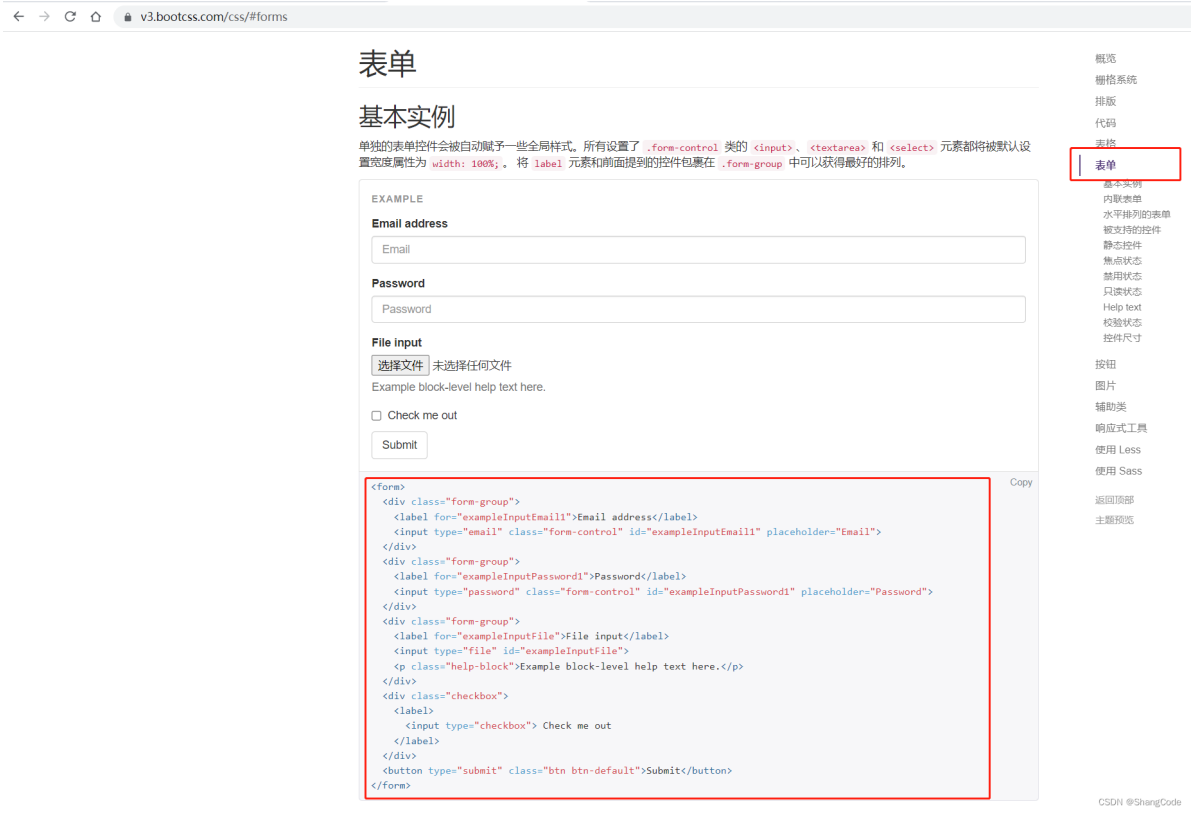
链接：https://v3.bootcss.com/components/#pagination




```html
<ul class="pagination">
                <li class="disabled"><a href="#" aria-label="Previous"><span aria-hidden="true">«</span></a></li>
                <li class="active"><a href="#">1 <span class="sr-only">(current)</span></a></li>
                <li><a href="#">2</a></li>
                <li><a href="#">3</a></li>
                <li><a href="#">4</a></li>
                <li><a href="#">5</a></li>
                <li><a href="#" aria-label="Next"><span aria-hidden="true">»</span></a></li>
            </ul>
```



案例: 登录



- 宽度 + 区域居中
- 内边距

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <link rel="stylesheet" href="static/plugins/bootstrap-3.4.1/css/bootstrap.css">

    <style>
        .ct {
            margin-left: auto;                  /* 设置水平居中 */
            margin-right: auto;
            margin-top: 200px;
            width: 500px;
            height: 350px;
            border: 2px solid black;
            padding: 20px 40px;                 /* 设置内边距 */
            border-radius: 10px;                /* 设置圆角*/
            box-shadow: 5px 5px 10px #aaa;     /* 设置阴影 水平 垂直 厚度 颜色*/
            background-color:bisque;
        }

        .ct h1 {
            text-align: center;
            margin-top: 10px;
        }
```

```
28
29          .ct button {
30              margin: 20px;
31          }
32      </style>
33
34  </head>
35
36  <body>
37      <div class="ct">
38          <div>
39              <h1>用户登录</h1>
40          </div>
41          <div>
42              <form>
43                  <div class="form-group">
44                      <label for="exampleInputEmail1">用户名</label>
45                      <input type="email" class="form-control"
    id="exampleInputEmail1" placeholder="请输入用户名">
46                  </div>
47                  <div class="form-group">
48                      <label for="exampleInputPassword1">密码</label>
49                      <input type="password" class="form-control"
    id="exampleInputPassword1" placeholder="Password">
50                  </div>
51                  <div style="text-align: center">
52                      <button type="submit" class="btn btn-primary">登 录</button>
53                      <button type="submit" class="btn btn-default">注 册</button>
54                  </div>
55              </form>
56          </div>
57      </div>
58  </body>
59
60  </html>
```

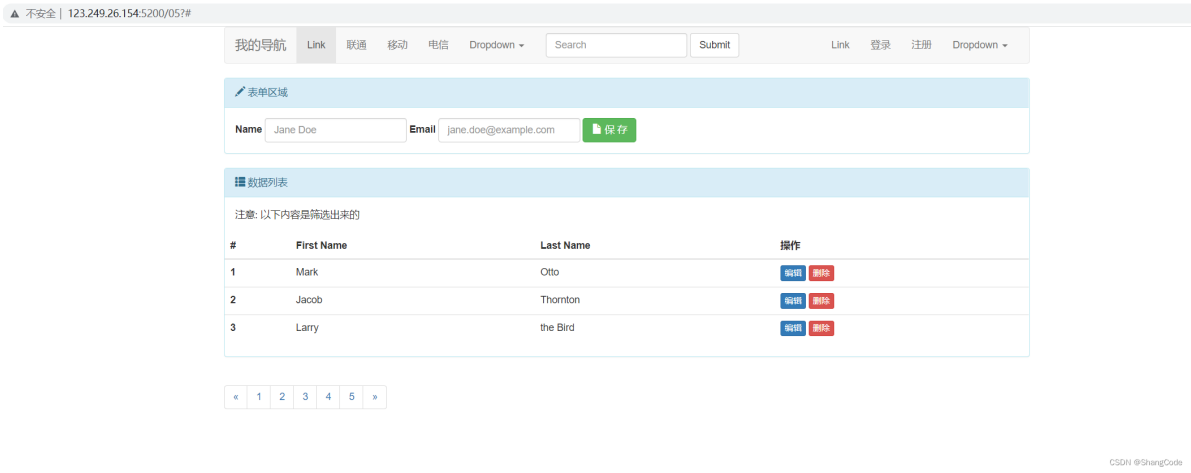

案例: 后台管理

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <!-- 开发版本 -->
    <link rel="stylesheet" href="static/plugins/bootstrap-3.4.1/css/bootstrap.css">

    <!-- 生产版本 -->
    <link rel="stylesheet" href="static/plugins/bootstrap-3.4.1/css/bootstrap.min.css">

    <style>
        .bt {
            margin: 20px;
        }
    </style>

</head>

<body>
    <div class="container">
        <nav class="navbar navbar-default">
            <div class="container-fluid">
                <!-- Brand and toggle get grouped for better mobile display -->
                <div class="navbar-header">
                    <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
                        data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
                        <span class="sr-only">Toggle navigation</span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                    </button>
                    <a class="navbar-brand" href="#">我的导航</a>
                </div>

                <!-- Collect the nav links, forms, and other content for toggling -->
                <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                    <ul class="nav navbar-nav">
                        <li class="active"><a href="#">Link <span class="sr-only">(current)</span></a></li>
                        <li><a href="#">联通</a></li>
                        <li><a href="#">移动</a></li>
                        <li><a href="#">电信</a></li>
                        <li class="dropdown">
                            <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button"
```

```
                                aria-haspopup="true" aria-
expanded="false">Dropdown <span class="caret"></span></a>
                            <ul class="dropdown-menu">
                                <li><a href="#">Action</a></li>
                                <li><a href="#">Another action</a></li>
                                <li><a href="#">Something else here</a></li>
                                <li role="separator" class="divider"></li>
                                <li><a href="#">Separated link</a></li>
                                <li role="separator" class="divider"></li>
                                <li><a href="#">One more separated link</a></li>
                            </ul>
                        </li>
                    </ul>
                    <form class="navbar-form navbar-left">
                        <div class="form-group">
                            <input type="text" class="form-control"
placeholder="Search">
                        </div>
                        <button type="submit" class="btn btn-
default">Submit</button>
                    </form>
                    <ul class="nav navbar-nav navbar-right">
                        <li><a href="#">Link</a></li>
                        <li><a href="#">登录</a></li>
                        <li><a href="#">注册</a></li>
                        <li class="dropdown">
                            <a href="#" class="dropdown-toggle" data-
toggle="dropdown" role="button"
                                aria-haspopup="true" aria-
expanded="false">Dropdown <span class="caret"></span></a>
                            <ul class="dropdown-menu">
                                <li><a href="#">Action</a></li>
                                <li><a href="#">Another action</a></li>
                                <li><a href="#">Something else here</a></li>
                                <li role="separator" class="divider"></li>
                                <li><a href="#">Separated link</a></li>
                            </ul>
                        </li>
                    </ul>
                </div><!-- /.navbar-collapse -->
            </div><!-- /.container-fluid -->
        </nav>

        <div class="panel panel-info">
            <div class="panel-heading">
                <span class="glyphicon glyphicon-pencil" aria-hidden="true">
</span>
                表单区域
            </div>
            <div class="panel-body">
                <form class="form-inline">
                    <div class="form-group">
                        <label for="exampleInputName2">Name</label>
                        <input type="text" class="form-control"
id="exampleInputName2" placeholder="Jane Doe">
                    </div>
```

```
                    <div class="form-group">
                        <label for="exampleInputEmail2">Email</label>
                        <input type="email" class="form-control"
id="exampleInputEmail2"
                               placeholder="jane.doe@example.com">
                    </div>
                    <button type="submit" class="btn btn-success">
                        <span class="glyphicon glyphicon-file" aria-
hidden="true"></span> 保 存
                    </button>
                </form>
            </div>
        </div>

        <div class="panel panel-info">
            <div class="panel-heading">
                <span class="glyphicon glyphicon-th-list" aria-hidden="true">
</span>
                数据列表
            </div>
            <div class="panel-body">
                注意：以下内容是筛选出来的
            </div>
            <div>
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>#</th>
                            <th>First Name</th>
                            <th>Last Name</th>
                            <th>操作</th>
                        </tr>
                    </thead>
                    <tbody>
                        <tr>
                            <th scope="row">1</th>
                            <td>Mark</td>
                            <td>Otto</td>
                            <td>
                                <a class="btn btn-primary btn-xs">编辑</a>
                                <a class="btn btn-danger btn-xs">删除</a>
                            </td>
                        </tr>
                        <tr>
                            <th scope="row">2</th>
                            <td>Jacob</td>
                            <td>Thornton</td>
                            <td>
                                <a class="btn btn-primary btn-xs">编辑</a>
                                <a class="btn btn-danger btn-xs">删除</a>
                            </td>
                        </tr>
                        <tr>
                            <th scope="row">3</th>
                            <td>Larry</td>
                            <td>the Bird</td>
```

```
                            <td>
                                <a class="btn btn-primary btn-xs">编辑</a>
                                <a class="btn btn-danger btn-xs">删除</a>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>


        <nav aria-label="Page navigation">
            <ul class="pagination">
              <li>
                <a href="#" aria-label="Previous">
                  <span aria-hidden="true">&laquo;</span>
                </a>
              </li>
              <li><a href="#">1</a></li>
              <li><a href="#">2</a></li>
              <li><a href="#">3</a></li>
              <li><a href="#">4</a></li>
              <li><a href="#">5</a></li>
              <li>
                <a href="#" aria_label="Next">
                  <span aria-hidden="true">&raquo;</span>
                </a>
              </li>
            </ul>
          </nav>
    </div>
</body>
</html>
```

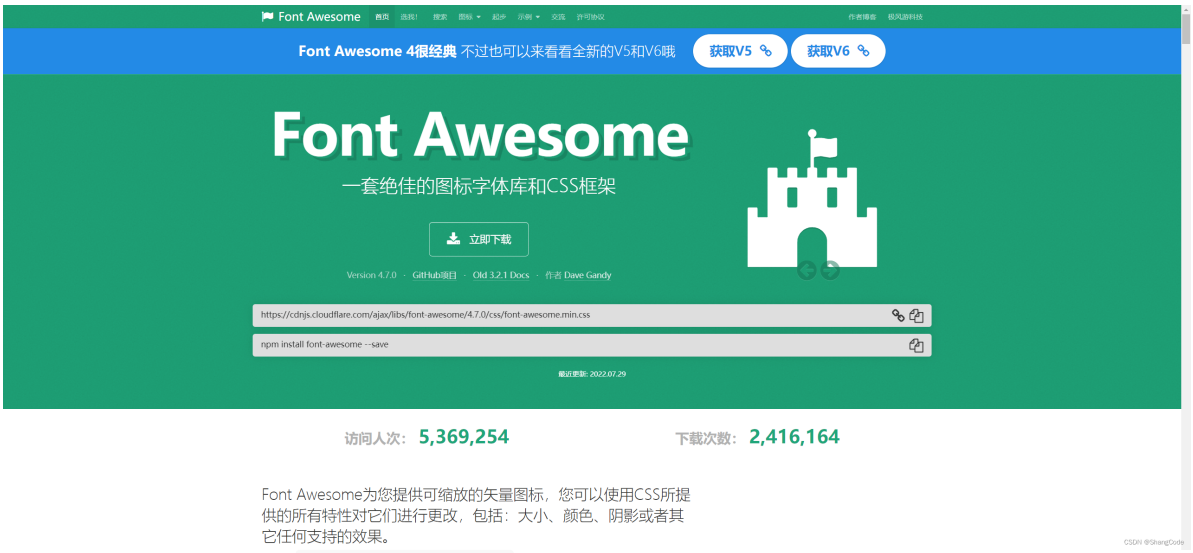


5.7 图标

上面的后台管理案例中,Bootstrap提供的图标不是太够用,我们需要一个专业做图标的网站

> 地址: **https://fontawesome.dashgame.com/**

下载

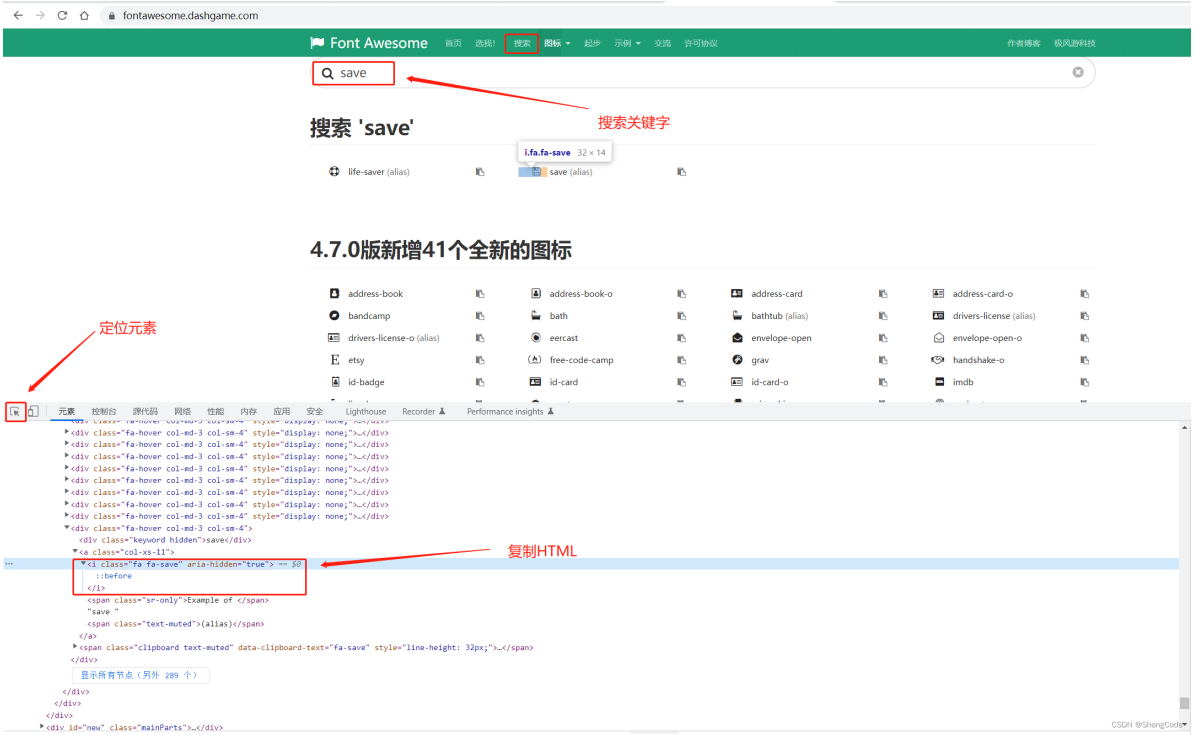


下载好后,上传至服务器的 `static/plugins` 下并解压

```
[root@hecs-33592 plugins]# pwd
/root/python/bootstrap/static/plugins
[root@hecs-33592 plugins]# ls
bootstrap-3.4.1  bootstrap-3.4.1-dist.zip  font-awesome-4.7.0  font-awesome-4.7.0.zip
[root@hecs-33592 plugins]# tree font-awesome-4.7.0
font-awesome-4.7.0
├── css
│   ├── font-awesome.css
│   └── font-awesome.min.css
├── fonts
│   ├── FontAwesome.otf
│   ├── fontawesome-webfont.eot
│   ├── fontawesome-webfont.svg
│   ├── fontawesome-webfont.ttf
│   ├── fontawesome-webfont.woff
│   └── fontawesome-webfont.woff2
├── HELP-US-OUT.txt
├── less
│   ├── animated.less
│   ├── bordered-pulled.less
│   ├── core.less
│   ├── fixed-width.less
│   ├── font-awesome.less
│   ├── icons.less
│   ├── larger.less
│   ├── list.less
│   ├── mixins.less
│   ├── path.less
│   ├── rotated-flipped.less
│   ├── screen-reader.less
│   ├── stacked.less
│   └── variables.less
```

```
└── scss
    ├── _animated.scss
    ├── _bordered-pulled.scss
    ├── _core.scss
    ├── _fixed-width.scss
    ├── font-awesome.scss
    ├── _icons.scss
    ├── _larger.scss
    ├── _list.scss
    ├── _mixins.scss
    ├── _path.scss
    ├── _rotated-flipped.scss
    ├── _screen-reader.scss
    ├── _stacked.scss
    └── _variables.scss
```

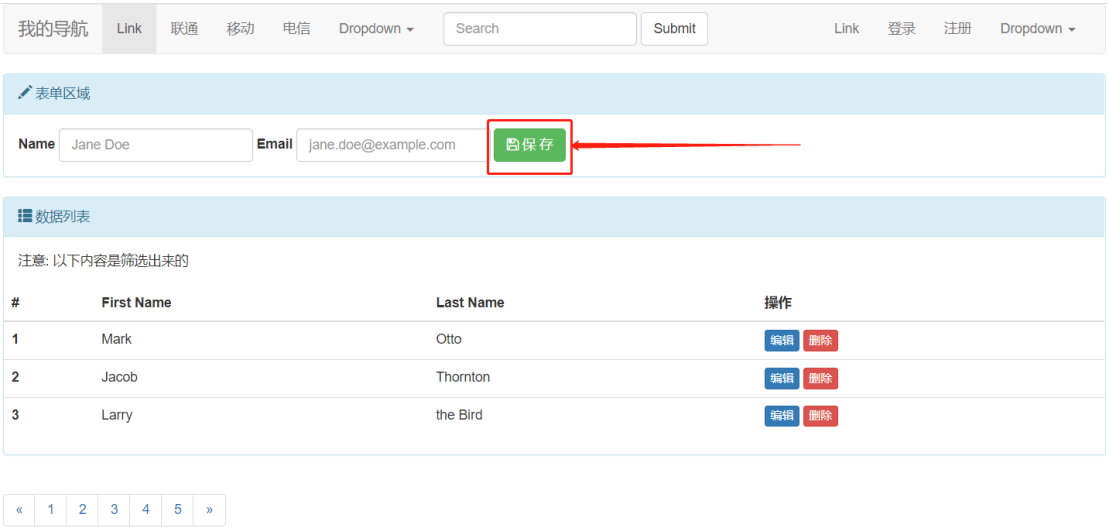

打开网址 https://fontawesome.dashgame.com/



放在代码的这里

```
101                         placeholder="jane.doe@example.com">
102                 </div>
103                 <button type="submit" class="btn btn-success">
104                     <i class="fa fa-save" aria-hidden="true"></i> 保 存
105                 </button>
106             </form>
107         </div>
108     </div>
```


访问



以此类推,很简单

优化

> 针对前面的导航页面进行优化

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <!-- 开发版本 -->
    <link rel="stylesheet" href="static/plugins/bootstrap-3.4.1/css/bootstrap.css">

    <!-- 生产版本 -->
    <link rel="stylesheet" href="static/plugins/bootstrap-3.4.1/css/bootstrap.min.css">

    <link rel="stylesheet" href="static/plugins/font-awesome-4.7.0/css/font-awesome.css">

    <style>
        .distance {
            margin-left: 40px;
        }
    </style>

</head>

<body>
    <nav class="navbar navbar-default">
        <div class="container-fluid">
            <!-- Brand and toggle get grouped for better mobile display -->
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
```

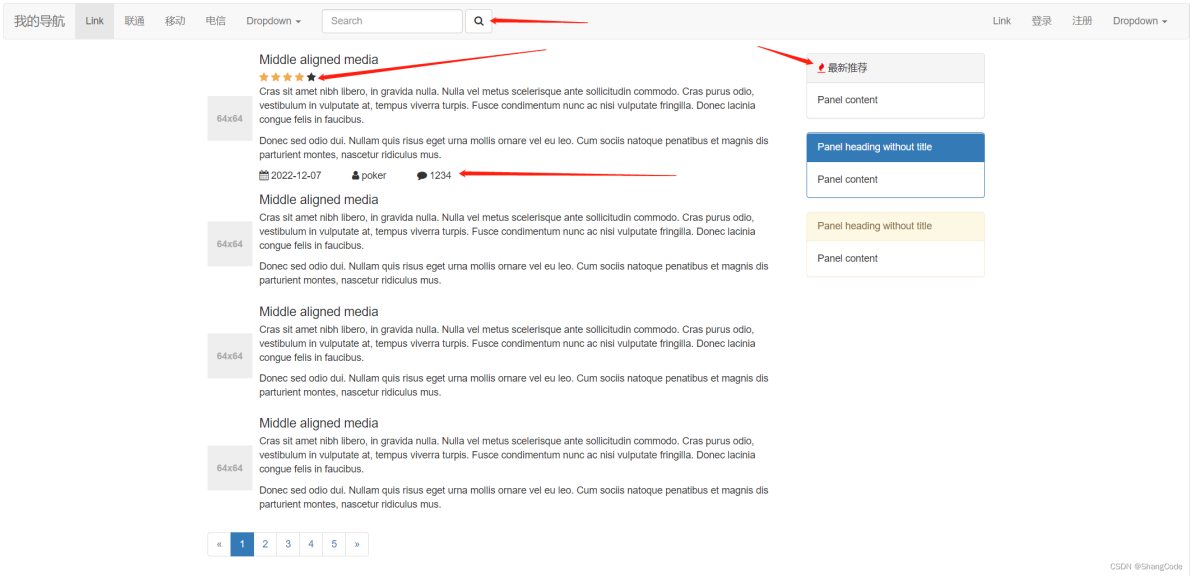
```
                    data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="#">我的导航</a>
            </div>

            <!-- Collect the nav links, forms, and other content for toggling -->
            <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                <ul class="nav navbar-nav">
                    <li class="active"><a href="#">Link <span class="sr-only">(current)</span></a></li>
                    <li><a href="#">联通</a></li>
                    <li><a href="#">移动</a></li>
                    <li><a href="#">电信</a></li>
                    <li class="dropdown">
                        <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true"
                            aria-expanded="false">Dropdown <span class="caret"></span></a>
                        <ul class="dropdown-menu">
                            <li><a href="#">Action</a></li>
                            <li><a href="#">Another action</a></li>
                            <li><a href="#">Something else here</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">Separated link</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">One more separated link</a></li>
                        </ul>
                    </li>
                </ul>
                <form class="navbar-form navbar-left">
                    <div class="form-group">
                        <input type="text" class="form-control" placeholder="Search">
                    </div>
                    <button type="submit" class="btn btn-default">
                        <i class="fa fa-search" aria-hidden="true"></i>
                    </button>
                </form>
                <ul class="nav navbar-nav navbar-right">
                    <li><a href="#">Link</a></li>
                    <li><a href="#">登录</a></li>
                    <li><a href="#">注册</a></li>
                    <li class="dropdown">
                        <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true"
                            aria-expanded="false">Dropdown <span class="caret"></span></a>
                        <ul class="dropdown-menu">
                            <li><a href="#">Action</a></li>
```

```
                            <li><a href="#">Another action</a></li>
                            <li><a href="#">Something else here</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">Separated link</a></li>
                        </ul>
                    </li>
                </ul>
            </div><!-- /.navbar-collapse -->
        </div><!-- /.container-fluid -->
    </nav>
    <div class="container clearfix">
        <div class="col-sm-9">
            <div class="media">
                <div class="media-left media-middle">
                    <a href="#">
                        <img class="media-object" data-src="holder.js/64x64" alt="64x64"
                            style="width: 64px; height: 64px;"
 src=""
                            data-holder-rendered="true">
                    </a>
                </div>
                <div class="media-body">
                    <h4 class="media-heading">Middle aligned media</h4>
                    <div>
                        <i class="fa fa-star" aria-hidden="true" style="color: #f0ad4e;"></i>
                        <i class="fa fa-star" aria-hidden="true" style="color: #f0ad4e;"></i>
                        <i class="fa fa-star" aria-hidden="true" style="color: #f0ad4e;"></i>
                        <i class="fa fa-star" aria-hidden="true" style="color: #f0ad4e;"></i>
                        <i class="fa fa-star" aria-hidden="true"></i>
                    </div>
                    <p>Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin
                        commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum
                        nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.</p>
                    <p>Donec sed odio dui. Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque
```

```
                        penatibus et magnis dis parturient montes, nascetur
ridiculus mus.</p>
                    <div>
                        <i class="fa fa-calendar" aria-hidden="true"></i> 2022-
12-07
                        <i class="fa fa-user distance" aria-hidden="true"></i>
poker
                        <i class="fa fa-comment distance" aria-hidden="true">
</i> 1234
                    </div>
                </div>
            </div>
            <div class="media">
                <div class="media-left media-middle">
                    <a href="#">
                        <img class="media-object" data-src="holder.js/64x64"
alt="64x64"
                            style="width: 64px; height: 64px;"

 src="
IHN0YW5kYWxvbmU9InllcyI/PjxzdmcgeG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB3
aWR0aD0iNjQiIGhlaWdodD0iNjQiIHZpZXdCb3g9IjAgMCA2NCA2NCIgcHJlc2VydmVBc3BlY3RSYXRp
bz0ibm9uZSI+PCEtLQpTb3VyY2UgVVJMOiBob2xkZXIuanMvNjR4NjQKQ3JlYXRlZCB3aXRoIEhvbGRl
ci5qcyAyLjYuMC4KTGVhcm4gbW9yZSBhdCBodHRwOi8vaG9sZGVyanMuY29tCihjKSAyMDEyLTIwMTUg
SXZhbiBNYWxvcGluc2t5IC0gaHR0cDovL2ltc2t5LmNvCi0tPjxkZWZzPjxzdHlsZSB0eXBlPSJ0ZXh0
L2NzcyI+PCFbQ0RBVEFbI2hvbGRlcl8xODRlYTE3NjE2OSB0ZXh0IHsgZmlsbDojQUFBQUFBO2ZvbnQt
d2VpZ2h0OmJvbGQ7Zm9udC1mYW1pbHk6QXJpYWwsIEhlbHZldGljYSwgT3BlbiBTYW5zLCBzYW5zLXNl
cmlmLCBtb25vc3BhY2U7Zm9udC1zaXplOjEwcHQgfSBdXT48L3N0eWxlPjwvZGVmcz48cmVjdCB3aWR0
Lz48dGV4dCB4PSIxMy4xNzUyMTg1ODIxNTMzMiIgeT0iMzYuNTU5OTk5OTQyNzc5NTQiPjY0eDY0
PC90ZXh0PjwvZz48L2c+PC9zdmc+"
                            data-holder-rendered="true">
                    </a>
                </div>
                <div class="media-body">
                    <h4 class="media-heading">Middle aligned media</h4>
                    <p>Cras sit amet nibh libero, in gravida nulla. Nulla vel
metus scelerisque ante sollicitudin
                        commodo. Cras purus odio, vestibulum in vulputate at,
tempus viverra turpis. Fusce condimentum
                        nunc ac nisi vulputate fringilla. Donec lacinia congue
felis in faucibus.</p>
                    <p>Donec sed odio dui. Nullam quis risus eget urna mollis
ornare vel eu leo. Cum sociis natoque
                        penatibus et magnis dis parturient montes, nascetur
ridiculus mus.</p>
                </div>
            </div>
            <div class="media">
                <div class="media-left media-middle">
                    <a href="#">
                        <img class="media-object" data-src="holder.js/64x64"
alt="64x64"
                            style="width: 64px; height: 64px;"
```

```
142
 src="
IHN0YW5kYWxvbmU9InllcyI/PjxzdmcgeG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB3
aWR0aD0iNjQiIGhlaWdodD0iNjQiIHZpZXdCb3g9IjAgMCA2NCA2NCIgcHJlc2VydmVBc3BlY3RSYXRp
bz0ibm9uZSI+PCEtLQpTb3VyY2UgVVJMOiBob2xkZXIuanMvNjR4NjQKQ3JlYXRlZCB3aXRoIEhvbGRl
ci5qcyAyLjYuMC4KTGVhcm4gbW9yZSBhdCBodHRwOi8vaG9sZGVyanMuY29tCihjKSAyMDEyLTIwMTUg
SXZhbiBNYWxvcGluc2t5IC0gaHR0cDovL2ltc2t5LmNvCi0tPjxkZWZzPjxzdHlsZSB0eXBlPSJ0ZXh0
L2NzcyI+PCFbQ0RBVEFbI2hvbGRlcl8xODRlYTE3NjE2OSB0ZXh0IHsgZmlsbDojQUFBQUFBO2ZvbnQt
d2VpZ2h0OmJvbGQ7Zm9udC1mYW1pbHk6QXJpYWwsIEhlbHZldGljYSwgT3BlbiBTYW5zLCBzYW5zLXNl
cmlmLCBtb25vc3BhY2U7Zm9udC1zaXplOjEwcHQgfSBdXT48L3N0eWxlPjwvZGVmcz48cmVjdCB3aWR0aD0iNjQiIGhlaWdodD0iNjQiIGZpbGw9IiNFRUVFRUUi
Lz48Zz48dGV4dCB4PSIxMy4xNzUyMTg1ODIxNTMzMiIgeT0iMzYuNTU5OTk5OTQyNzc5NTQiPjY0eDY0
PC90ZXh0PjwvZz48L2c+PC9zdmc+"
143                              data-holder-rendered="true">
144                      </a>
145                  </div>
146                  <div class="media-body">
147                      <h4 class="media-heading">Middle aligned media</h4>
148                      <p>Cras sit amet nibh libero, in gravida nulla. Nulla vel
    metus scelerisque ante sollicitudin
149                          commodo. Cras purus odio, vestibulum in vulputate at,
    tempus viverra turpis. Fusce condimentum
150                          nunc ac nisi vulputate fringilla. Donec lacinia congue
    felis in faucibus.</p>
151                      <p>Donec sed odio dui. Nullam quis risus eget urna mollis
    ornare vel eu leo. Cum sociis natoque
152                          penatibus et magnis dis parturient montes, nascetur
    ridiculus mus.</p>
153                  </div>
154              </div>
155              <div class="media">
156                  <div class="media-left media-middle">
157                      <a href="#">
158                          <img class="media-object" data-src="holder.js/64x64"
    alt="64x64"
159                              style="width: 64px; height: 64px;"
160
 src="
IHN0YW5kYWxvbmU9InllcyI/PjxzdmcgeG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB3
aWR0aD0iNjQiIGhlaWdodD0iNjQiIHZpZXdCb3g9IjAgMCA2NCA2NCIgcHJlc2VydmVBc3BlY3RSYXRp
bz0ibm9uZSI+PCEtLQpTb3VyY2UgVVJMOiBob2xkZXIuanMvNjR4NjQKQ3JlYXRlZCB3aXRoIEhvbGRl
ci5qcyAyLjYuMC4KTGVhcm4gbW9yZSBhdCBodHRwOi8vaG9sZGVyanMuY29tCihjKSAyMDEyLTIwMTUg
SXZhbiBNYWxvcGluc2t5IC0gaHR0cDovL2ltc2t5LmNvCi0tPjxkZWZzPjxzdHlsZSB0eXBlPSJ0ZXh0
L2NzcyI+PCFbQ0RBVEFbI2hvbGRlcl8xODRlYTE3NjE2OSB0ZXh0IHsgZmlsbDojQUFBQUFBO2ZvbnQt
d2VpZ2h0OmJvbGQ7Zm9udC1mYW1pbHk6QXJpYWwsIEhlbHZldGljYSwgT3BlbiBTYW5zLCBzYW5zLXNl
cmlmLCBtb25vc3BhY2U7Zm9udC1zaXplOjEwcHQgfSBdXT48L3N0eWxlPjwvZGVmcz48cmVjdCB3aWR0aD0iNjQiIGhlaWdodD0iNjQiIGZpbGw9IiNFRUVFRUUi
Lz48Zz48dGV4dCB4PSIxMy4xNzUyMTg1ODIxNTMzMiIgeT0iMzYuNTU5OTk5OTQyNzc5NTQiPjY0eDY0
PC90ZXh0PjwvZz48L2c+PC9zdmc+"
161                              data-holder-rendered="true">
162                      </a>
163                  </div>
164                  <div class="media-body">
165                      <h4 class="media-heading">Middle aligned media</h4>
166                      <p>Cras sit amet nibh libero, in gravida nulla. Nulla vel
    metus scelerisque ante sollicitudin
```

```
                        commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum
                        nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.</p>
                    <p>Donec sed odio dui. Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque
                        penatibus et magnis dis parturient montes, nascetur ridiculus mus.</p>
                </div>
            </div>
            <ul class="pagination">
                <li class="disabled"><a href="#" aria-label="Previous"><span aria-hidden="true">«</span></a></li>
                <li class="active"><a href="#">1 <span class="sr-only">(current)</span></a></li>
                <li><a href="#">2</a></li>
                <li><a href="#">3</a></li>
                <li><a href="#">4</a></li>
                <li><a href="#">5</a></li>
                <li><a href="#" aria-label="Next"><span aria-hidden="true">»</span></a></li>
            </ul>
        </div>
        <div class="col-sm-3">
            <div class="panel panel-default">
                <div class="panel-heading">
                    <i class="fa fa-fire" aria-hidden="true" style="color: red;"></i> 最新推荐
                </div>
                <div class="panel-body">
                    Panel content
                </div>
            </div>
            <div class="panel panel-primary">
                <div class="panel-heading">Panel heading without title</div>
                <div class="panel-body">
                    Panel content
                </div>
            </div>
            <div class="panel panel-warning">
                <div class="panel-heading">Panel heading without title</div>
                <div class="panel-body">
                    Panel content
                </div>
            </div>
        </div>
    </div>
</body>
</html>
123456789101112131415161718192021222324252627282930313233343536373839404142434445464748495051525354555657585960616263646566676869707172737475767778798081828384858687888990919293949596979899100101102103104105106107108109110111112113114115116117118119120121122123124125126127128129130131132133134135136137138139140141142143144145146147148149150151152153154155156157158159160161162163164165166167168169170171172173174175176177178179180181182183184185186187188189190191192193194195196197198199200201202203204205206207
```



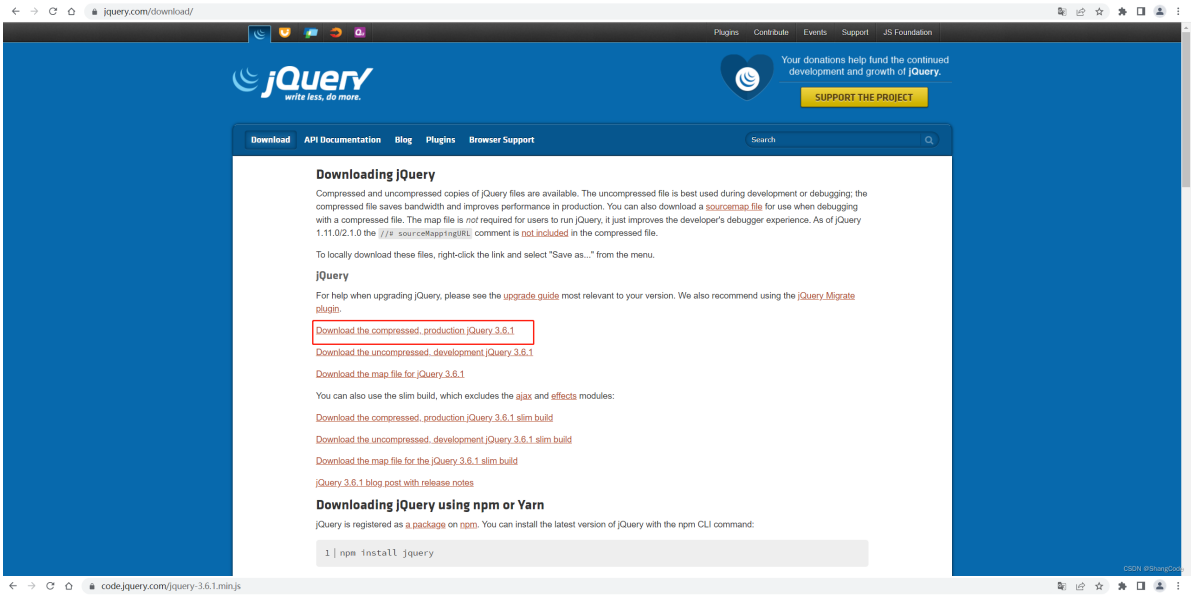
5.8 Bootstrap实现动态效果

依赖:

- JQuery
- Javascript

> JQuery 是 Javascript 的类库

下载JQuery: https://jquery.com/download/



因为我的项目在服务器中,所以我可以在服务器中使用 `wget` 命令直接下载

放在 `static/js` 下

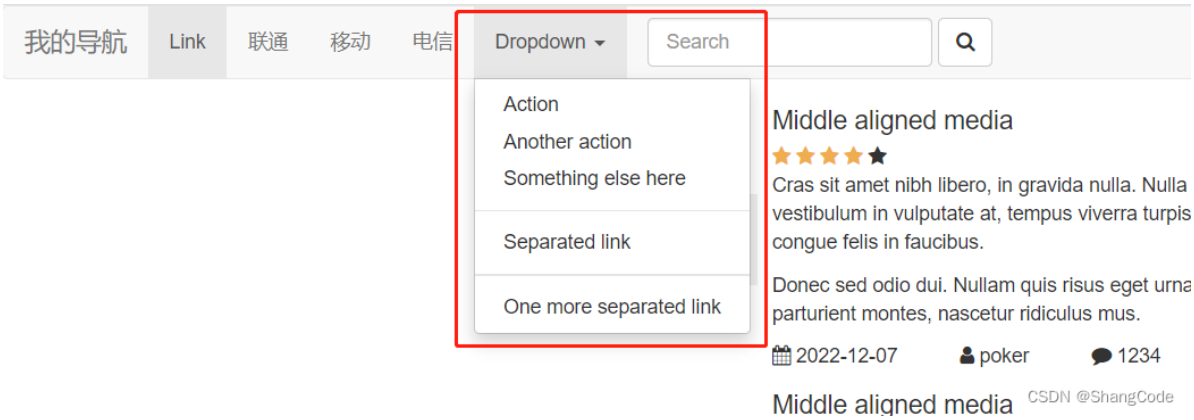
```
[root@hecs-33592 js]# pwd
/root/python/bootstrap/static/js
[root@hecs-33592 js]# wget https://code.jquery.com/jquery-3.6.1.min.js
--2022-12-07 14:08:39--  https://code.jquery.com/jquery-3.6.1.min.js
Resolving code.jquery.com (code.jquery.com)... 69.16.175.42, 69.16.175.10,
2001:4de0:ac18::1:a:1a, ...
Connecting to code.jquery.com (code.jquery.com)|69.16.175.42|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 89664 (88K) [application/javascript]
Saving to: ‘jquery-3.6.1.min.js’

100%
[==========================================================================================
==========================================================================================
=======>] 89,664       209KB/s   in 0.4s

2022-12-07 14:08:40 (209 KB/s) - ‘jquery-3.6.1.min.js’ saved [89664/89664]
```

以之前的导航页面做演示

> 放在 `body` 标签中的最下面

```
<body>
...
    <script src="static/js/jquery-3.6.1.min.js"></script>
    <script src="static/plugins/bootstrap-3.4.1/js/bootstrap.min.js"></script>
</body>
```

浏览器刷新访问



接下来看一下Javascript

地址: https://v3.bootcss.com/javascript/

JavaScript 插件

jQuery 插件为 Bootstrap 的组件赋予了“生命”。可以简单地一次性引入所有插件，或者逐个引入到你的页面中。

概览

单个还是全部引入

JavaScript 插件可以单个引入（使用 Bootstrap 提供的单个 `*.js` 文件），或者一次性全部引入（使用 `bootstrap.js` 或压缩版的 `bootstrap.min.js`）。

> 建议使用压缩版的 JavaScript 文件
> `bootstrap.js` 和 `bootstrap.min.js` 都包含了所有插件，你在使用时，只需选择一个引入页面就可以了。

> 插件之间的依赖关系

动态实例

点击下面的按钮即可通过 JavaScript 启动一个模态框。此模态框将从上到下、逐渐浮现到页面前。

EXAMPLE

Launch demo modal

Copy

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary btn-lg" data-toggle="modal" data-target="#myModal">
  Launch demo modal
</button>

<!-- Modal -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-
hidden="true">&times;</span></button>
        <h4 class="modal-title" id="myModalLabel">Modal title</h4>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>
```

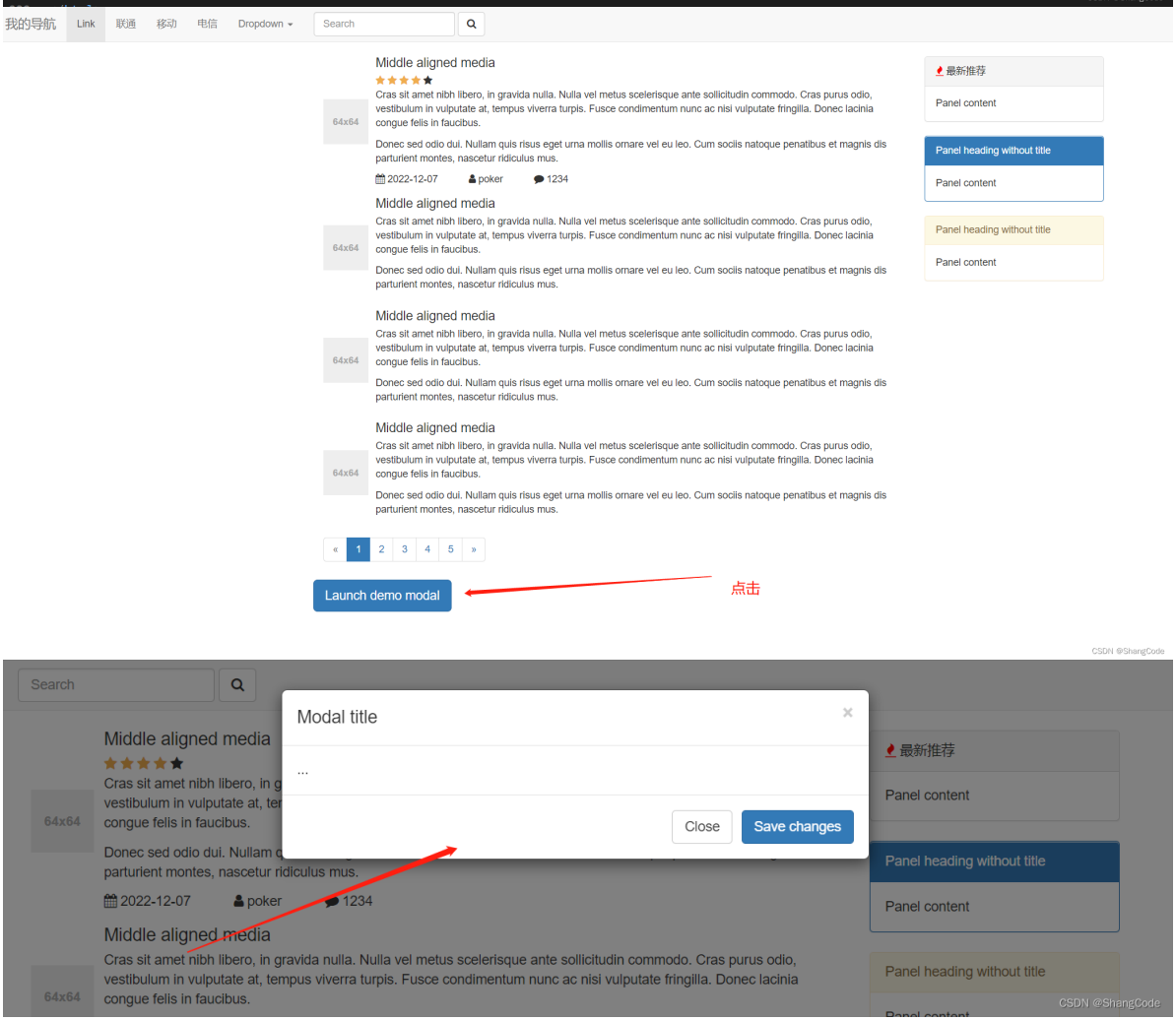

> 增强模态框的可访问性
> 务必为 `.modal` 添加 `role="dialog"` 和 `aria-labelledby="..."` 属性，用于指向模态框的标题栏；为 `.modal-dialog` 添加 `aria-hidden="true"` 属性。
>
> 另外，你还应该通过 `aria-describedby` 属性为模态框 `.modal` 添加描述性信息。

> 嵌入 YouTube 视频（天朝无用）


```
<div class="container">
    <!-- Button trigger modal -->
    <button type="button" class="btn btn-primary btn-lg" data-toggle="modal"
data-target="#myModal">
        Launch demo modal
    </button>

    <!-- Modal -->
    <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel">
        <div class="modal-dialog" role="document">
            <div class="modal-content">
                <div class="modal-header">
                    <button type="button" class="close" data-dismiss="modal"
aria-label="Close"><span
                            aria-hidden="true">&times;</span></button>
                    <h4 class="modal-title" id="myModalLabel">Modal title</h4>
                </div>
```

```
                <div class="modal-body">
                    ...
                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
                    <button type="button" class="btn btn-primary">Save changes</button>
                </div>
            </div>
        </div>
    </div>
</div>
```

```
    </div>

    <div class="container">
        <!-- Button trigger modal -->
        <button type="button" class="btn btn-primary btn-lg" data-toggle="modal" data-target="#myModal">
            Launch demo modal
        </button>

        <!-- Modal -->
        <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">
            <div class="modal-dialog" role="document">
                <div class="modal-content">
                    <div class="modal-header">
                        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
                                aria-hidden="true">&times;</span></button>
                        <h4 class="modal-title" id="myModalLabel">Modal title</h4>
                    </div>
                    <div class="modal-body">
                        ...
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
                        <button type="button" class="btn btn-primary">Save changes</button>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <script src="static/js/jquery-3.6.1.min.js 打开链接 (ctrl + 单击)
    <script src="static/plugins/bootstrap-3.4.1/js/bootstrap.min.js"></script>

</body>
```



> 可以观察一下这个是怎么实现点击跳出窗口的

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary btn-lg" data-toggle="modal" data-target="#myModal">
    Launch demo modal
</button>
1234
```

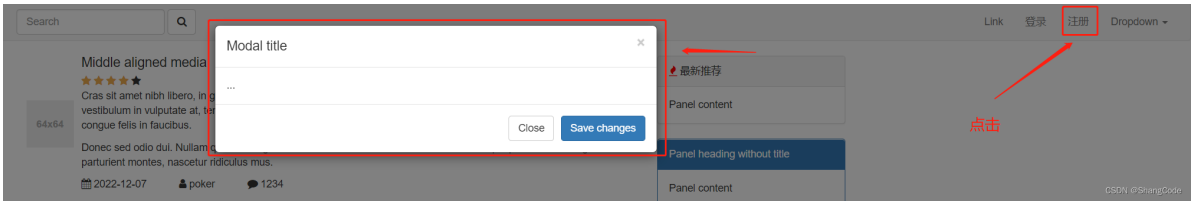
上面的代码中,`data-toggle="modal" data-target="#myModal"`,点击按钮后会寻找带有`id=myModal`的标签

```
<!-- Modal -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
                        aria-hidden="true">&times;</span></button>
                <h4 class="modal-title" id="myModalLabel">Modal title</h4>
            </div>
            <div class="modal-body">
                ...
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
                <button type="button" class="btn btn-primary">Save changes</button>
            </div>
        </div>
    </div>
</div>
```

因此我们可以知道,使用其他的标签一样可以触发点击跳转

> 将右上角的注册按钮设置为点击跳转窗口

```
<li><a href="#" data-toggle="modal" data-target="#myModal">注册</a></li>
```



6. Javascript

- JavaScript 是一门编程语言，浏览器就是 JacaScript 语言的解释器。
- DOM 和 BOM

```
相当于编程语言的内置模块
例如：python中的re、random、time、json模块等
```

- JQuery

```
相当于是编程语言的第三方模块
例如：requests openpyxl
```

> **意义:** 实现动态效果

先准备基础目录,拷贝之前的`bootstrap`目录

```
[root@hecs-33592 python]# cp -r bootstrap javascript
```

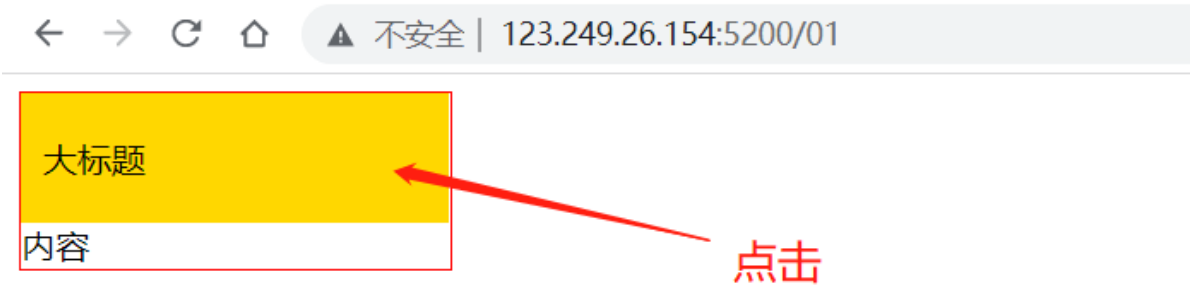
删除`javascript`中无用的`html`文件

```
[root@hecs-33592 python]# cd javascript/
[root@hecs-33592 javascript]# ls
main.py  static  templates
[root@hecs-33592 javascript]# cd templates/
[root@hecs-33592 templates]# ls
01.html  02.html  03.html  04.html  05.html
[root@hecs-33592 templates]# rm -rf ./*
```

首先编写一个小样例

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        .menus {
            width: 200px;
            border: 1px solid red;
        }

        .menus .header {
            background-color: gold;
            padding: 20px 10px;
        }

    </style>
</head>
<body>
    <div class="menus">
        <div class="header" onclick="myFunc()">大标题</div>
        <div class="item">内容</div>
    </div>

    <script type="text/javascript">
        function myFunc() {
            alert("hello")
        }
    </script>

</body>
</html>
```

访问测试



CSDN @ShangCode

跳出对话框

CSDN @ShangCode

更改,使用 `confirm`

```
<script type="text/javascript">
    function myFunc() {
        // alert("hello")
        confirm("是否要继续?")
    }
</script>
123456
```

浏览器刷新访问

CSDN @ShangCode

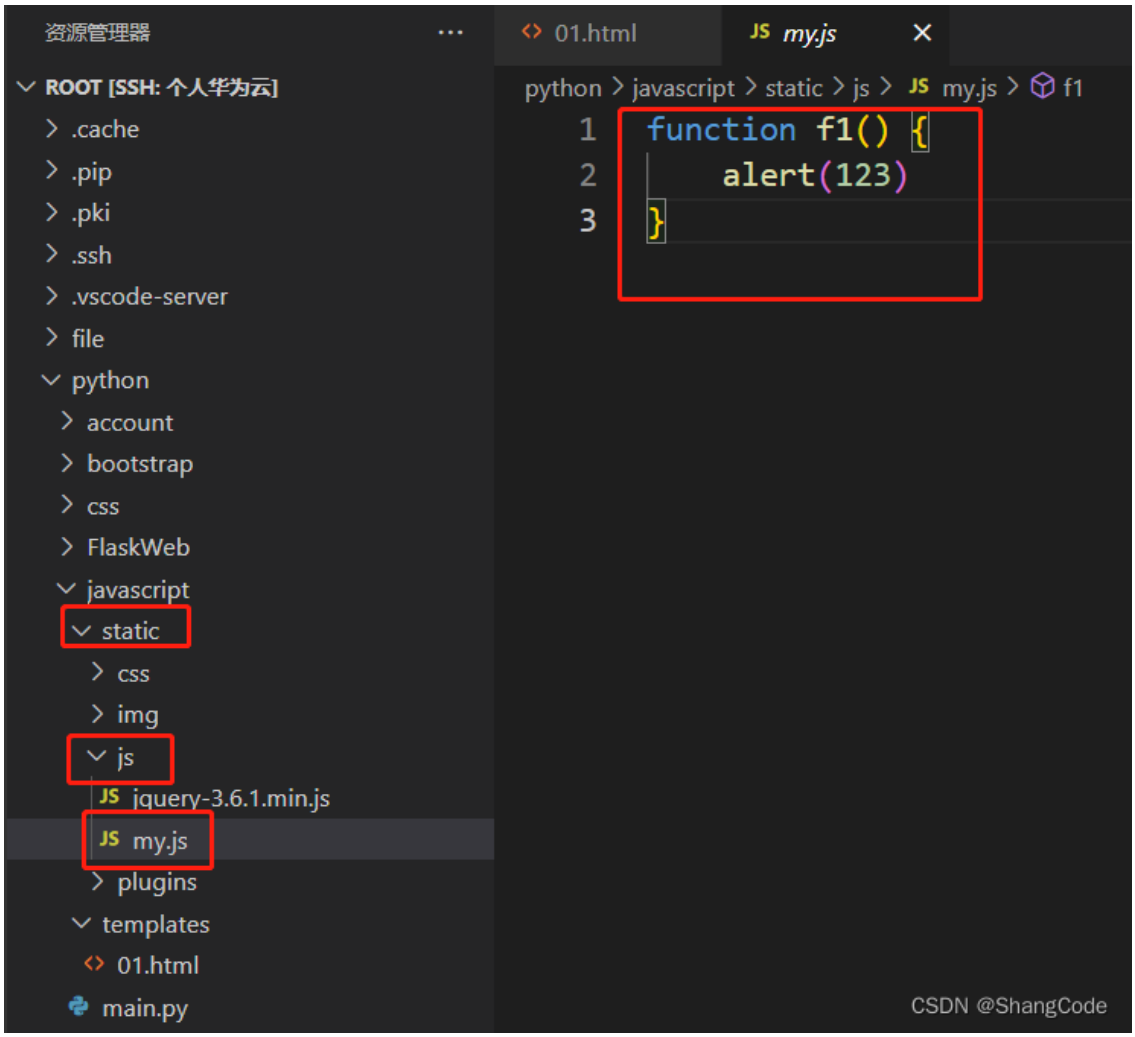
6.1 代码位置

js代码的存在形式:

- 在当前HTML文件中
  - head中
  - body中(推荐)

> 注意: 如果写在了 `head` 中, `body` 的 `html` 代码不会执行,会先执行 `head` 中的 `javascript` 之后,才会显示 `html` 代码

- 在其他js文件中,导入使用(一般放在 body 中)




```
<body>
    <script src="static/js/my.js"></script>
</body>
123
```

6.2 注释

- HTML的注释

```
<!-- 注释内容 -->
```

- CSS的注释

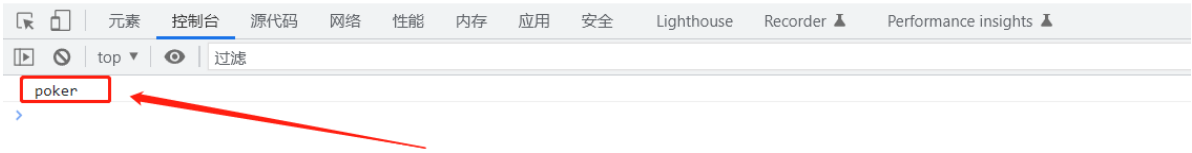
```
/* 注释内容 */
```

- Javascript的注释

```
// 注释内容

/* 注释内容 */
```

6.3 变量

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <script type="text/javascript">
        var name = "poker";
        console.log(name);   //打印变量
    </script>
</body>
</html>·
```



6.3.1 字符串类型

```
//声明
var name = "helloworld";
var name = String("helloworld");
```

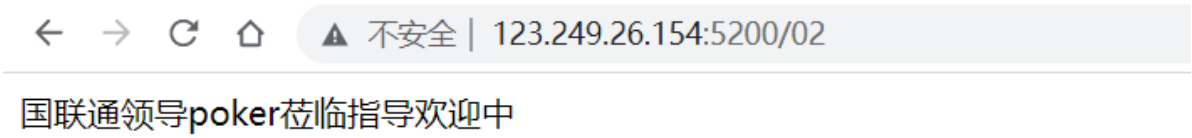
常见功能

```
var name = "中国联通"
var v1 = name.length;
var v2 = name[0];
var v3 = name.trim();           //去除空白
var v4 = name.substring(0,2)    //切片, 前取后不取
12345
```

案例: 跑马灯

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>

<body>

    <div id="txt">欢迎中国联通领导poker莅临指导</div>

    <script type="text/javascript">

        function show() {
            //1.去HTML中找到某个标签并获取他的内容 (DOM)
```

```
            var tag = document.getElementById("txt");
            var dataString = tag.innerText;

            //2.动态起来,把文本中的第一个字符放在字符串的最后面
            var firstChar = dataString[0];
            var otherString = dataString.substring(1, dataString.length);
            var newText = otherString + firstChar;

            //3.在HTML标签中更新内容
            tag.innerText = newText;
        }

        //Javascript中的定时器
        //每秒钟执行这个show函数
        setInterval(show, 1000);    //毫秒
    </script>
</body>
</html>
```



国联通领导poker莅临指导欢迎中

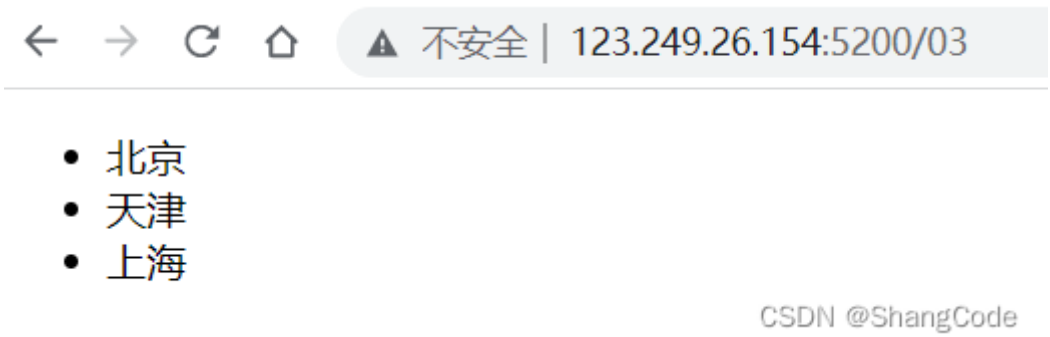
6.3.2 数组

```
var v1 = [11,22,33,44];
var v2 = Array([11,22,33,44]);

//操作
var v1 = [11,22,33,44];

v1[1]
v1[0] = "poker"

//追加
v1.push("联通");           //尾部追加 [11,22,33,44,"联通"]
v1.unshift("联通");        //头部追加 ["联通",11,22,33,44]
v1.splice(索引,0,元素);
v1.splice(1,0,"中国");     //指定位置追加 [11,"中国",22,33,44]

//删除
v1.pop();                 //尾部删除
v1.shift();               //头部删除
v1.splice(索引位置,1);
v1.splice(2,1);           //索引为 2 的元素删除 [11,22,44]



//循环
```

```
var v1 = [11,22,33,44];
//循环的是索引
for(var index in v1){
    //data=v1[index]
    ...
}


for(var i=0; i<v1.length; i++){
    ...
}
```

案例: 动态数据

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>

    <ul id="city">
        <!-- <li>北京</li>
        <li>天津</li>
        <li>上海</li> -->
    </ul>

    <script type="text/javascript">
        var cityList = ["北京","天津","上海"];
        for(var idx in cityList) {
            var text = cityList[idx];

            //创建 <li></li> 标签
            var tag = document.createElement("li");
            //在 li 标签中写入内容
            tag.innerText = text;

            //添加到 id=city 那个标签的里面 DOM
            var parentTag = document.getElementById("city");
            parentTag.appendChild(tag);
        }
    </script>
</body>
</html>
```



6.3.3 对象(字典)

```
info = {
    "name":"poker",
    "age":18,
}

info = {
    name:"poker",
    age:18
}

info.age;
info.name = "toker"

info["age"]
info["name"] = "toker";

delete info["age"]

//循环
for(var key in info){
    //key值 data=info[key]
    ...
}

```

案例: 动态表格

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>

<body>
    <table border="1">
        <thead>
            <tr>
                <th>ID</th>
                <th>姓名</th>
                <th>年龄</th>
            </tr>
        </thead>
        <tbody id="body">
            <tr>
                <!-- <td>1</td>
                <td>poker</td>
                <td>25</td> -->
            </tr>
        </tbody>
    </table>

    <script type="text/javascript">
```

```
28
29          var dataList = [
30              { id: 1, name: "poker", age: 25 },
31              { id: 1, name: "poker", age: 25 },
32              { id: 1, name: "poker", age: 25 },
33              { id: 1, name: "poker", age: 25 },
34              { id: 1, name: "poker", age: 25 },
35              { id: 1, name: "poker", age: 25 },
36          ];
37
38          for (var idx in dataList) {
39              var info = dataList[idx]
40              //1.创建 tr 标签
41              var tr = document.createElement("tr")
42              for (var key in info) {
43                  var text = info[key];
44                  //2.创建 td 标签
45                  var td = document.createElement("td");
46                  td.innerText = text;
47                  tr.appendChild(td);
48              }
49              //3. 追加数据
50              var bodyTag = document.getElementById("body");
51              bodyTag.appendChild(tr);
52          }
53      </script>
54  </body>
55  </html>
```

ID	姓名	年龄
1	poker	25
1	poker	25
1	poker	25
1	poker	25
1	poker	25
1	poker	25

6.4 条件语句

```
if (条件) {
    ...
}else{
    ...
}

if (条件) {
    ...
else if (条件){
    ...
}else{
    ...
}
```

6.5 函数

```
function func(){
    ...
}

//执行
func()
```

6.6 DOM

> DOM 是一个模块,模块可以对HTML页面中的标签进行操作

```
//根据 ID 获取标签
var tag = doucment.getElementById("xx");

//获取标签中的文本
tag.innerText

//修改标签中的文本
tag.innerText = "hhhhhhh";

// 创建标签
var tag = document.createElement("div")

// 标签写内容
tag.innerText = "xxx"
```

如标题 6.3.2 中的案例

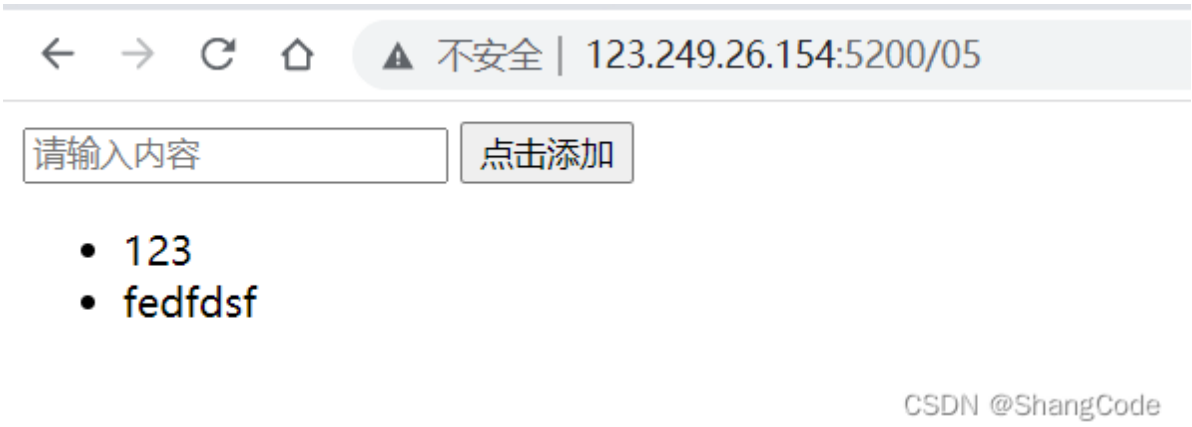
```
<body>

    <ul id="city">
        <!-- <li>北京</li>
        <li>天津</li>
        <li>上海</li> -->
    </ul>

    <script type="text/javascript">
        var cityList = ["北京","天津","上海"];
```

```
        for(var idx in cityList) {
            var text = cityList[idx];

            //创建 <li></li> 标签
            var tag = document.createElement("li");
            //在 li 标签中写入内容
            tag.innerText = text;

            //添加到 id=city 那个标签的里面 DOM
            var parentTag = document.getElementById("city");
            parentTag.appendChild(tag);
        }
    </script>
</body>
12345678910111213141516171819202122232425
```

事件的绑定

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>

<body>

    <input type="text" placeholder="请输入内容" id="content">
    <input type="button" value="点击添加" onclick="addCityInfo()">

    <ul id="city">
    </ul>

    <script type="text/javascript">
        function addCityInfo() {
            //1.找到标签
            var userContent = document.getElementById("content");
            //2.获取input中用户输入的内容
            var newString = userContent.value;
            //判断用户输入是否为空
            if (newString.length > 0) {
                //3.创建 li 标签,传入用户输入的内容
                var newTag = document.createElement("li");
                newTag.innerText = newString;
                //4.标签添加到 ul 中
                var parentTag = document.getElementById("city");
                parentTag.appendChild(newTag);
                //5.将 input text 内容清空
                userContent.value = "";
            }else{
                alert("输入不能为空!")
            }
        }
    </script>
</body>
```

```
39    </html>
```



> 还有很多的 `DOM` 操作没有介绍,我们后面会使用 `JQuery` 来实现 `DOM` 的功能,所以这里的内容了解即可

6.7知识总结

'

- 数据类型

```
常见的数据类型：int bool str list tuple dict set float None
    - 哪些转化弄成布尔类型为 False: 空 None 0
    - 可变和不可变划分，可变的有哪些:list set dict
    - 可哈希和不可哈希，不可哈希的有哪些：list set dict
    - 字典的键/集合的元素，必须是可哈希的类型( list set dict 不能做字典的键和集合元素 )

主要数据类型：
    - str
        - 独有功能：upper / lowwer / startswitch / split / strip / join
        注意：str 不可变 不会对原字符串进行修改 新的内容
        - 公共功能：len / 索引 / 切片 / for循环 / 判断是否包含
    - list
        - 独有功能：append insert remove pop...
        注意：list 可变 功能很多都是对原数据操作
        - 公共功能：len / 索引 / 切片 / for循环 / 判断是否包含
    - dict
        - 独有功能：get keys items values
        - 公共功能：len / 索引 / 切片 / for循环 / 判断是否包含
```

- 函数编程

```
函数的基础指数：
    - 定义
    - 参数、概念：位置传参 / 关键字传参 / 参数默认值 / 动态参数 *args **kwargs
    - 返回值
        - 函数中一旦遇到 return 九立即返回，后续代码不再执行
函数的进阶：
    - python 中是为函数为作用域
    - 全局变量和局部变量、规范：全局变量( 大写 ) 、 局部变量( 小写 )
```

```
    - 在局部变量中可以使用 global 关键字，global 的作用？引用全局的那个变量 ( 不是在局部创建 )
内置函数：
    - bin/hex/odc/max/min/dicmod/sorted/open 文件操作

文件操作：
    - 基本操作：打开 操作 关闭 为了防止忘记关闭文件 `with`
    - 打开文件时有模式：
        - r / rb 读            【文件不存在，报错】                【文件不存在，报错】
        - w / wb 写( 清空 )    【文件不存在 自动创建】             【文件不存在，报错】
        - a / ab 追加          【文件不存在 自动创建】             【文件不存在，报错】

        注意：os.mkdirs / os,path.exsits 是否存在 不存在 创建新建目录
```

- 模块

```
模块的分类：
    - 自定义模块
        - os.path 导入模块时 python 内部都回去那个目录先
        - 自己写 py 文件时，不要与python内置模块同名
        - import / from xx import xxx
    - 内置模块：time datatime json re random os...
    - 第三方模块：request openpyxl python-docx flask bs4
查看当前目录下的所有文件：os.listdir / os.walk
关于时间模块：时间戳 / datetime格式 / 字符串，三种时间格式可以互相转化。
关于 Json 模块：
    - JSON 本质是字符串，有一些自己格式的要求，例如：无元组 / 无单引
关于 re 正则模块：
    - 正则：\d \w
    - 贪婪匹配和非贪婪匹配( 默认 )，想让他不贪婪 个数后面？
    - re.search / re.match / re.findall
第三方模块：都有哪些可以让我们安装第三方模块。
    - pip 安装工具‘
    - 源码
    - wheel 包

```

- 面向对象

```
目标：不是为了用面向对象编程
面向对象三大特性：封装 继承  多态
```

- 前端开发

```
- 前端知识分为三部分：
    - HTML 标签具有模式特点
    - CSS  修改标签的特点
    - JacaScript  动态
- HTML 标签
    - div / span / a / img / input / form / table /ul...
    - 块级和行内标签，例如：div span 默认谁是块级标签？ div
        注意：CSS 样式，发现行内标签设置高度，宽度，内边距，外边距都是无效
    - form 表单 + input/selecet/textarea 数据框
        - action 递交地址
```

```
            - method 提交方式
            - form 标签中一个 submit 标签
            - 内部标签都需要设置name属性
    - CSS 样式
        - 局部一定会用到的样式：div + float( 脱离文档流，clear;both;clearfix )
        - 高度和狂赌
        - 边距
            - 内边距 padding
            - 外边距 margin
        - 字体大小 / 颜色
        - 边框
        - 背景颜色
        - hover 鼠标放上去就会触发 CSS 的样式。

```

7. JQuery

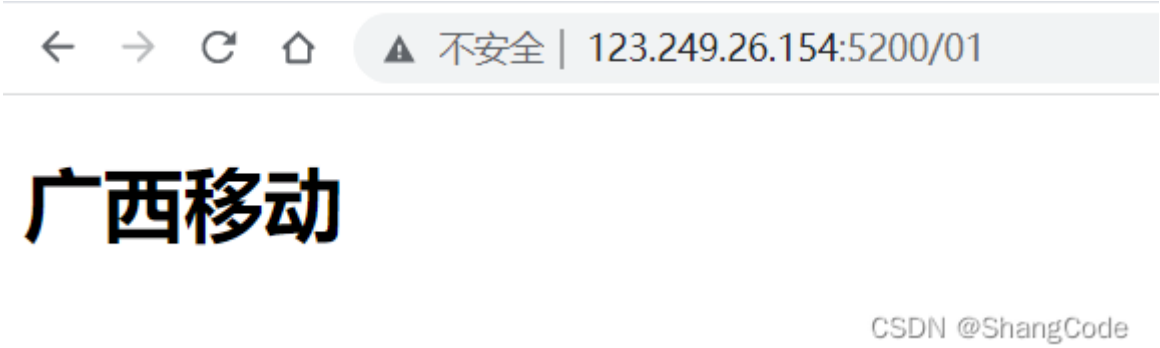
JQuery是一个JavaScript的第三方模块(第三方类库)

- 基于JQuery自己开发一个功能
- 现成的工具依赖JQuery, 例如: Bootstrap动态效果

JQuery的安装方式参考本文的 **5.8 Bootstrap实现动态效果**

7.1 快速上手

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>

    <h1 id="txt">中国联通</h1>

    <script src="static/js/jquery-3.6.1.min.js"></script>
    <script type="text/javascript">
        //使用JQuery修改内容
        $("#txt").text("广西移动");
    </script>

</body>
</html>
```



7.2 寻找标签(直接)

7.2.1 ID选择器

```
<h1 id="txt">中国联通</h1>
<h1>中国联通</h1>
<h1>中国联通</h1>
```

JQuery操作:

```
$("#txt")
```

7.2.2 样式选择器

```
<h1 class="c1">中国联通</h1>
<h1 class="c2">中国联通</h1>
<h1 class="c3">中国联通</h1>
```

JQuery操作:

```
$(".c1")
```

7.2.3 标签选择器

```
<h1 class="c1">中国联通</h1>
<h1 class="c2">中国联通</h1>
<h1 class="c3">中国联通</h1>
```

JQuery操作:

```
$("h1")
```

7.2.4 层级选择器

```
<div class="c1">
    <div class="c2">
        <h1>123</h1>
    </div>
</div>
```

JQuery操作:

```
$(".c1 .c2 h1")
```

7.2.5 多选择器

```
<div class="c1">
    <div class="c2">
        <h1>123</h1>
    </div>
</div>
<div class="c3">
    <div class="c4">
        <h1>123</h1>
        <li>456</li>
    </div>
</div>
```

JQuery操作:

```
$("#c1,#c2,li")
```

7.2.6 属性选择器

```
<input type="text" name="n1" />
<input type="text" name="n2" />
<input type="text" name="n3" />
```

JQuery操作:

```
$("input[name='n1']")
```

7.3 寻找标签(间接)

7.3.1 找到兄弟

```
<div>
    <div>北京</div>
    <div id="c1">上海</div>
    <div>深圳</div>
    <div>广州</div>
</div>
```

JQuery操作:

```
$("#c1").prev()          //上一个 <div>北京</div>
$("#c1")                 // <div id="c1">上海</div>
$("#c1").next()          //下一个 <div>深圳</div>
$("#c1").next().next()   //下一个的下一个 <div>广州</div>
$("#c1").siblings()      //所有的兄弟
```

7.3.2 找父子

```
<div>
    <div>
        <div>北京</div>
        <div id="c1">
            <div>浦东新区</div>
            <div class="p10">静安区</div>
            <div>奉贤区</div>
```

```
            </div>
            <div>深圳</div>
            <div>广州</div>
        </div>
        <div>
            <div>北京</div>
            <div>上海</div>
            <div>深圳</div>
            <div>广州</div>
        </div>
    </div>
```

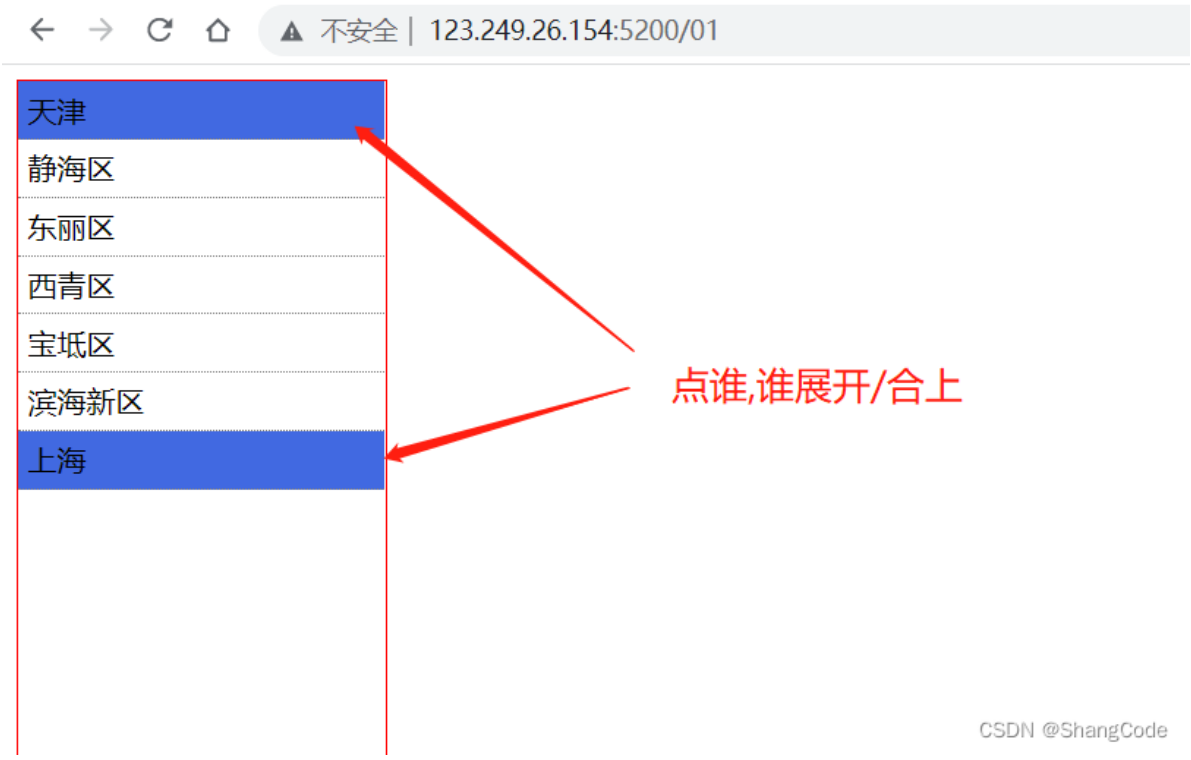
JQuery操作:

```
$("#c1").parent()           //父亲
$("#c1").parent().parent()  //父亲的父亲

$("#c1").children()         //所有的儿子
$("#c1").children(".p10")   //所有的儿子中寻找class=p10

$("#c1").find(".p10")       //所有的子孙中寻找class=p10
$("#c1").children("div")    //所有的儿子中寻找标签 div
```

案例: 菜单的切换

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <style>
        .menus {
            width: 200px;
            height: 1000px;
            border: 1px solid red;
        }

        .menus .header {
            background-color: royalblue;
            padding: 5px 5px;
            border-bottom: 1px dotted gray;
            cursor: pointer;
        }

        .menus .content a {
            display: block;
            padding: 5px 5px;
            border-bottom: 1px dotted gray;
        }

        .hide {
            display: none;
        }
```

```
        </style>
    </head>

    <body>

        <div class="menus">
            <div class="item">
                <div class="header" onclick="clickMe(this);">天津</div>
                <div class="content hide">
                    <a>静海区</a>
                    <a>东丽区</a>
                    <a>西青区</a>
                    <a>宝坻区</a>
                    <a>滨海新区</a>
                </div>
            </div>
            <div class="item">
                <div class="header" onclick="clickMe(this);">上海</div>
                <div class="content hide">
                    <a>静安区</a>
                    <a>奉贤区</a>
                    <a>浦东新区</a>
                    <a>徐汇区</a>
                    <a>青浦区</a>
                </div>
            </div>
        </div>

        <script src="static/js/jquery-3.6.1.min.js"></script>
        <script type="text/javascript">
            function clickMe(self) {
                var hasHide = $(self).next().hasClass("hide");
                if (hasHide) {
                    $(self).next().removeClass("hide");
                } else {
                    $(self).next().addClass("hide");
                }
            }
        </script>
    </body>
    </html>
```

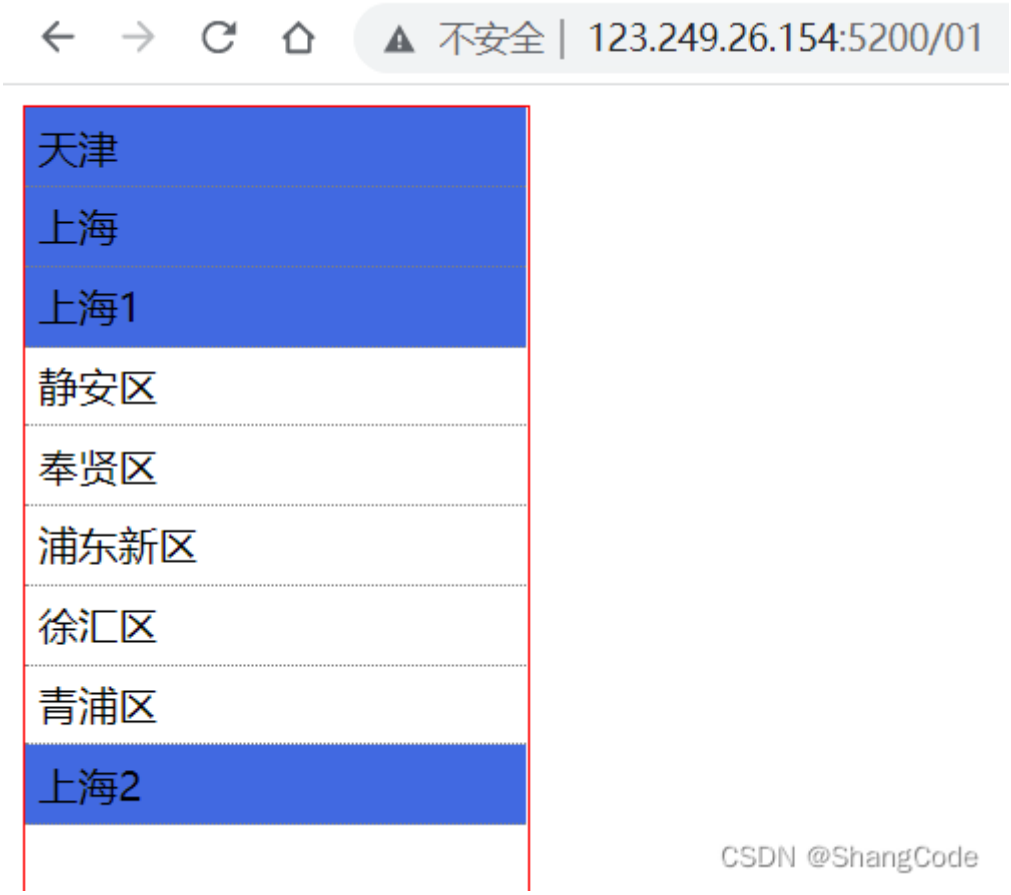


> 功能升级: 只允许有一个是展开的

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <style>
        .menus {
            width: 200px;
            height: 1000px;
            border: 1px solid red;
        }

        .menus .header {
            background-color: royalblue;
            padding: 5px 5px;
            border-bottom: 1px dotted gray;
            cursor: pointer;
        }

        .menus .content a {
            display: block;
            padding: 5px 5px;
            border-bottom: 1px dotted gray;
        }

        .hide {
            display: none;
        }
    </style>
</head>
```

```
33
34  <body>
35
36      <div class="menus">
37          <div class="item">
38              <div class="header" onclick="clickMe(this);">天津</div>
39              <div class="content">
40                  <a>静海区</a>
41                  <a>东丽区</a>
42                  <a>西青区</a>
43                  <a>宝坻区</a>
44                  <a>滨海新区</a>
45              </div>
46          </div>
47          <div class="item">
48              <div class="header" onclick="clickMe(this);">上海</div>
49              <div class="content hide">
50                  <a>静安区</a>
51                  <a>奉贤区</a>
52                  <a>浦东新区</a>
53                  <a>徐汇区</a>
54                  <a>青浦区</a>
55              </div>
56          </div>
57          <div class="item">
58              <div class="header" onclick="clickMe(this);">上海1</div>
59              <div class="content hide">
60                  <a>静安区</a>
61                  <a>奉贤区</a>
62                  <a>浦东新区</a>
63                  <a>徐汇区</a>
64                  <a>青浦区</a>
65              </div>
66          </div>
67          <div class="item">
68              <div class="header" onclick="clickMe(this);">上海2</div>
69              <div class="content hide">
70                  <a>静安区</a>
71                  <a>奉贤区</a>
72                  <a>浦东新区</a>
73                  <a>徐汇区</a>
74                  <a>青浦区</a>
75              </div>
76          </div>
77      </div>
78
79      <script src="static/js/jquery-3.6.1.min.js"></script>
80      <script type="text/javascript">
81          function clickMe(self) {
82              //1.让菜单展示出来
83              $(self).next().removeClass("hide");
84
85              //2.找父亲,父亲的所有兄弟,再去每个兄弟的子孙中寻找 class="content", 添加 hide
    样式
86              $(self).parent().siblings().find(".content").addClass("hide");
87          }
```

```
88          </script>
89      </body>
90      </html>
```



7.4 值的操作

```
1   <div id="c1">内容</div>
2   <input type="text " id="c2"/>
3   12
```

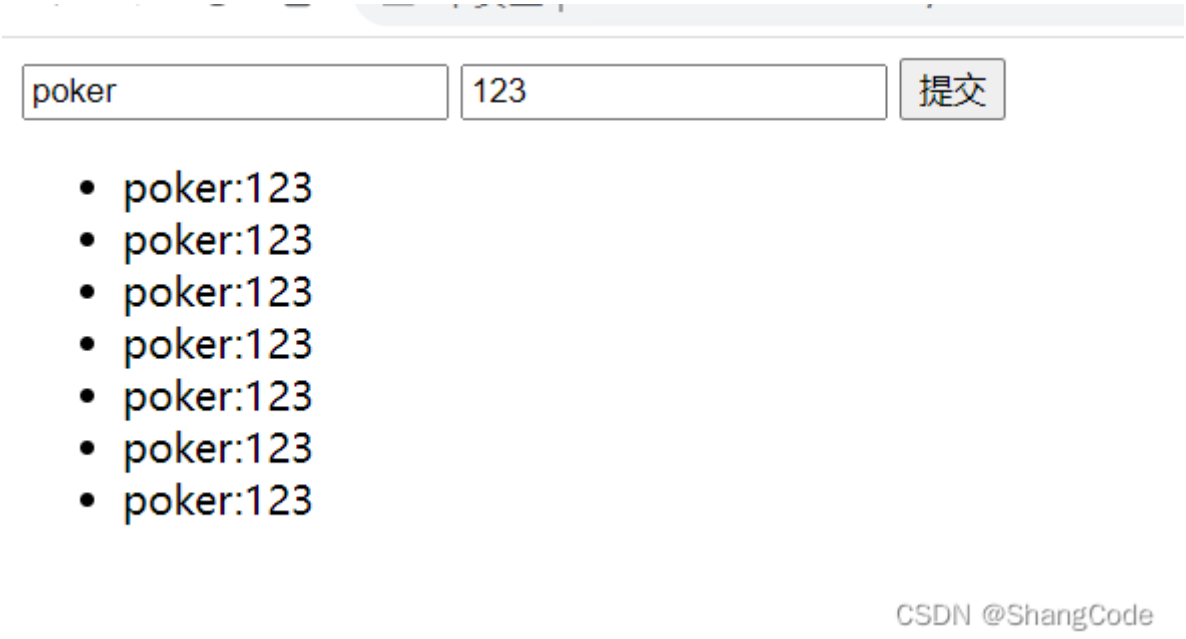
JQuery操作:

```
1   $("#c1").text()              //获取文本内容
2   $("#c1").text("abc")         //设置文本内容
3
4   $("#c2").val()               //获取用户输入的值
5   $("#c2").val("嘿嘿嘿")      //设置值
```

案例: 动态创建数据

```
1   <!DOCTYPE html>
2   <html lang="en">
3
4   <head>
5       <meta charset="UTF-8">
6       <title>Document</title>
7   </head>
8
9   <body>
10
```

```
11      <input type="text" id="txtUser" placeholder="用户名">
12      <input type="text" id="txtEmail" placeholder="密码">
13      <input type="button" value="提交" onclick="getInfo()">
14
15      <ul id="view">
16      </ul>
17
18      <script src="static/js/jquery-3.6.1.min.js"></script>
19      <script>
20
21          function getInfo() {
22              //1.获取用户输入的用户名与密码
23              var username = $("#txtUser").val();
24              var email = $("#txtEmail").val();
25
26              dataString = username + ":" + email
27
28              //2.创建li标签, 在li内部写入内容
29              var newLi = $("<li>").text(dataString);
30
31              //3.把新创建的li标签添加到ul里面
32              $("#view").append(newLi);
33          }
34
35      </script>
36  </body>
37  </html>
```



7.5 事件

```
1   <body>
2
3       <ul>
4           <li>百度</li>
5           <li>谷歌</li>
6           <li>搜狗</li>
7       </ul>
```

```
    <script src="static/js/jquery-3.6.1.min.js"></script>
    <script>
        $("li").click(function(){
            // 点击li标签时,自动执行这和函数
            // $(this) 当前你点击的是哪个标签
        });
    </script>
</body>
```

在JQuery可以删除指定的标签

```
<script src="static/js/jquery-3.6.1.min.js"></script>
<script>
    $("li").click(function(){
        // 点击li标签时,自动执行这和函数
        // $(this) 当前你点击的是哪个标签
        $(this).remove();
    });
</script>
```

当页面框架加载完成之后执行代码

案例: 表格操作

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>

<body>

    <table border="1">
        <thead>
            <tr>
                <th>ID</th>
                <th>姓名</th>
                <th>年龄</th>
            </tr>
        </thead>
        <tbody id="body">
            <tr>
                <td>1</td>
                <td>poker</td>
                <td>
                    <input type="button" value="删除" class="delete" />
                </td>
            </tr>
            <tr>
                <td>1</td>
                <td>poker</td>
                <td>
                    <input type="button" value="删除" class="delete" />
```

```
                </td>
            </tr>
            <tr>
                <td>1</td>
                <td>poker</td>
                <td>
                    <input type="button" value="删除" class="delete" />
                </td>
            </tr>
            <tr>
                <td>1</td>
                <td>poker</td>
                <td>
                    <input type="button" value="删除" class="delete" />
                </td>
            </tr>
        </tbody>
    </table>

    <script src="static/js/jquery-3.6.1.min.js"></script>
    <script>
        $(
            function () {
                $(".delete").click(function () {
                    $(this).parent().parent().remove();
                });
            }
        )
    </script>
</body>
</html>
```

8. 前端整合

- HTML
- CSS
- JavaScript、jQuery
- BootStrap ）（动态效果依赖 jQuery ）

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <!-- 开发版本 -->
    <link rel="stylesheet" href="static/plugins/bootstrap-3.4.1/css/bootstrap.css">
    <link rel="stylesheet" href="static/plugins/font-awesome-4.7.0/css/font-awesome.css">

    <!-- 生产版本 -->
    <!-- <link rel="stylesheet" href="static/plugins/bootstrap-3.4.1/css/bootstrap.min.css"> -->

    <style>
        /* 去除导航栏圆角 */
        .navbar {
            border-radius: 0;
        }
    </style>

</head>

<body>


    <nav class="navbar navbar-inverse">
        <div class="container-fluid">
            <!-- Brand and toggle get grouped for better mobile display -->
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
                    data-target="#bs-example-navbar-collapse-9" aria-expanded="false">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="#">Brand</a>
            </div>

            <!-- Collect the nav links, forms, and other content for toggling -->
            <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-9">
                <ul class="nav navbar-nav">
                    <li class="active"><a href="#">Home</a></li>
                    <li><a href="#">Link</a></li>
                    <li><a href="#">Link</a></li>
                    <li class="dropdown">
                        <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true"
```

```
                            aria-expanded="false">Dropdown <span class="caret">
</span></a>
                        <ul class="dropdown-menu">
                            <li><a href="#">Action</a></li>
                            <li><a href="#">Another action</a></li>
                            <li><a href="#">Something else here</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">Separated link</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">One more separated link</a></li>
                        </ul>
                    </li>
                </ul>
                <ul class="nav navbar-nav navbar-right">
                    <li><a href="#">登录</a></li>
                    <li><a href="#">注册</a></li>
                    <li class="dropdown">
                        <a href="#" class="dropdown-toggle" data-
toggle="dropdown" role="button" aria-haspopup="true"
                            aria-expanded="false">poker <span class="caret">
</span></a>
                        <ul class="dropdown-menu">
                            <li><a href="#">个人资料</a></li>
                            <li><a href="#">我的账户</a></li>
                            <li><a href="#">个性设置</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">Separated link</a></li>
                        </ul>
                    </li>
                </ul>
            </div><!-- /.navbar-collapse -->
        </div><!-- /.container-fluid -->
    </nav>



    <!-- Button trigger modal -->
    <button type="button" class="btn btn-primary btn-lg" data-toggle="modal"
data-target="#myModal">
        Launch demo modal
    </button>

    <!-- Modal -->
    <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel">
        <div class="modal-dialog" role="document">
            <div class="modal-content">
                <div class="modal-header">
                    <button type="button" class="close" data-dismiss="modal"
aria-label="Close"><span
                            aria-hidden="true">&times;</span></button>
                    <h4 class="modal-title" id="myModalLabel">Modal title</h4>
                </div>
                <div class="modal-body">
                    ...
                </div>
```

```
                <div class="modal-footer">
                    <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
                    <button type="button" class="btn btn-primary">Save changes</button>
                </div>
            </div>
        </div>
    </div>

    <div>
        <button type="button" class="btn btn-default" data-toggle="tooltip" data-placement="bottom"
            title="Tooltip on left">Tooltip on left</button>

        <button type="button" class="btn btn-default" data-toggle="tooltip" data-placement="top"
            title="Tooltip on top">Tooltip on top</button>

        <button type="button" class="btn btn-default" data-toggle="tooltip" data-placement="bottom"
            title="Tooltip on bottom">Tooltip on bottom</button>

        <button type="button" class="btn btn-default" data-toggle="tooltip" data-placement="right"
            title="Tooltip on right">Tooltip on right</button>
    </div>

    <div>
        <a id="tab" tabindex="0" class="btn btn-lg btn-danger" role="button" data-toggle="popover" data-trigger="focus" title="Dismissible popover" data-content="And here's some amazing content. It's very engaging. Right?">可消失的弹出框</a>
    </div>

    <script src="static/js/jquery-3.6.1.min.js"></script>
    <script src="static/plugins/bootstrap-3.4.1/js/bootstrap.js"></script>


    <div class="container" style="width: 800px;height:600px">
        <div id="carousel-example-generic" class="carousel slide" data-ride="carousel">
            <ol class="carousel-indicators">
              <li data-target="#carousel-example-generic" data-slide-to="0" class=""></li>
              <li data-target="#carousel-example-generic" data-slide-to="1" class=""></li>
              <li data-target="#carousel-example-generic" data-slide-to="2" class="active"></li>
            </ol>
            <div class="carousel-inner" role="listbox">
              <div class="item">
```

```
                  <img data-src="holder.js/900x500/auto/#777:#555/text:First slide" alt="First slide [900x500]" src="" data-holder-rendered="true">
                </div>
                <div class="item">
                  <img data-src="holder.js/900x500/auto/#666:#444/text:Second slide" alt="Second slide [900x500]" src="" data-holder-rendered="true">
                </div>
                <div class="item active">
                  <img data-src="holder.js/900x500/auto/#555:#333/text:Third slide" alt="Third slide [900x500]" src="" data-holder-rendered="true">
                </div>
              </div>
              <a class="left carousel-control" href="#carousel-example-generic" role="button" data-slide="prev">
                <span class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span>
                <span class="sr-only">Previous</span>
```

```
            </a>
            <a class="right carousel-control" href="#carousel-example-generic" role="button" data-slide="next">
              <span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>
              <span class="sr-only">Next</span>
            </a>
          </div>
    </div>

    <script>
        $(function () {
            $('[data-toggle="tooltip"]').tooltip()
        });
        $('#tab').popover('hide').popover({trigger: "click", placement: "bottom"});
    </script>
</body>
</html>
```

案例: 添加数据页面

> 人员信息录入功能,需要提供用户信息:
>
> - 用户名
> - 年龄
> - 薪资
> - 部门
> - 入职时间
>
> ...

对于时间的选择: 插件(datetimepicker),或者可以直接使用 `<input type="date" class="form-control" placeholder="入职时间">`

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <!-- 开发版本 -->
    <link rel="stylesheet" href="static/plugins/bootstrap-3.4.1/css/bootstrap.css">
    <link rel="stylesheet" href="static/plugins/font-awesome-4.7.0/css/font-awesome.css">

    <!-- 生产版本 -->
    <!-- <link rel="stylesheet" href="static/plugins/bootstrap-3.4.1/css/bootstrap.min.css"> -->

    <style>

    </style>
```

```
18
19  </head>
20
21  <body>
22
23
24      <div class="container">
25          <form class="form-horizontal" style="margin-top: 30px;">
26              <!-- 引入栅格系统 -->
27              <!-- 姓名和年龄 -->
28              <div class="row clearfix">
29                  <div class="col-xs-6">
30                      <div class="form-group">
31                          <label class="col-sm-2 control-label">姓名</label>
32                          <div class="col-sm-10">
33                              <input type="text" class="form-control"
    placeholder="姓名">
34                          </div>
35                      </div>
36                  </div>
37                  <div class="col-xs-6">
38                      <div class="form-group">
39                          <label class="col-sm-2 control-label">年龄</label>
40                          <div class="col-sm-10">
41                              <input type="text" class="form-control"
    placeholder="年龄">
42                          </div>
43                      </div>
44                  </div>
45              </div>
46
47              <!-- 部门和薪资 -->
48              <div class="row clearfix">
49                  <div class="col-xs-6">
50                      <div class="form-group">
51                          <label class="col-sm-2 control-label">部门</label>
52                          <div class="col-sm-10">
53                              <select class="form-control">
54                                  <option>IT部</option>
55                                  <option>运营部</option>
56                                  <option>销售部</option>
57                                  <option>售前部</option>
58                              </select>
59                          </div>
60                      </div>
61                  </div>
62                  <div class="col-xs-6">
63                      <div class="form-group">
64                          <label class="col-sm-2 control-label">薪资</label>
65                          <div class="col-sm-10">
66                              <input type="text" class="form-control"
    placeholder="薪资">
67                          </div>
68                      </div>
69                  </div>
70              </div>
```

```
71
72              <!-- 入职时间和密码 -->
73              <div class="row clearfix">
74                  <div class="col-xs-6">
75                      <div class="form-group">
76                          <label class="col-sm-2 control-label">入职时间</label>
77                          <div class="col-sm-10">
78                              <input type="date" class="form-control"
    placeholder="入职时间">
79                          </div>
80                      </div>
81                  </div>
82                  <div class="col-xs-6">
83                      <div class="form-group">
84                          <label for="inputPassword3" class="col-sm-2 control-
    label">密码</label>
85                          <div class="col-sm-10">
86                              <input type="password" class="form-control"
    placeholder="密码">
87                          </div>
88                      </div>
89                  </div>
90              </div>
91
92              <div class="row clearfix">
93                  <div class="col-xs-6">
94                      <div class="form-group">
95                          <div class="col-sm-offset-2 col-sm-10">
96                              <button type="submit" class="btn btn-primary">Sign
    in</button>
97                          </div>
98                      </div>
99                  </div>
100             </div>
101         </form>
102     </div>
103
104 </body>
105 </html>
106
```

Python管理数据库

添加数据

代码实现

> Python 代码实现:
>
> - 添加用户
> - 删除用户
> - 查看用户
> - 更新用户信息

安装pymysql包

```
pip3 install pymysql
1
```

编辑python文件

```
#!/usr/bin/env python3

import pymysql

# 1.连接Mysql
conn = pymysql.connect(host='127.0.0.1', port=3306, user='root', passwd='Syz123!@#', charset='utf8', db='unicom')
cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

# 2.发送指令
cursor.execute("insert into admin(username, password, mobile) values('poker', '123456', '12345678912');")
conn.commit()

# 3.关闭
cursor.close()
conn.close()
```

运行

```
/bin/python3 /root/python/Mysql/createData.py
```

验证

```
mysql> select * from admin;
+----+----------+----------+-------------+
| id | username | password | mobile      |
+----+----------+----------+-------------+
|  3 | poker    | 123456   | 12345678912 |
+----+----------+----------+-------------+
1 row in set (0.00 sec)

```

> 优化

```
#!/usr/bin/env python3

```

```
import pymysql

# 1.连接Mysql
conn = pymysql.connect(host='127.0.0.1', port=3306, user='root',
passwd='Syz123!@#', charset='utf8', db='unicom')
cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

# 2.发送指令
sql = "insert into admin(username, password, mobile) values(%s, %s, %s);"
cursor.execute(sql, ['toker', '123456', '12355674325'])
conn.commit()

# 3.关闭
cursor.close()
conn.close()
```

> 注意: sql语句不要使用字符串格式化,有会SQL注入的风险,需要使用 cursor.execute(sql, [参数1, 参数2, ...])

查询数据

```
#!/usr/bin/env python3

import pymysql

# 1.连接Mysql
conn = pymysql.connect(host='127.0.0.1', port=3306, user='root',
                       passwd='Syz123!@#', charset='utf8', db='unicom')
cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

# 2.发送指令
sql = "select * from admin where id > %s"
cursor.execute(sql, [2, ])
# data_list = cursor.fetchall()     查询一条数据,为字典
data_list = cursor.fetchall()       # 查询所有符合条件的数据,为列表套多个
字典
for row_dict in data_list:
    print(row_dict)

# 3.关闭
cursor.close()
conn.close()
```

输出结果如下

```
[root@hecs-33592 ~]# /bin/python3 /root/python/Mysql/searchData.py
{'id': 3, 'username': 'poker', 'password': '123456', 'mobile': '12345678912'}
{'id': 4, 'username': 'toker', 'password': '123456', 'mobile': '12355674325'}
```

删除数据

> 删除 `id` 大于 3 的行

```
#!/usr/bin/env python3
```

```
2
3   import pymysql
4
5   # 1.连接Mysql
6   conn = pymysql.connect(host='127.0.0.1', port=3306, user='root',
7                          passwd='Syz123!@#', charset='utf8', db='unicom')
8   cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)
9
10  # 2.发送指令
11  sql = "delete from admin where id > %s"
12  cursor.execute(sql, [3, ])
13  conn.commit()
14
15  # 3.关闭
16  cursor.close()
17  conn.close()
```

修改数据

```
1   #!/usr/bin/env python3
2
3   import pymysql
4
5   # 1.连接Mysql
6   conn = pymysql.connect(host='127.0.0.1', port=3306, user='root',
7                          passwd='Syz123!@#', charset='utf8', db='unicom')
8   cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)
9
10  # 2.发送指令
11  sql = "update admin set mobile=%s where id = %s"
12  cursor.execute(sql, ['12332145665', 3])
13  conn.commit()
14
15  # 3.关闭
16  cursor.close()
17  conn.close()
```

强调

- 在进行 新增 删除 修改时，一定要记得 commit 不然数据库没有数据

  cursor.execurte(sql)

  conn.commit()

- 在查询时，不需要 commit，执行 fetchall / fetchone

  ```
  1   cursor.exceccute("...")
  2
  3   # 第一条数据 字典 无数据时是空列表
  4   v1 = cursor,fetchone()
  5   # 所有数据 列表套字典 无数据时是 None
  6   v1 = cursor.fetchall()
  ```

- 对于 sql 语句不要使用 python 的字符串格式化(会被 sql 注入)，一定要使用 execute( )+参数

  ```
  1   cursor.execute("%s....%s",["xx","xxx"])
  ```

案例: Flask + Mysql

main.py

```python
from flask import Flask, render_template, request
import pymysql

app = Flask(__name__)


@app.route("/add/user", methods=['GET', 'POST'])
def addUser():
    if request.method == 'GET':
        return render_template("addUser.html")
    else:
        username = request.form.get('user')
        password = request.form.get('pwd')
        mobile = request.form.get('mobile')

        # 1.连接Mysql
        conn = pymysql.connect(host='127.0.0.1', port=3306, user='root',
                               passwd='Syz123!@#', charset='utf8', db='unicom')
        cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

        # 2.发送指令
        sql = "insert into admin(username, password, mobile) values(%s, %s, %s);"
        cursor.execute(sql, [username, password, mobile])
        conn.commit()

        # 3.关闭
        cursor.close()
        conn.close()

        return "添加成功"


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5200, debug=True)
```

编写一个简单的前端页面添加数据

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <h1>添加用户</h1>
    <form method="post" action="/add/user">
        <input type="text" name="user" placeholder="用户名">
        <input type="text" name="pwd" placeholder="密码">
```

```
        <input type="text" name="mobile" placeholder="手机号">
        <input type="submit" value="提 交">
    </form>
</body>
</html>
```

添加用户

roker | 123456 | 4563112345 | 提交

添加成功


```
mysql> select * from admin;
+----+----------+----------+-------------+
| id | username | password | mobile      |
+----+----------+----------+-------------+
|  3 | poker    | 123456   | 12332145665 |
|  5 | roker    | 123456   | 4563112345  |
+----+----------+----------+-------------+
1234567
```

案例: 查询所有用户

main.py

```
from flask import Flask, render_template, request
import pymysql

app = Flask(__name__)


@app.route("/add/user", methods=['GET', 'POST'])
def addUser():
    if request.method == 'GET':
        return render_template("addUser.html")
    else:
        username = request.form.get('user')
        password = request.form.get('pwd')
        mobile = request.form.get('mobile')

        # 1.连接Mysql
        conn = pymysql.connect(host='127.0.0.1', port=3306, user='root',
                               passwd='Syz123!@#', charset='utf8', db='unicom')
        cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)
```

```
        # 2.发送指令
        sql = "insert into admin(username, password, mobile) values(%s, %s, %s);"
        cursor.execute(sql, [username, password, mobile])
        conn.commit()

        # 3.关闭
        cursor.close()
        conn.close()

        return "添加成功"


@app.route("/show/user", methods=['GET', 'POST'])
def showUser():
    username = request.form.get('user')
    password = request.form.get('pwd')
    mobile = request.form.get('mobile')

    # 1.连接Mysql
    conn = pymysql.connect(host='127.0.0.1', port=3306, user='root',
                           passwd='Syz123!@#', charset='utf8', db='unicom')
    cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

    # 2.发送指令
    sql = "select * from admin"
    cursor.execute(sql)
    data_list = cursor.fetchall()

    # 3.关闭
    cursor.close()
    conn.close()

    return render_template("showUser.html", data_list=data_list)


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5200, debug=True)

```

编写HTML文件

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <h1>用户列表</h1>
    <table border="1">
        <thead>
            <tr>
                <th>ID</th>
                <th>姓名</th>
                <th>密码</th>
```

```
                <th>手机号</th>
            </tr>
        </thead>
        <tbody>
            {% for item in data_list %}
            <tr>
                <td>{{ item.id }}</td>
                <td>{{ item.username }}</td>
                <td>{{ item.password }}</td>
                <td>{{ item.mobile }}</td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
</body>
</html>
```

用户列表

ID	姓名	密码	手机号
3	poker	123456	12332145665
5	roker	123456	4563112345

优化

> 加入 `bootstrap.css`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <link rel="stylesheet" href="../static/plugins/bootstrap-3.4.1/css/bootstrap.css">

</head>
<body>
    <div class="container">
        <h1>用户列表</h1>
        <table class="table table-bordered">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>姓名</th>
                    <th>密码</th>
                    <th>手机号</th>
                </tr>
```

```
            </thead>
            <tbody>
                {% for item in data_list %}
                <tr>
                    <td>{{ item.id }}</td>
                    <td>{{ item.username }}</td>
                    <td>{{ item.password }}</td>
                    <td>{{ item.mobile }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>

</body>
</html>

```

用户列表

ID	姓名	密码	手机号
3	poker	123456	12332145665
5	roker	123456	4563112345

Django

安装Django

```
c:/python39
    - python.exe
    - Scripts
        - pip.exe
        - django-admin.exe   【工具，创建 django 项目】
    - Lib
        - 内置模块
        - site-packages
            - openpyxl
            - python-docx
            - flask
            - django         【框架的源码】
```

安装Python

```
# 安装依赖
yum -y install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel gdbm-devel db4-devel libpcap-devel xz-devel libffi-devel gcc


wget https://www.python.org/ftp/python/3.8.15/Python-3.8.15.tgz
tar xf Python-3.8.15.tar.xz
cd Python-3.8.15
```

```
./configure --prefix=/usr/local/python3
make && make install

# 删除旧软链接
rm -rf /usr/bin/python3
rm -rf /usr/bin/pip3

# 新添加软链接
ln -s /usr/local/python3/bin/python3.8 /usr/bin/python3
ln -s /usr/local/python3/bin/pip3.8 /usr/bin/pip3
```

pip加速

```
cd ~
mkdir .pip
vi .pip/pip.conf
# 增加如下内容
[global]
index-url = https://pypi.tuna.tsinghua.edu.cn/simple
[install]
trusted-host = pypi.tuna.tsinghua.edu.cn
```

安装Django

```
pip3 install django

# 新版跟老版不太一样,需要自己设置软链接
ln -s /usr/local/python3/bin/django-admin /usr/bin/django-admin
```

创建项目

- 打开终端
- 进入某个目录

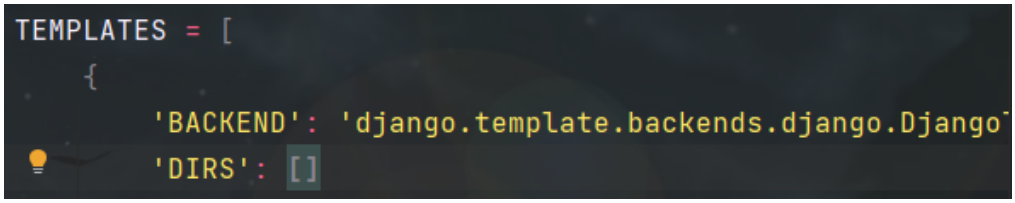
  ```
  /User/Django_learn/Django_
  ```

- 执行命令创建项目

  ```
  "c:\python39\Scripts\django-admin.exe" startproject 项目名称
  ```

  ```
  # 如果 c:\python39\Scripts 已经加入环境系统变量
  django-admin startproject 项目名称
  ```

- 特殊说明
  - 命令行，创建的项目是标准的
  - pycharm 在标准的基础上默认加了点东西
    - 创建了一个 temlpates目录 [删除]
    - settings.py 中 [删除]



Linux

```
django-admin startproject web
```

报错

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
Exception in thread django-main-thread:
Traceback (most recent call last):
  File "/usr/local/python3/lib/python3.8/site-packages/django/db/backends/base/base.py", line 282, in ensure_connection

  ...

  File "/usr/local/python3/lib/python3.8/site-packages/django/db/backends/sqlite3/_functions.py", line 45, in register
    create_deterministic_function("django_date_extract", 2, _sqlite_datetime_extract)
pysqlite3.dbapi2.NotSupportedError: deterministic=True requires SQLite 3.8.3 or higher

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/usr/local/python3/lib/python3.8/threading.py", line 932, in _bootstrap_inner

    ...

  File "/usr/local/python3/lib/python3.8/site-packages/django/db/backends/sqlite3/_functions.py", line 45, in register
    create_deterministic_function("django_date_extract", 2, _sqlite_datetime_extract)
django.db.utils.NotSupportedError: deterministic=True requires SQLite 3.8.3 or higher
```

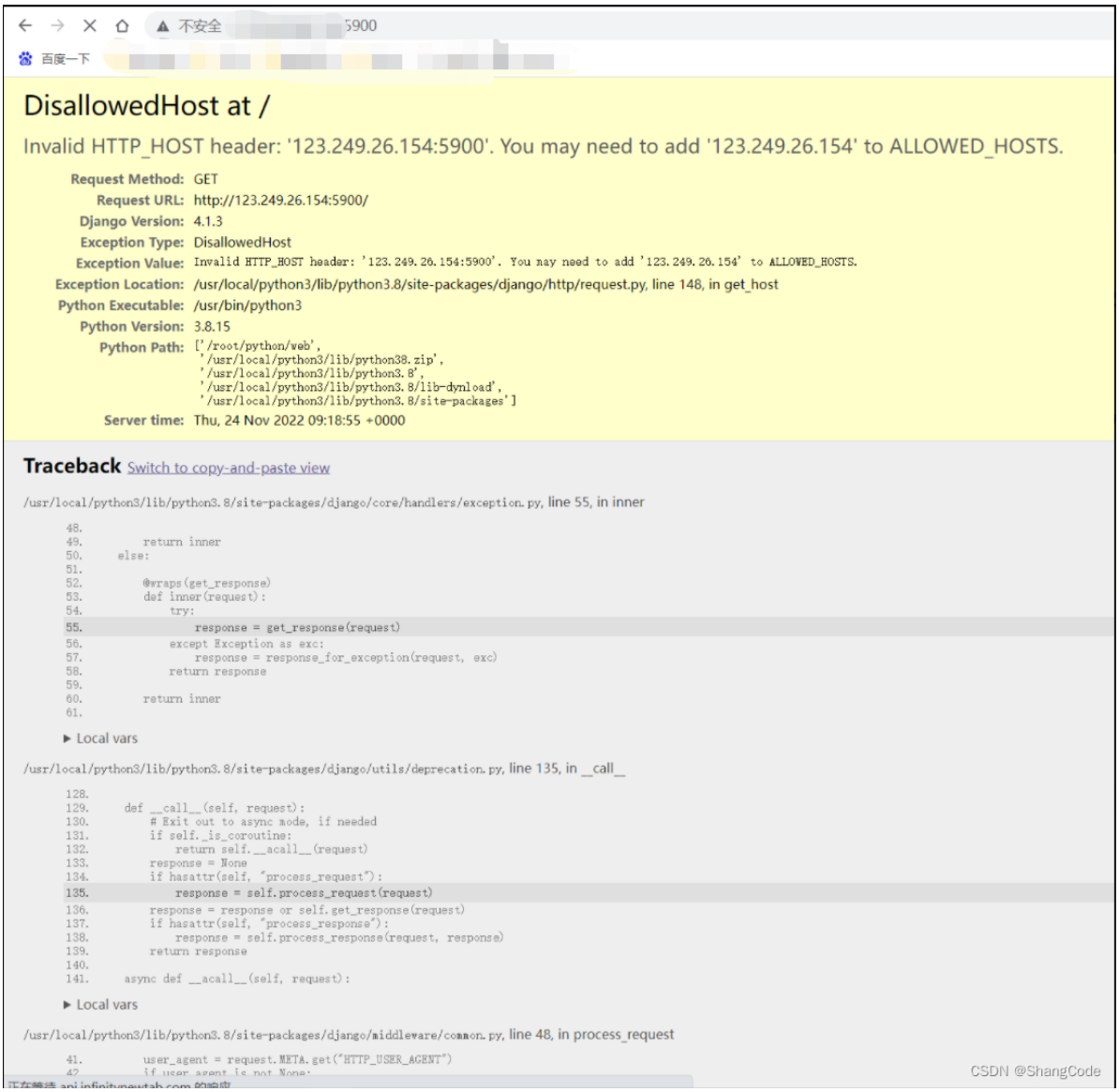
解决办法

```
pip3 install pysqlite3
pip3 install pysqlite3-binary

vim /usr/local/python3/lib/python3.8/site-packages/django/db/backends/sqlite3/base.py
# 修改内容
# from sqlite3 import dbapi2 as Database (注释掉这段)
from pysqlite3 import dbapi2 as Database # 启用pysqlite3
# :wq 保存即可
```

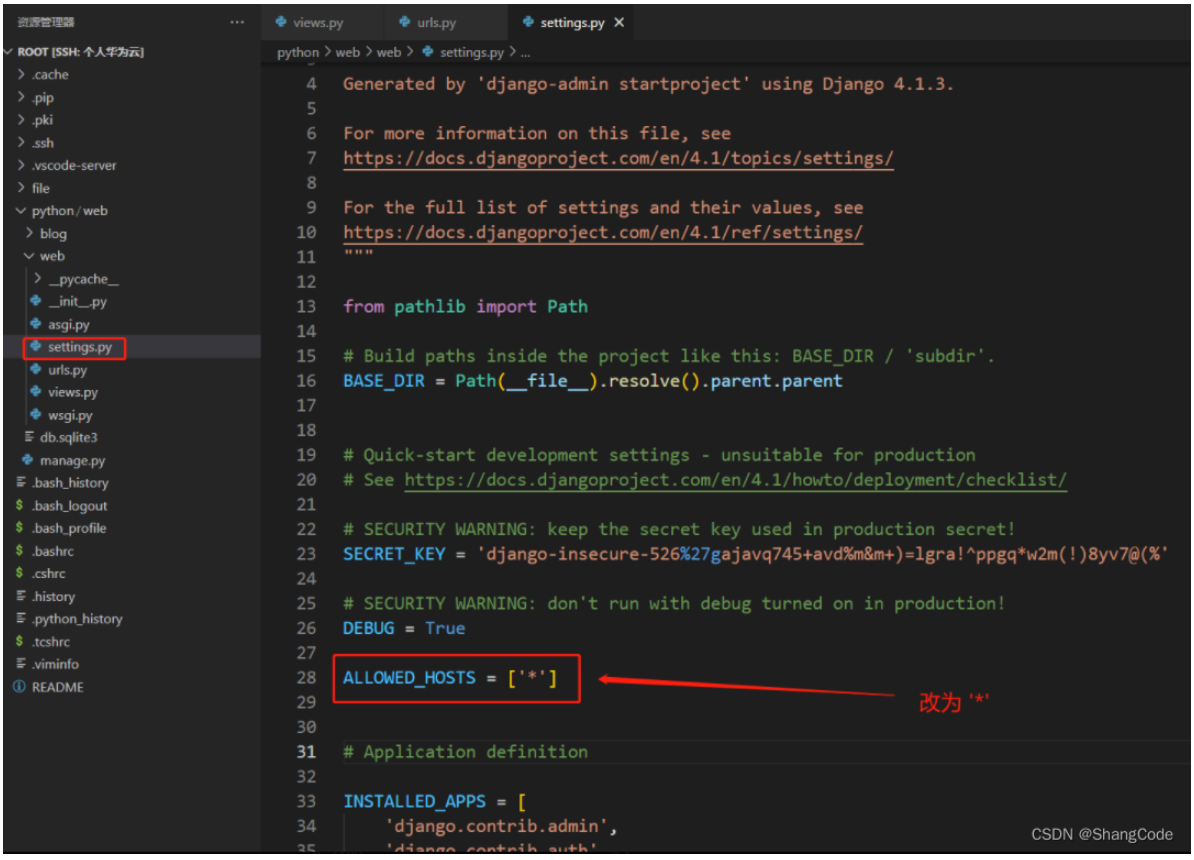
再次运行

```
[root@hecs-33592 web]# python3 manage.py runserver 0.0.0.0:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
November 24, 2022 - 09:16:47
Django version 4.1.3, using settings 'web.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

浏览器访问



报错了,修改ALLOWED_HOSTS



后面我们直接使用`VSCode`进行项目的编辑与运行,有条件的同学可以考虑使用`Pycharm`

文件介绍

```
web
 ├─── db.sqlite3
 ├─── manage.py            # 项目的管理,包括: 启动项目,创建app, 数据管理
 └─── web
     ├─── asgi.py          # 接收网络请求
     ├─── __init__.py
     ├─── settings.py      # 项目配置(模板配置,数据库配置,注册app)
     ├─── urls.py          # url和函数的对应关系
     └─── wsgi.py          # 接收网络请求


Django_
 ├─── manage.py            # 项目的管理,包括: 启动项目,创建app, 数据管理
 └─── Django_
     ├─── __init__.py
     ├─── settings.py      # 项目配置(模板配置,数据库配置,注册app)
     ├─── urls.py          # url和函数的对应关系
     └─── wsgi.py          # 接收网络请求 同步
     ├─── asgi.py          # 接收网络请求 异步

```

简单访问

在`/root/python/web/web`下新增一个`views.py`文件

```
from django.http import HttpResponse
def index(req):
    return HttpResponse('<h1>welcome to Django</h1>')
```

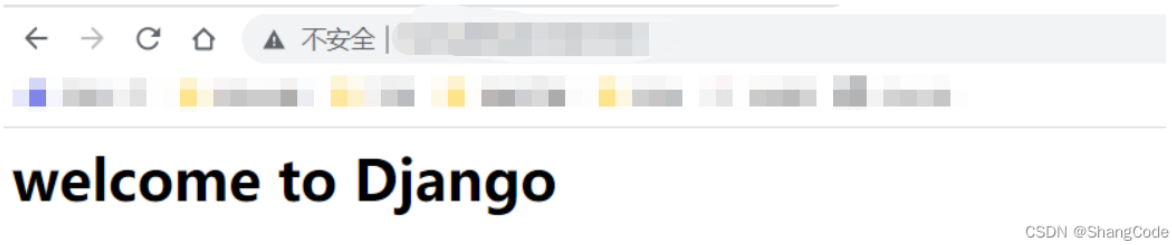
配置`/root/python/web/web`下的`urls.py`文件

```
from django.contrib import admin
from django.urls import path
from . import views    # 导入我们刚刚创建的views.py文件

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.index),  # 新增这一行,''为空表示默认访问 ip:端口
]
```

启动

```
cd /root/python/web/
python3 manage.py runserver 0.0.0.0:5900
```

浏览器访问测试



APP

```
- 项目
    - app，用户管理 【表结构，函数，HTML模板，CSS】
    - app，订单管理 【表结构，函数，HTML模板，CSS】
    - app，后台管理 【表结构，函数，HTML模板，CSS】
    - app，网站     【表结构，函数，HTML模板，CSS】
    - app，API      【表结构，函数，HTML模板，CSS】
    ....
```

添加新的app

进入linux命令行,执行以下命令

```
[root@hecs-33592 web]# pwd
/root/python/web
[root@hecs-33592 web]# django-admin startapp blog
[root@hecs-33592 web]# ls
blog  db.sqlite3  manage.py  web
[root@hecs-33592 web]# tree blog/

Django_learn
├── Django_
    ├── admin.py        # 【固定不用动】 django 默认提供了 admin 后台管理
    ├── apps.py         # 【固定不用动】app 启动类
    ├── __init__.py
    ├── migrations      # 【固定不用动】数据库变更记录
    │   └── __init__.py
    ├── models.py       # 【**重要**】对数据库操作
    ├── tests.py        # 单元测试
    └── views.py        # 【**重要**】函数
├── manage.py           # 项目的管理,包括: 启动项目,创建app, 数据管理
└── Django_learn
    ├── __init__.py
    ├── settings.py     # 项目配置(模板配置,数据库配置,注册app)
    ├── urls.py         # url和函数的对应关系
    └── wsgi.py         # 接收网络请求 同步
    ├── asgi.py         # 接收网络请求 异步
```

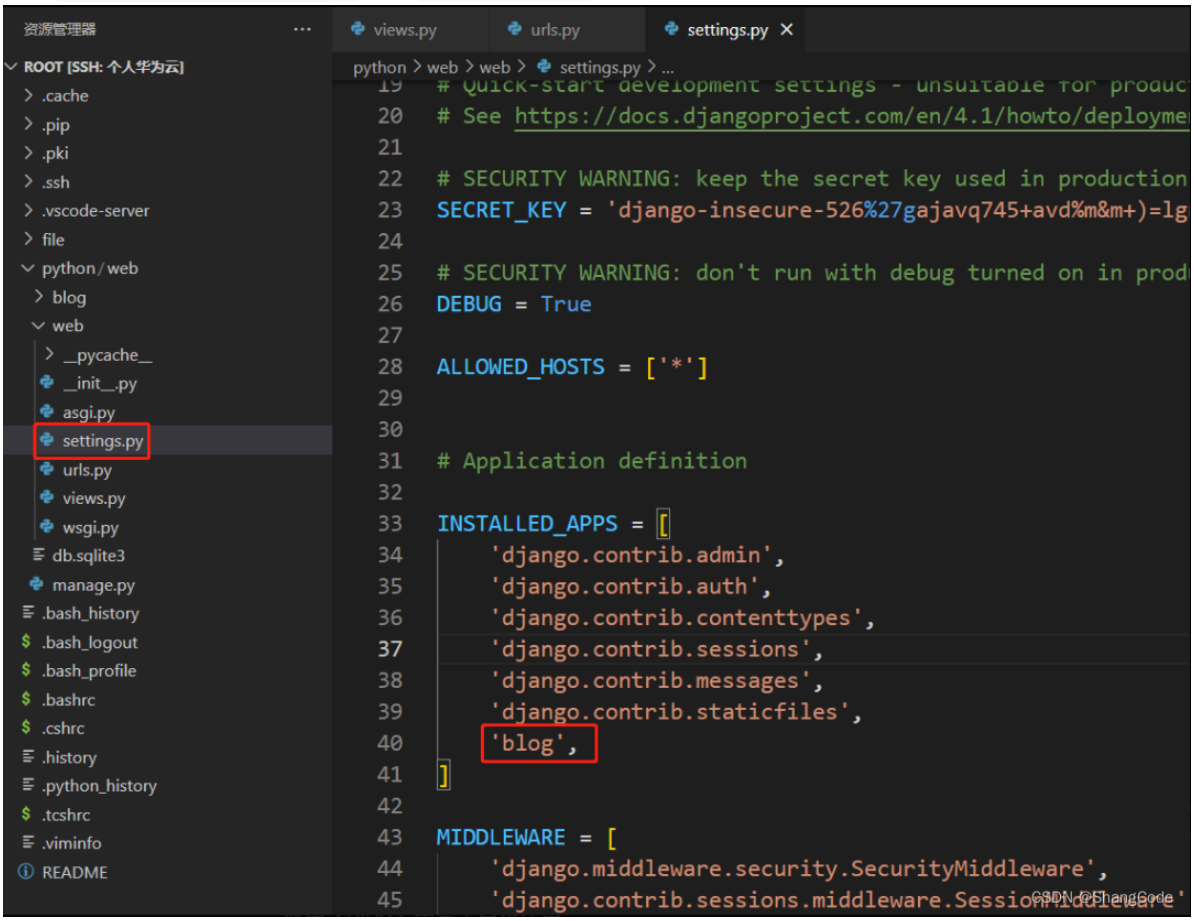
应用【blog】中各个目录作用:

- **migrations:** 用于记录models中数据的变更。
- **admin.py:** 映射models中的数据到Django自带的admin后台。
- **apps.py:** 在新的Django版本中新增，用于应用程序的配置。

- **models.py:** 创建应用程序数据表模型（对应数据库的相关操作）。
- **test.py:** 创建Django测试。
- **views.py:** 控制向前端显示那些数据

注册app

修改 `/root/python/web/web` 下的 `settings.py`



创建blog的页面

编辑 `/root/python/web/blog` 下的 `views.py` 视图函数

```
from django.shortcuts import render
from django.http import HttpResponse
def index_app(req):
    return HttpResponse('welcome to Django blog!')
```

编辑 `/root/python/web/web` 下的 `urls.py` 来指定访问路由

```
from django.contrib import admin
from django.urls import path
from blog.views import index_app

urlpatterns = [
    path('admin/', admin.site.urls),
    path('index_app/', index_app),
]
```

命令行启动Django应用

```
cd /root/python/web/
python3 manage.py runserver 0.0.0.0:5900
```

浏览器访问

> 再来一个

编辑 `/root/python/web/blog` 下的 `views.py` 视图函数

```
from django.shortcuts import render
from django.http import HttpResponse

def user_list(request):
    return HttpResponse("用户列表")

def index_app(req):
    return HttpResponse('<h1>welcome to Django blog!</h1>')
12345678
```

编辑 `/root/python/web/web` 下的 `urls.py` 来指定访问路由

```
 from django.contrib import admin
 from django.urls import path
 from blog.views import index_app
 from blog.views import user_list

 urlpatterns = [
     path('admin/', admin.site.urls),
     path('index_app/', index_app),
     path('user_list/', user_list),
 ]
 12345678910
```

用户列表

templates模板

> 为了使用HTML,这里开始引入templates模板

注意: 可以在app下创建 `templates` 目录,也可以在主目录下创建 `templates` 目录

接下来可以做个测试

编辑 `/root/python/web/blog` 下的 `views.py` 视图函数

```
from django.shortcuts import render
from django.http import HttpResponse

def user_list(request):
    # 1.优先去项目的根目录下寻找
    # 2.根据app的注册顺序去app的目录下templates下寻找
    return render(request, "user_list.html")
1234567
```

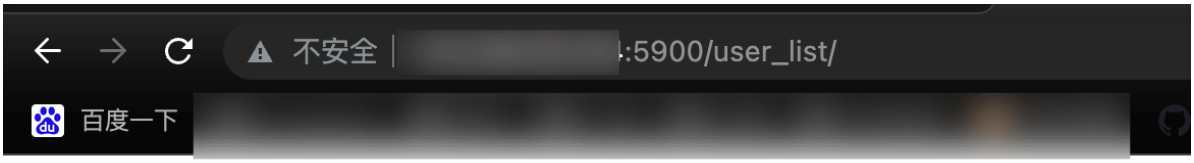
编辑 `/root/python/web/blog/templates` 下的 `user_list.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <h1>用户列表</h1>
</body>
</html>
```

编辑 `/root/python/web/templates` 下的 `user_list.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <h1>根目录,用户列表</h1>
</body>
</html>
```

浏览器访问测试



根目录,用户列表

> 1.优先去项目的根目录 `templates` 下寻找
> 2.根据 `app` 的注册顺序去项目 `app` 的目录下的 `templates` 下寻找

templates模板语法

创建一个新的模板页面

编辑`web`下的`urls.py`

```
from django.contrib import admin
from django.urls import path
from blog import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('index_app/', views.index_app),
    path('user_list/', views.user_list),
    path('tpl/', views.tpl),
]
12345678910
```

编辑APP`blog`下的`views.py`

```
from django.shortcuts import render
from django.http import HttpResponse

def user_list(request):
    # 1.优先去项目的根目录下寻找
    # 2.根据app的注册顺序去app的目录下templates下寻找
    return render(request, "user_list.html")

def index_app(req):
    return HttpResponse('<h1>welcome to Django blog!</h1>')

# 新增下面的内容
def tpl(request):
    return render(request, "tpl.html")
```

在`blog/templates`下新建`tpl.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>templates模板语法</h1>
</body>
</html>
```

templates模板语法

单一变量

> 如果说我们从数据库中取到了数据,如何在`html`页面中进行展示呢,这里需要用到templates的基本语法

修改`blog`下的`views.py`

```
def tpl(request):
    name = "poker"
    return render(request, "tpl.html", {"name":name})
```

修改`blog/templates`下的`tpl.html`

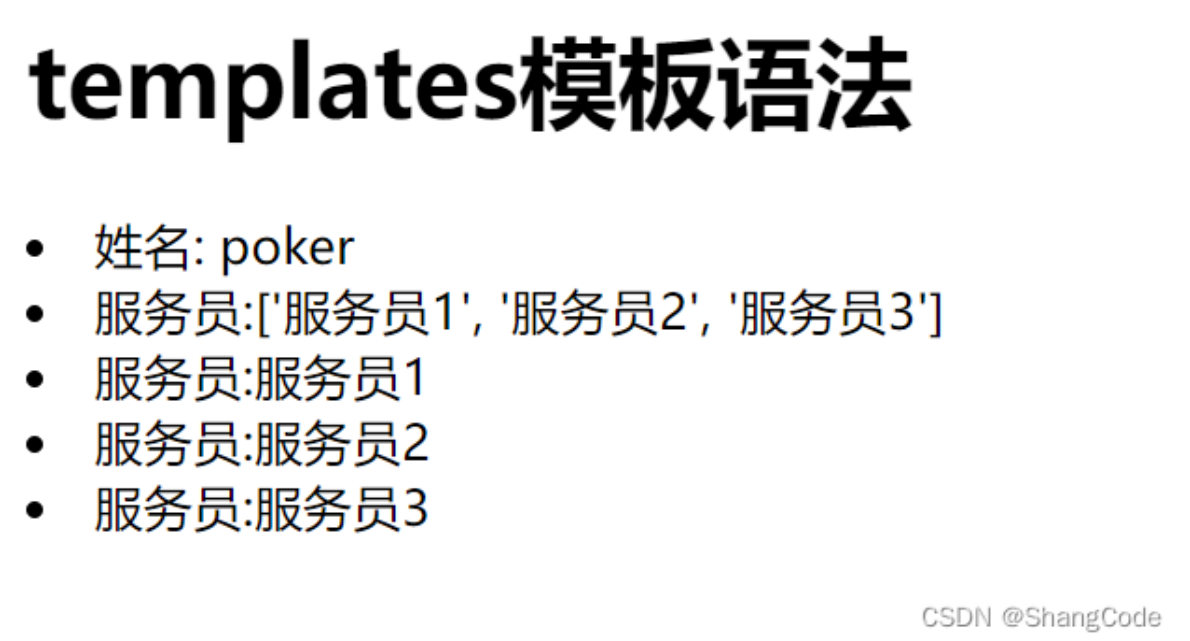
```
<body>
    <h1>templates模板语法</h1>
    <li>姓名: {{ name }}</li>
</body>
```

> 这样,`name`参数就被传到HTML页面中展示了出来

列表

```
def tpl(request):

    name = "poker"
    roles_list = ['服务员1', '服务员2', '服务员3']

    return render(request, "tpl.html", {"name":name, "roles_list":roles_list})
<body>
    <h1>templates模板语法</h1>
    <li>姓名: {{ name }}</li>
    <li>服务员:{{ roles_list }}</li>
    <li>服务员:{{ roles_list.0 }}</li>
```

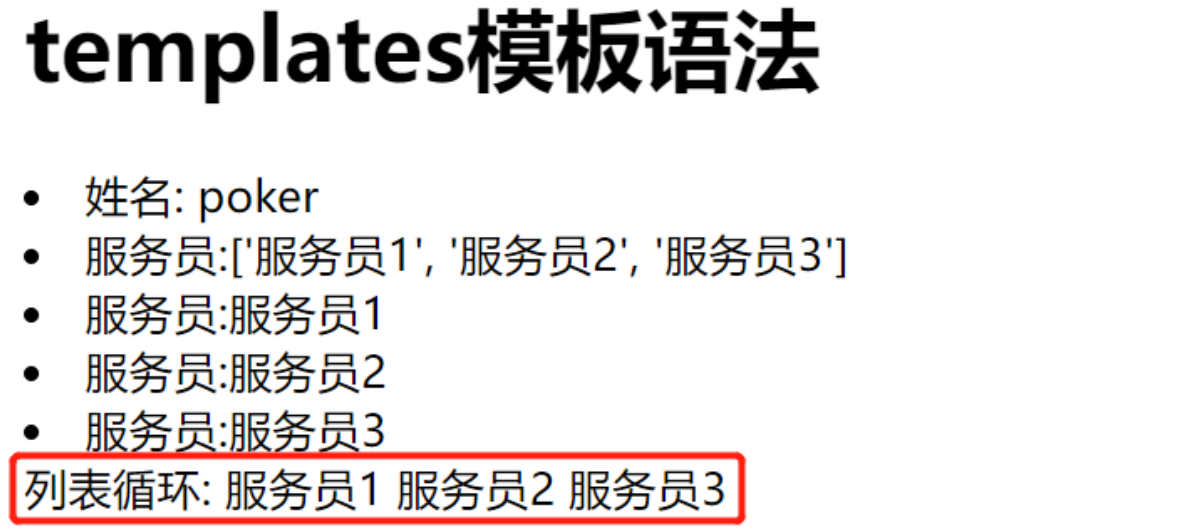
```
        <li>服务员:{{ roles_list.1 }}</li>
        <li>服务员:{{ roles_list.2 }}</li>
    </body>
```



循环(列表)

修改`blog/templates`下的`tpl.html`

```
<div>
    列表循环:
    {% for item in roles_list %}
        <span>{{ item }}</span>
    {% endfor %}
</div>
```



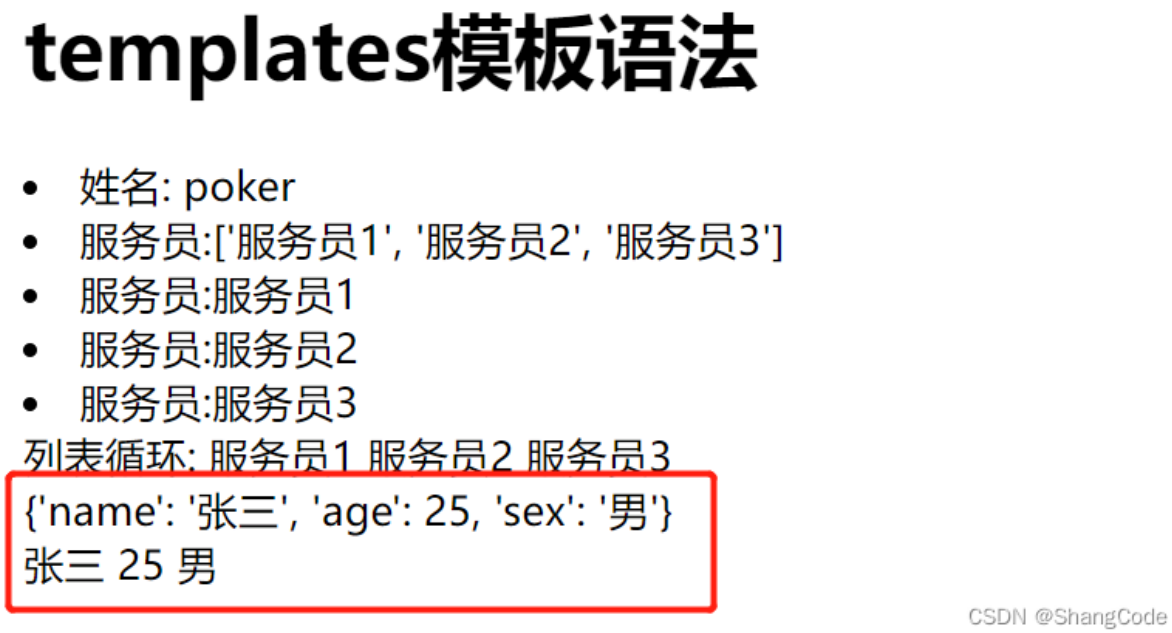
字典

修改`blog`下的`views.py`

```
def tpl(request):

    name = "poker"
    roles_list = ['服务员1', '服务员2', '服务员3']
    user_info = {"name": "张三", "age": 25, "sex": "男"}

    return render(request, "tpl.html", {"name": name, "roles_list": roles_list, "user_info": user_info})
```

修改`blog/templates`下的`tpl.html`

```
<div>
    {{ user_info }}<br>
    {{ user_info.name }}
    {{ user_info.age }}
    {{ user_info.sex }}
</div>
```



循环(字典)

修改`blog/templates`下的`tpl.html`

> 获取值`values`

```
<div>
    {% for v in user_info.values %}
        <li>{{ v }}</li>
    {% endfor %}
</div>
```

> 获取键`keys`

```
    <div>
        {% for k in user_info.keys %}
            <li>{{ k }}</li>
        {% endfor %}
    </div>
```

获取键和值 `items`

```
<div>
    {% for k,v in user_info.items %}
        <li>{{ k }} = {{ v }}</li>
    {% endfor %}
</div>
```

列表套字典

修改 `blog` 下的 `views.py`

```
def tpl(request):

    name = "poker"
    roles_list = ['服务员1', '服务员2', '服务员3']
    user_info = {"name": "张三", "age": 25, "sex": "男"}

    date_list = [
        {"name": "张三", "age": 25, "sex": "男"},
        {"name": "李四", "age": 18, "sex": "男"},
        {"name": "王五", "age": 22, "sex": "女"},
    ]

    return render(request, "tpl.html", {"name": name, "roles_list": roles_list, "user_info": user_info, "data_list":date_list})
```

修改`blog/templates`下的`tpl.html`

```
<div>
    {% for item in data_list %}
        <div>{{ item.name }} {{ item.age }} {{ item.sex }}</div>
    {% endfor %}
</div>
```

条件判断

修改`blog/templates`下的`tpl.html`

```
{%  if name == "poker" %}
    <h3>嘿嘿嘿</h3>
{% elif name == "toker" %}
    <h3>哈哈哈</h3>
{% else %}
    <h3>呵呵呵</h3>
{% endif %}
```

> 先介绍这些常用的语法,其实还有很多的语法,感兴趣的可自行百度.

请求和响应

重定向: 浏览器向某个网站发送请求,网站返回给浏览器一个新的URL,浏览器去访问这个新的URL地址
修改`blog`下的`views.py`,根据情况去掉下面代码的注释进行测试

```
# 导入 redirect 包
```

```python
from django.shortcuts import render, redirect
...

# 增加新函数
def something(request):
    # request 是一个对象,封装了用户发送过来的所有请求相关数据

    # 1.[请求]获取请求的方式
    print("用户请求的方式: " + request.method)

    # 2.[请求]在URL上传递值, 例如: http://123.249.26.154:5900/something/?n1=1&n2=2
    print(request.GET)

    # 3.[请求]在请求体中提交数据,目前是空值
    print(request.POST)

    # 4.[响应]HttpResponse("返回内容"), 内容字符串内容返回给请求者
    # return HttpResponse("返回内容")

    # 5.[响应]读取HTML的内容 + 渲染(替换) => 字符串,返回给用户浏览器
    # 需要在 blog/templates 下新建`something.html`
    #return render(request, 'something.html', {"title": "hello"})

    # 6.[响应] 让浏览器重定向到其他的页面
    return redirect("http://www.baidu.com")
```

修改`web/web/urls.py`,增加`something`

```python
from django.contrib import admin
from django.urls import path
from blog import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('index_app/', views.index_app),
    path('user_list/', views.user_list),
    path('tpl/', views.tpl),
    path('something/', views.something),
]
```

案例: 用户登录

修改`blog`下的`views.py`,新增`login`函数

```python
def login(request):
    if request.method == "GET":
        return render(request, "login.html")

    # 如果是 POST 请求,获取用户提交的数据
    print(request.POST)
    username = request.POST.get("user")
    password = request.POST.get("password")
    if username == "poker" and password == "123":
        return HttpResponse("登录成功")

    #return HttpResponse("登录失败")
```

```
        return render(request, "login.html", {"error_msg": "用户名或密码错误"})
12345678910111213
```

修改 `web/web/urls.py` ,增加 `login`

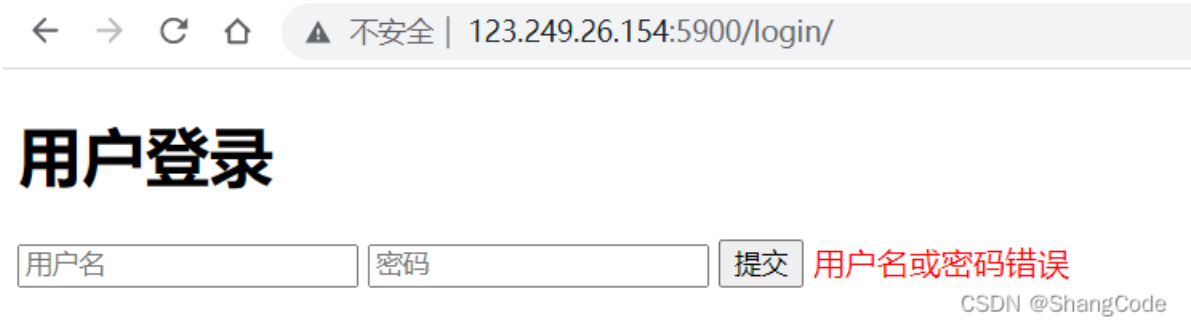
```
from django.contrib import admin
from django.urls import path
from blog import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('index_app/', views.index_app),
    path('user_list/', views.user_list),
    path('tpl/', views.tpl),
    path('something/', views.something),
    path('login/', views.login),
]
```

在 `blog/templates` 下新建 `login.html`

> {% csrf_token %} 必须加上

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <h1>用户登录</h1>
    <form method="post" action="/login/">

        {% csrf_token %}

        <input type="text" name="user", placeholder="用户名">
        <input type="password" name="password", placeholder="密码">
        <input type="submit" value="提交">
        <span style="color: red;">{{ error_msg }}</span>
    </form>
</body>
</html>
```

浏览器访问,输入错误的用户名和密码



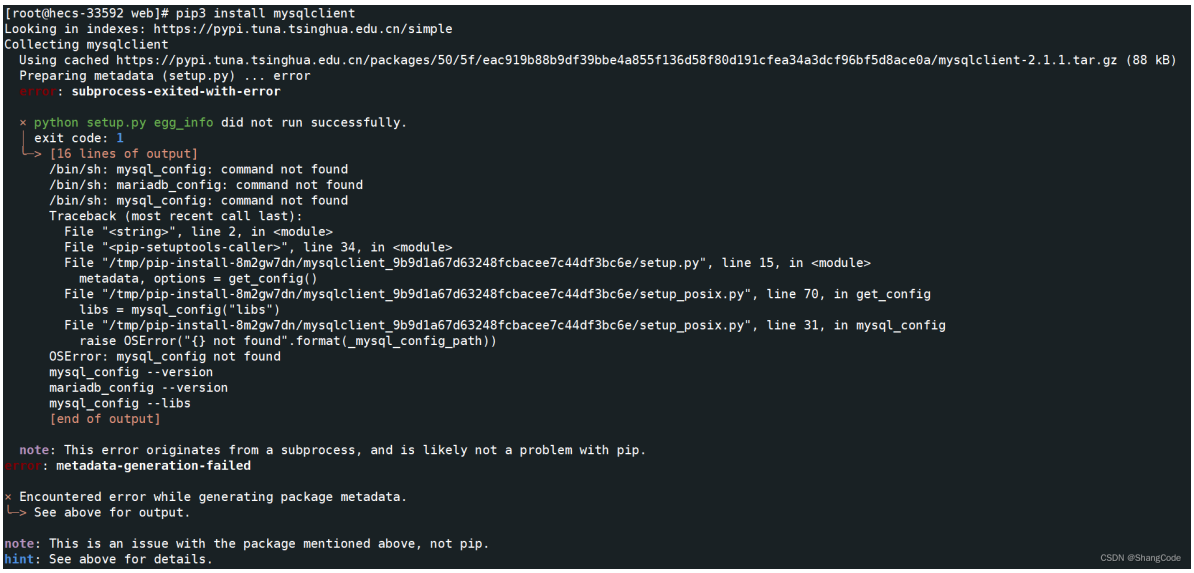
数据库操作

Django开发操作数据库更简单,内部提供了ORM框架

安装第三方模块

> mysqlclient

直接安装可能会报错



解决办法

参考链接: https://blog.csdn.net/m0_67155975/article/details/123138225

```
yum -y install mysql-devel
yum -y install python-devel
pip3 install mysqlclient
```

ORM

ORM可以帮助我们做两件事;

- 创建/修改/删除数据库中的表(无法创建数据库)
- 操作表中的数据

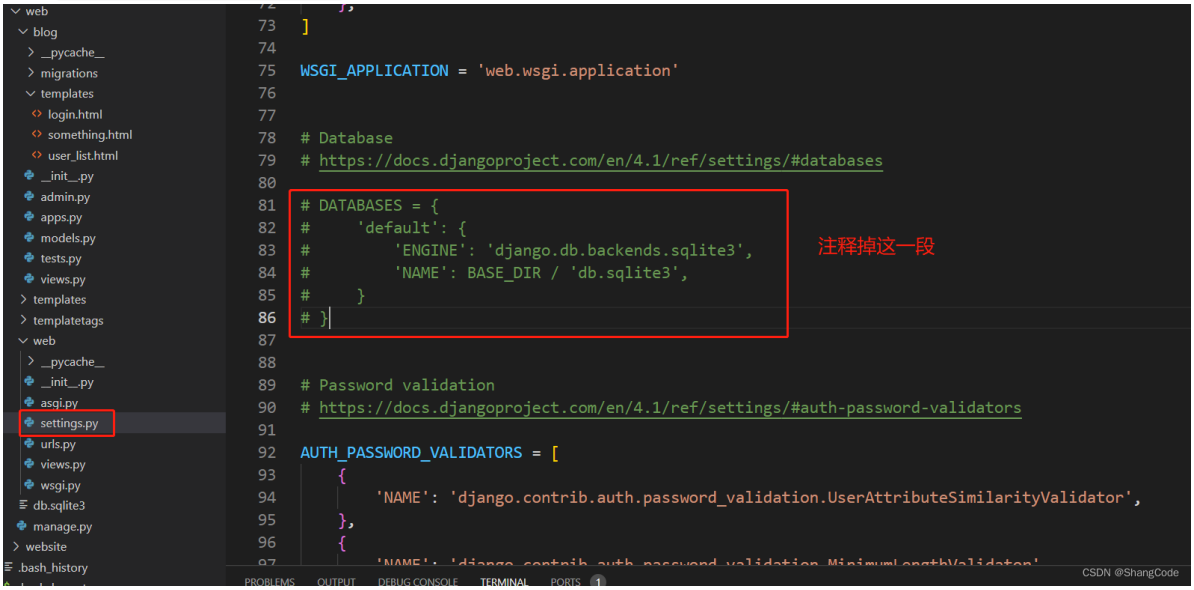
创建数据库

```
create database mydb DEFAULT CHARSET utf8 COLLATE utf8_general_ci;
```

```
mysql> create database mydb DEFAULT CHARSET utf8 COLLATE utf8_general_ci;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| d1                 |
| mydb               |
| mysql              |
| performance_schema |
| sys                |
| unicom             |
+--------------------+
7 rows in set (0.00 sec)
```


Django连接数据库

修改`web/web/settings.py`



增加如下内容

```python
DATABASES = {
    'default':{
        'ENGINE':'django.db.backends.mysql',
        'NAME':'mydb',
        'USER':'root',
        'PASSWORD':'Syz123!@#',
        'HOST':'127.0.0.1',
        'PORT':'3306',
    }
}
```

Django操作表

- 创建表
- 删除表
- 修改表

配置`blog`下的`models.py`

> 会根据自定义的类创建跟app同名的表

```python
from django.db import models

# Create your models here.
class UserInfo(models.Model):
    name = models.CharField(max_length=20)
    password = models.CharField(max_length=20)
    age = models.IntegerField()

"""
create table UserInfo(
    id bigint auto_increment primary key,
    name varchar(20),
    password varchar(20),
    age int
)
"""
```

在服务器中项目根目录下执行命令

> 如果不想要某个表了,将**class类**注释后,重新执行下面的命令即可

```
python3 manage.py makemigrations
python3 manage.py migrate
```

注意: app需要提前注册

```
[root@hecs-33592 web]# python3 manage.py makemigrations
Migrations for 'blog':
  blog/migrations/0001_initial.py
    - Create model UserInfo
[root@hecs-33592 web]# python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, blog, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying blog.0001_initial... OK
  Applying sessions.0001_initial... OK
```


查看Mysql数据库

```
mysql> use mydb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------------------+
| Tables_in_mydb             |
+----------------------------+
| auth_group                 |
| auth_group_permissions     |
| auth_permission            |
| auth_user                  |
| auth_user_groups           |
| auth_user_user_permissions |
| blog_userinfo              |
| django_admin_log           |
| django_content_type        |
| django_migrations          |
| django_session             |
+----------------------------+
11 rows in set (0.00 sec)
```

这个是app对应的表

修改表的话,如果原表中存有数据,此时如果增加一个新的列,需要设定一个默认值

- 手动设定

```
1    age = models.IntegerField(default=2)
```

- 允许为空

```
1    data = models.IntegerField(null=True, blank=True)
```

Django操作表数据

- 添加数据

修改 `blog` 下的 `views.py`

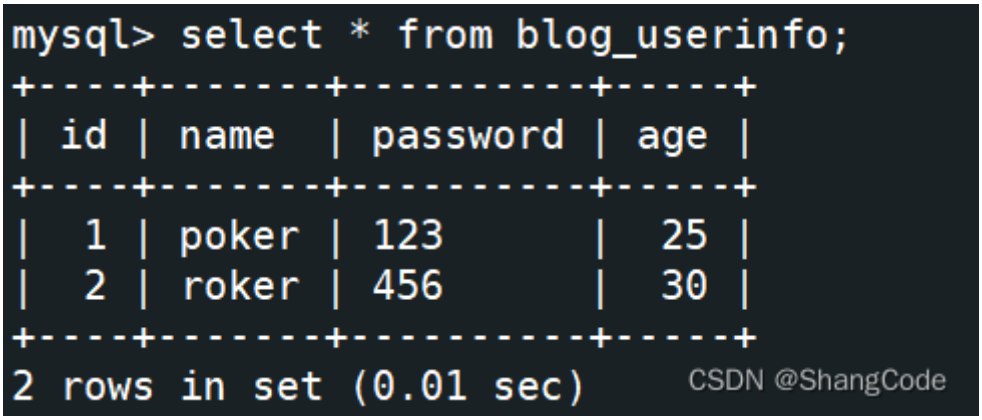
```
from blog.models import UserInfo
...

def orm(request):
    # 新建数据
    UserInfo.objects.create(name="poker", password="123", age=25)
    UserInfo.objects.create(name="roker", password="456", age=30)

    return HttpResponse("成功")

```

修改 `web/web/urls.py` ,增加 `orm`

```
from django.contrib import admin
from django.urls import path
from blog import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('index_app/', views.index_app),
    path('user_list/', views.user_list),
    path('tpl/', views.tpl),
    path('something/', views.something),
    path('login/', views.login),
    path('orm/', views.orm),
]
```

浏览器访问页面

查看Mysql数据库



- 删除数据

```
    # 删除数据
    UserInfo.objects.filter(id=2).delete()
    # 删除表中所有数据
    UserInfo.objects.all().delete()
```

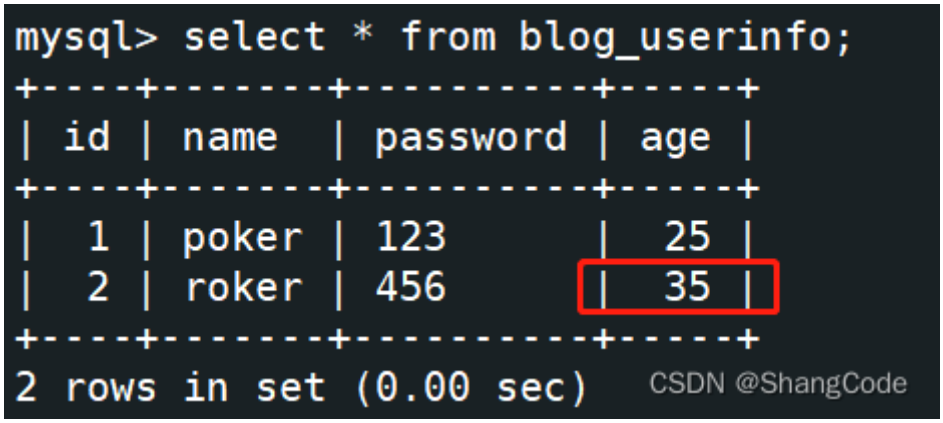
- 获取数据

```
# data_list = [行,行,行] QuerySet 类型 每一行都是对象
data_list = UserInfo.objects.all()
print(data_list)
for obj in data_list:
    print(obj.id, obj.name, obj.password, obj.age)
# 获取第一行的数据
row_obj = UserInfo.objects.filters(id=1).first()
```

浏览器刷新访问,观察工作台输出

- 更新数据

```
UserInfo.objects.filter(name="roker").update(age=35)
```



案例:用户管理

展示用户列表

修改`web/web/urls.py`,增加`info/list`

```
from django.contrib import admin
from django.urls import path
from blog import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('index_app/', views.index_app),
    path('user_list/', views.user_list),
    path('tpl/', views.tpl),
    path('something/', views.something),
    path('login/', views.login),
```

```
    path('orm/', views.orm),
    path('info/list/', views.info_list)
]
```

修改 `blog` 下的 `views.py`

```
from blog.models import UserInfo
...

def info_list(request):
    data_list = UserInfo.objects.all()

    return render(request, "info_list.html", {"data_list": data_list})
```

在 `blog/templates` 下新增 `info_list.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <table border="1">
        <thead>
            <tr>
                <th>ID</th>
                <th>姓名</th>
                <th>密码</th>
                <th>年龄</th>
            </tr>
        </thead>
        <tbody>
            {% for obj in data_list %}
                <tr>
                    <td>{{ obj.id }}</td>
                    <td>{{ obj.name }}</td>
                    <td>{{ obj.password }}</td>
                    <td>{{ obj.age }}</td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
</body>
</html>
```

浏览器访问测试

ID	姓名	密码	年龄
1	poker	123	25
2	roker	456	35

添加用户

修改`web/web/urls.py`,增加`info/list`

```python
from django.contrib import admin
from django.urls import path
from blog import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('index_app/', views.index_app),
    path('user_list/', views.user_list),
    path('tpl/', views.tpl),
    path('something/', views.something),
    path('login/', views.login),
    path('orm/', views.orm),
    path('info/list/', views.info_list),
    path('info/add/', views.info_add),
]
```

修改`blog`下的`views.py`

```python
def info_add(request):
    if request.method == "GET":
        return render(request, 'info_add.html')

    # 获取用户提交的数据
    name = request.POST.get("name")
    password = request.POST.get("password")
    age = request.POST.get("age")

    # 添加到数据库
    UserInfo.objects.create(name=name, password=password, age=age)

    # 自动跳转
    return redirect("/info/list/")
```

在`blog/templates`下新增`info_add.html`

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <form action="/info/add/" method="post">

        {% csrf_token %}

        <input type="text" name="name" placeholder="用户名">
        <input type="text" name="password" placeholder="密码">
        <input type="text" name="age" placeholder="年龄">
        <input type="submit" value="提交">
    </form>
</body>
```

```
18  </html>
```

浏览器访问

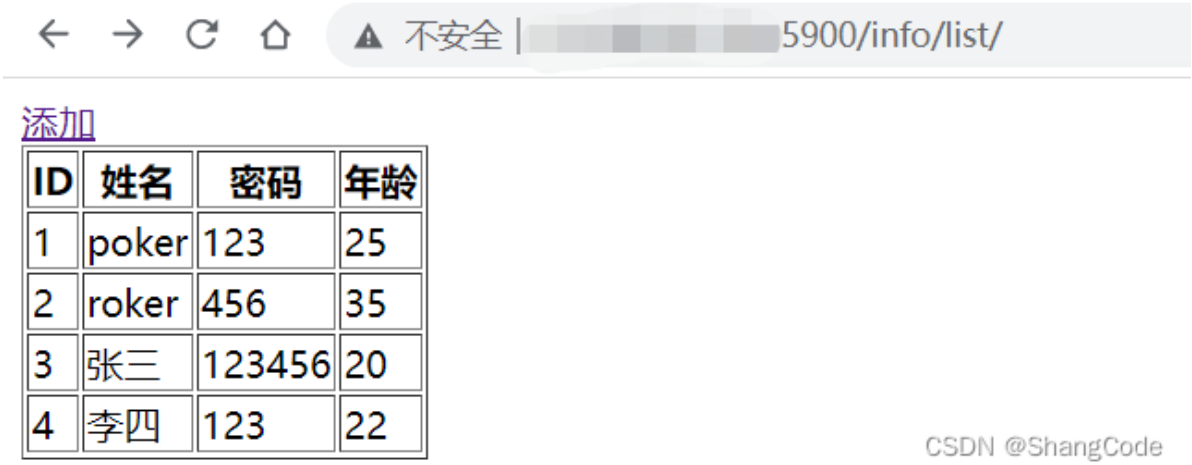
点击"提交"

ID	姓名	密码	年龄
1	poker	123	25
2	roker	456	35
3	张三	123456	20

在 `info/list` 页面增加"添加"按钮

修改 `blog/templates` 下 `info_list.html`

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>Document</title>
6   </head>
7   <body>
8       <a href="/info/add">添加</a>
9       <table border="1">
10          <thead>
11              <tr>
12                  <th>ID</th>
13                  <th>姓名</th>
14                  <th>密码</th>
15                  <th>年龄</th>
16              </tr>
17          </thead>
18          <tbody>
19              {% for obj in data_list %}
20                  <tr>
21                      <td>{{ obj.id }}</td>
22                      <td>{{ obj.name }}</td>
23                      <td>{{ obj.password }}</td>
24                      <td>{{ obj.age }}</td>
25                  </tr>
26              {% endfor %}
27          </tbody>
28      </table>
29  </body>
30  </html>
```



点击"添加"后,即可跳转回添加页面

删除用户

修改`web/web/urls.py`,增加`info/list`

```
from django.contrib import admin
from django.urls import path
from blog import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('index_app/', views.index_app),
    path('user_list/', views.user_list),
    path('tpl/', views.tpl),
    path('something/', views.something),
    path('login/', views.login),
    path('orm/', views.orm),
    path('info/list/', views.info_list),
    path('info/add/', views.info_add),
    path('info/delete/', views.info_delete)
]
```

修改`blog`下的`views.py`

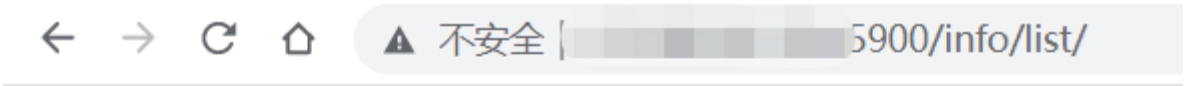
```
def info_delete(request):
    nid = request.GET.get("nid")
    UserInfo.objects.filter(id=nid).delete()
    return redirect("/info/list/")
1234
```

修改`blog/templates`下的`info_list.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <a href="/info/add">添加</a>
    <table border="1">
        <thead>
            <tr>
```

```
                <th>ID</th>
                <th>姓名</th>
                <th>密码</th>
                <th>年龄</th>
                <th>操作</th>
            </tr>
        </thead>
        <tbody>
            {% for obj in data_list %}
                <tr>
                    <td>{{ obj.id }}</td>
                    <td>{{ obj.name }}</td>
                    <td>{{ obj.password }}</td>
                    <td>{{ obj.age }}</td>
                    <td>
                        <a href="/info/delete?nid={{ obj.id }}">删除</a>
                    </td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
</body>
</html>
```

浏览器访问,点击"删除"即可将对应的行删除



添加

ID	姓名	密码	年龄	操作
1	poker	123	25	删除
2	roker	456	35	删除
5	张三	123456	20	删除

部门管理

部门列表

修改`myproject/myproject/urls.py`

```
from django.contrib import admin
from django.urls import path
from employee_management import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('depart/list/', views.depart_list),
]
```

设计数据库表结构

```python
from django.db import models
class Department(models.Model):
    """部门表"""
    title = models.CharField(verbose_name="标题", max_length=32)


class UserInfo(models.Model):
    """员工表"""
    name = models.CharField(verbose_name="姓名",max_length=16)
    password = models.CharField(verbose_name="密码",max_length=64)
    age = models.IntegerField(verbose_name="年龄")
    account = models.DecimalField(verbose_name="账户余额",max_digits=10,decimal_places=2,default=2)
    create_time = models.DateTimeField(verbose_name="入职时间")

    # 有约束
    # 外键约束
    # Django 自动
    #   - 写的 depart
    #   - 生成数据列 depart_id on_delete = models.CASCADE 删除级联
    depart = models.ForeignKey(to="Department",to_field="id",on_delete=models.CASCADE)
    # 删除置空
    # depart = models.ForeignKey(to="Department",to_field="id",on_delete=models.SET_NULL,null=True,blank=True)
    # 在 Django 中做的约束
    gender_choices = (
        (1:'男'),
        (2:'女'),
    )

    gender = models.SmallIntegerField(verbose_name="性别", choices=gender_choices)


```

修改 `myproject/employee_management/views.py`

```python
from turtle import title
from django.shortcuts import render, redirect
from employee_management.models import Department,UserInfo

# Create your views here.
def depart_list(request):
    """部门列表"""

    depart_list = Department.objects.all()

    return render(request, "depart_list.html", {"depart_list": depart_list})
```

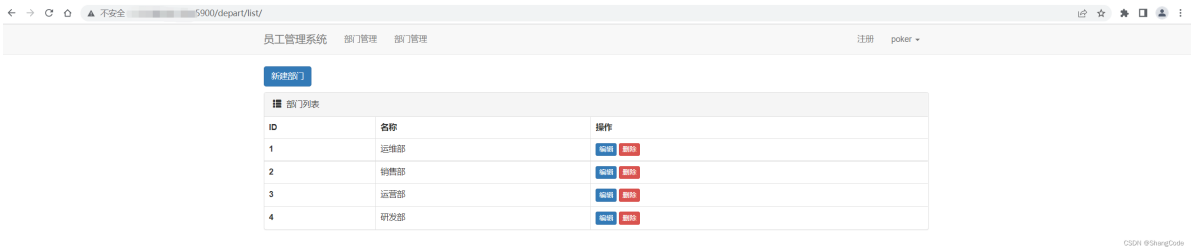
在 `myproject/employee_management/templates` 下新建 `depart_list.html`

```
<!DOCTYPE html>
```

```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <link rel="stylesheet" href="/static/plugins/bootstrap-3.4.1/css/bootstrap.css">
    <link rel="stylesheet" href="/static/plugins/font-awesome-4.7.0/css/font-awesome.css">

    <style>
        .navbar {
            border-radius: 0;
        }
    </style>
</head>

<body>

    <!-- 导航条, https://v3.bootcss.com/components/#navbar -->
    <nav class="navbar navbar-default">
        <div class="container">
            <!-- Brand and toggle get grouped for better mobile display -->
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
                    data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="#">员工管理系统</a>
            </div>

            <!-- Collect the nav links, forms, and other content for toggling -->
            <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                <ul class="nav navbar-nav">
                    <li><a href="#">部门管理</a></li>
                    <li><a href="#">部门管理</a></li>

                </ul>
                <ul class="nav navbar-nav navbar-right">
                    <li><a href="#">注册</a></li>
                    <li class="dropdown">
                        <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true"
                            aria-expanded="false">poker <span class="caret"></span></a>
                        <ul class="dropdown-menu">
                            <li><a href="#">Action</a></li>
                            <li><a href="#">Another action</a></li>
```

```
                            <li><a href="#">Something else here</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">Separated link</a></li>
                        </ul>
                    </li>
                </ul>
            </div><!-- /.navbar-collapse -->
        </div><!-- /.container-fluid -->
    </nav>
    <!-- 新建区域 -->
    <div>
        <div class="container">
            <div style="margin-bottom: 10px">
                <a class="btn btn-primary" href="/depart/add/" target="_blank">新建部门</a>
            </div>
            <div>
                <div class="panel panel-default">
                    <!-- Default panel contents -->
                    <div class="panel-heading">
                        <span class="glyphicon glyphicon-th-list" aria-hidden="true" style="margin-right: 5px;"></span>
                        <span>部门列表</span>
                    </div>

                    <!-- Table -->
                    <table class="table table-bordered">
                        <thead>
                            <tr>
                                <th>ID</th>
                                <th>名称</th>
                                <th>操作</th>
                            </tr>
                        </thead>
                        <tbody>
                            {% for obj in depart_list %}
                            <tr>
                                <th>{{ obj.id }}</th>
                                <td>{{ obj.title }}</td>
                                <td>
                                    <button type="button" class="btn btn-primary btn-xs">编辑</button>
                                    <button type="button" class="btn btn-danger btn-xs">删除</button>
                                </td>
                            </tr>
                            {% endfor %}
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>

    <script src="/static/js/jquery-3.6.1.min.js"></script>
    <script src="/static/plugins/bootstrap-3.4.1/js/bootstrap.min.js"></script>
```

```
102     </body>
103     </html>
```



部门添加

修改 `myproject/myproject/urls.py`

```
1   from django.contrib import admin
2   from django.urls import path
3   from employee_management import views
4
5   urlpatterns = [
6       path('admin/', admin.site.urls),
7       path('depart/list/', views.depart_list),
8       path('depart/add/', views.depart_add),
9   ]
```

修改 `myproject/employee_management/views.py`

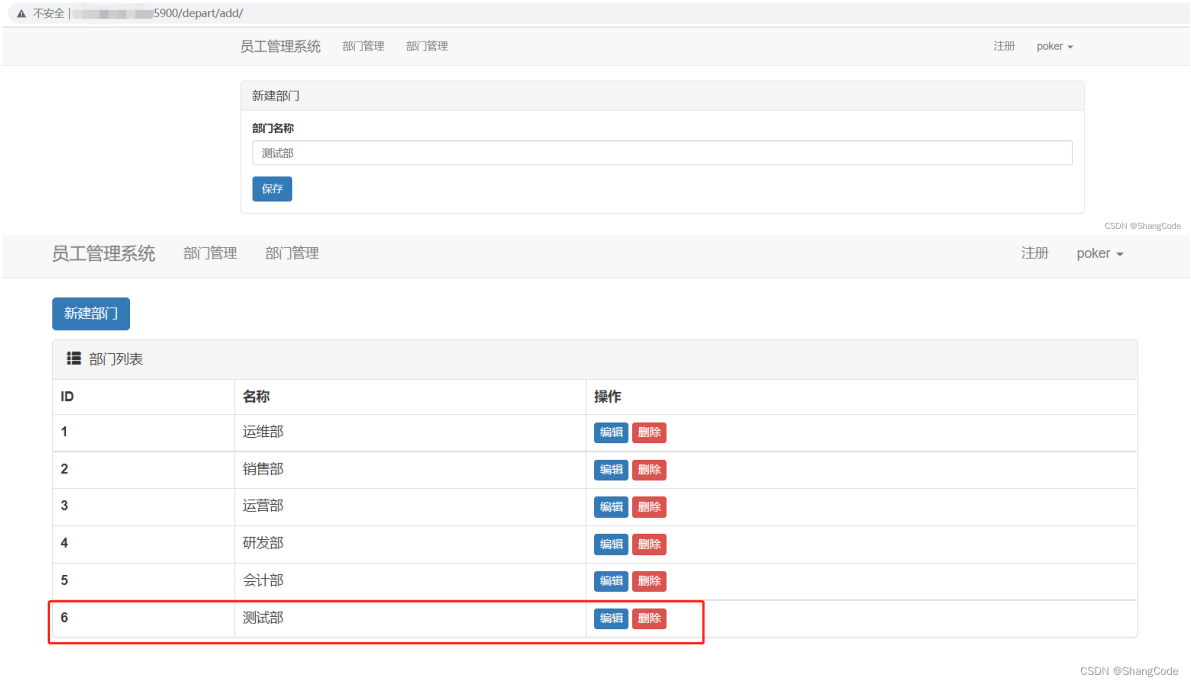
```
1   def depart_add(request):
2       """部门添加"""
3       if request.method == "GET":
4           return render(request, "depart_add.html")
5
6       # 获取用户提交的部门数据
7       depart_title = request.POST.get("depart_title")
8
9       # 保存到数据库
10      Department.objects.create(title=depart_title)
11
12      return redirect("/depart/list/")
```

在 `myproject/employee_management/templates` 下新建 `depart_list.html`

```
1   <!DOCTYPE html>
2   <html lang="en">
3
4   <head>
5       <meta charset="UTF-8">
6       <title>Document</title>
7
8       <link rel="stylesheet" href="/static/plugins/bootstrap-
    3.4.1/css/bootstrap.css">
9       <link rel="stylesheet" href="/static/plugins/font-awesome-4.7.0/css/font-
    awesome.css">
10
11      <style>
```

```
        .navbar {
            border-radius: 0;
        }
    </style>

</head>

<body>
    <div>
        <!-- 导航条, https://v3.bootcss.com/components/#navbar -->
        <nav class="navbar navbar-default">
            <div class="container">
                <!-- Brand and toggle get grouped for better mobile display -->
                <div class="navbar-header">
                    <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
                        data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
                        <span class="sr-only">Toggle navigation</span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                    </button>
                    <a class="navbar-brand" href="#">员工管理系统</a>
                </div>

                <!-- Collect the nav links, forms, and other content for toggling -->
                <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                    <ul class="nav navbar-nav">
                        <li><a href="#">部门管理</a></li>
                        <li><a href="#">部门管理</a></li>

                    </ul>
                    <ul class="nav navbar-nav navbar-right">
                        <li><a href="#">注册</a></li>
                        <li class="dropdown">
                            <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button"
                                aria-haspopup="true" aria-expanded="false">poker <span class="caret"></span></a>
                            <ul class="dropdown-menu">
                                <li><a href="#">Action</a></li>
                                <li><a href="#">Another action</a></li>
                                <li><a href="#">Something else here</a></li>
                                <li role="separator" class="divider"></li>
                                <li><a href="#">Separated link</a></li>
                            </ul>
                        </li>
                    </ul>
                </div><!-- /.navbar-collapse -->
            </div><!-- /.container-fluid -->
        </nav>
    </div>
    <div class="container">
```

```
62          <div class="panel panel-default">
63              <div class="panel-heading">
64                  <h3 class="panel-title">新建部门</h3>
65              </div>
66              <div class="panel-body">
67                  <form action="/depart/add/" method="post">
68                      {% csrf_token %}
69                      <div class="form-group">
70                          <label>部门名称</label>
71                          <input type="text" class="form-control" placeholder="部门名称" name="depart_title">
72                      </div>
73
74                      <button type="submit" class="btn btn-primary">保存</button>
75
76                  </form>
77              </div>
78          </div>
79      </div>
80  </body>
81  </html>
```



部门删除

修改`myproject/myproject/urls.py`

```python
from django.contrib import admin
from django.urls import path
from employee_management import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('depart/list/', views.depart_list),
    path('depart/add/', views.depart_add),
    path('depart/delete/', views.depart_delete),
]
```

修改 `myproject/employee_management/views.py`

```python
def depart_delete(request):
    """部门删除"""

    nid = request.GET.get('nid')
    Department.objects.filter(id=nid).delete()

    # 重定向回部门列表
    return redirect("/depart/list/")
```

修改 `myproject/employee_management/templates/depart_list.html`

```html
<tbody>
    {% for obj in depart_list %}
    <tr>
        <th>{{ obj.id }}</th>
        <td>{{ obj.title }}</td>
        <td>
            <a class="btn btn-primary btn-xs">编辑</a>
            <a class="btn btn-danger btn-xs" href="/depart/delete/?nid={{ obj.id }}">删除</a>
        </td>
    </tr>
    {% endfor %}
</tbody>
```

```
                </div>

                <!-- Table -->
                <table class="table table-bordered">
                    <thead>
                        <tr>
                            <th>ID</th>
                            <th>名称</th>
                            <th>操作</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for obj in depart_list %}
                        <tr>
                            <th>{{ obj.id }}</th>
                            <td>{{ obj.title }}</td>
                            <td>
                                <a class="btn btn-primary btn-xs">编辑</a>
                                <a class="btn btn-danger btn-xs" href="/depart/delete/?nid={{ obj.id }}">删除</a>
                            </td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>

<script src="/static/js/jquery-3.6.1.min.js"></script>
<script src="/static/plugins/bootstrap-3.4.1/js/bootstrap.min.js"></script>
</body>
```

CSDN @ShangCode

部门编辑

修改`myproject/myproject/urls.py`

```python
from django.contrib import admin
from django.urls import path
from employee_management import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('depart/list/', views.depart_list),
    path('depart/add/', views.depart_add),
    path('depart/delete/', views.depart_delete),
    path('depart/<int:nid>/edit/', views.depart_edit),
]
```

修改`myproject/employee_management/views.py`

```python
def depart_edit(request, nid):
    """部门编辑"""

    if request.method == "GET":
        # 根据nid,获取数据
        row_object = Department.objects.filter(id=nid).first()
        return render(request, 'depart_edit.html', {"row_object": row_object})

    # 如果是POST请求,保存修改
    depart_title = request.POST.get('depart_title')
    Department.objects.filter(id=nid).update(title=depart_title)

    # 重定向回部门列表
    return redirect('/depart/list/')
```

新建`myproject/employee_management/templates/depart_edit.html`

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <link rel="stylesheet" href="/static/plugins/bootstrap-3.4.1/css/bootstrap.css">
    <link rel="stylesheet" href="/static/plugins/font-awesome-4.7.0/css/font-awesome.css">

    <style>
        .navbar {
            border-radius: 0;
        }
    </style>

</head>

<body>
    <div>
        <!-- 导航条, https://v3.bootcss.com/components/#navbar -->
        <nav class="navbar navbar-default">
            <div class="container">
                <!-- Brand and toggle get grouped for better mobile display -->
                <div class="navbar-header">
                    <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
                        data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
                        <span class="sr-only">Toggle navigation</span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                    </button>
                    <a class="navbar-brand" href="#">员工管理系统</a>
                </div>

                <!-- Collect the nav links, forms, and other content for toggling -->
                <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                    <ul class="nav navbar-nav">
                        <li><a href="#">部门管理</a></li>
                        <li><a href="#">部门管理</a></li>

                    </ul>
                    <ul class="nav navbar-nav navbar-right">
                        <li><a href="#">注册</a></li>
                        <li class="dropdown">
                            <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button"
                                aria-haspopup="true" aria-expanded="false">poker <span class="caret"></span></a>
                            <ul class="dropdown-menu">
```

```
                                    <li><a href="#">Action</a></li>
                                    <li><a href="#">Another action</a></li>
                                    <li><a href="#">Something else here</a></li>
                                    <li role="separator" class="divider"></li>
                                    <li><a href="#">Separated link</a></li>
                                </ul>
                            </li>
                        </ul>
                    </div><!-- /.navbar-collapse -->
                </div><!-- /.container-fluid -->
            </nav>
        </div>
        <div class="container">
            <div class="panel panel-default">
                <div class="panel-heading">
                    <h3 class="panel-title">编辑部门</h3>
                </div>
                <div class="panel-body">
                    <form action="/depart/{{ row_object.id }}/edit/" method="post">
                        {% csrf_token %}
                        <div class="form-group">
                            <label>部门名称</label>
                            <input type="text" class="form-control" placeholder="部门名称" name="depart_title" value="{{ row_object.title }}">
                        </div>
                        <button type="submit" class="btn btn-primary">保存</button>
                    </form>
                </div>

            </div>
        </div>
    </body>
    </html>

```

修改 `myproject/employee_management/templates/depart_list.html`

```
<tbody>
    {% for obj in depart_list %}
    <tr>
        <th>{{ obj.id }}</th>
        <td>{{ obj.title }}</td>
        <td>
            <a class="btn btn-primary btn-xs" href="/depart/{{ obj.id }}/edit/">编辑</a>
            <a class="btn btn-danger btn-xs" href="/depart/delete/?nid={{ obj.id }}">删除</a>
        </td>
    </tr>
    {% endfor %}
</tbody>
```

浏览器访问 `/depart/list/` ,点击"编辑"

修改后"保存"观察数据变化

模板继承

定义模板: `layout.html`

> {% block content %}{% endblock %}

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <link rel="stylesheet" href="/static/plugins/bootstrap-3.4.1/css/bootstrap.css">
    <link rel="stylesheet" href="/static/plugins/font-awesome-4.7.0/css/font-awesome.css">

    <style>
        .navbar {
            border-radius: 0;
        }
    </style>
</head>

<body>

    <!-- 导航条, https://v3.bootcss.com/components/#navbar -->
```

```
    <nav class="navbar navbar-default">
        <div class="container">
            <!-- Brand and toggle get grouped for better mobile display -->
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
                    data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="#">员工管理系统</a>
            </div>

            <!-- Collect the nav links, forms, and other content for toggling -->
            <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                <ul class="nav navbar-nav">
                    <li><a href="#">部门管理</a></li>
                    <li><a href="#">部门管理</a></li>

                </ul>
                <ul class="nav navbar-nav navbar-right">
                    <li><a href="#">注册</a></li>
                    <li class="dropdown">
                        <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true"
                            aria-expanded="false">poker <span class="caret"></span></a>
                        <ul class="dropdown-menu">
                            <li><a href="#">Action</a></li>
                            <li><a href="#">Another action</a></li>
                            <li><a href="#">Something else here</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">Separated link</a></li>
                        </ul>
                    </li>
                </ul>
            </div><!-- /.navbar-collapse -->
        </div><!-- /.container-fluid -->
    </nav>
    <!-- 新建区域 -->
    <div>
        {% block content %}{% endblock %}
    </div>

    <script src="/static/js/jquery-3.6.1.min.js"></script>
    <script src="/static/plugins/bootstrap-3.4.1/js/bootstrap.min.js"></script>
</body>

</html>
```

继承模板:

```
{% extends 'layout.html' %}

{% block content %}
    <h1>首页</h1>
{% endblock %}
```

例如,可以将`myproject/employee_management/templates/depart_list.html`改成

```
{% extends 'layout.html' %}

{% block content %}
<div class="container">
    <div style="margin-bottom: 10px">
        <a class="btn btn-primary" href="/depart/add/" target="_blank">新建部门</a>
    </div>
    <div>
        <div class="panel panel-default">
            <!-- Default panel contents -->
            <div class="panel-heading">
                <span class="glyphicon glyphicon-th-list" aria-hidden="true" style="margin-right: 5px;"></span>
                <span>部门列表</span>
            </div>

            <!-- Table -->
            <table class="table table-bordered">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>名称</th>
                        <th>操作</th>
                    </tr>
                </thead>
                <tbody>
                    {% for obj in depart_list %}
                    <tr>
                        <th>{{ obj.id }}</th>
                        <td>{{ obj.title }}</td>
                        <td>
                            <a class="btn btn-primary btn-xs" href="/depart/{{ obj.id }}/edit/">编辑</a>
                            <a class="btn btn-danger btn-xs" href="/depart/delete/?nid={{ obj.id }}">删除</a>
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    </div>
</div>

{% endblock %}
```

其余页面以此类推,请自行修改

> 如果有个别页面不需要用到指定的css样式,可以这么做

```
{% block css %}{% endblock %}
```

在 `layout.html` 模板中将css区域独立出来

用户管理

```
mysql> desc employee_management_userinfo;
+-------------+---------------+------+-----+---------+----------------+
| Field       | Type          | Null | Key | Default | Extra          |
+-------------+---------------+------+-----+---------+----------------+
| id          | bigint(20)    | NO   | PRI | NULL    | auto_increment |
| name        | varchar(16)   | NO   |     | NULL    |                |
| password    | varchar(64)   | NO   |     | NULL    |                |
| age         | int(11)       | NO   |     | NULL    |                |
| account     | decimal(10,2) | NO   |     | NULL    |                |
| create_time | datetime(6)   | NO   |     | NULL    |                |
| gender      | smallint(6)   | NO   |     | NULL    |                |
| depart_id   | bigint(20)    | NO   | MUL | NULL    |                |
+-------------+---------------+------+-----+---------+----------------+
```

向用户数据表中插入几行数据方便后面进行测试

```
insert into employee_management_userinfo(name,password,age,account,create_time,gender,depart_id) values("李云龙","123456",45,50000,"2020-03-24",1,2);
insert into employee_management_userinfo(name,password,age,account,create_time,gender,depart_id) values("张三丰","123456",45,60000,"2021-03-24",1,3);
insert into employee_management_userinfo(name,password,age,account,create_time,gender,depart_id) values("周杰伦","123456",45,70000,"2022-03-24",1,4);
```

用户列表

修改 `myproject/myproject/urls.py`

```
from django.contrib import admin
from django.urls import path
from employee_management import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('depart/list/', views.depart_list),
    path('depart/add/', views.depart_add),
    path('depart/delete/', views.depart_delete),
    path('depart/<int:nid>/edit/', views.depart_edit),
    path('user/list/', views.user_list),
]
```

修改`myproject/employee_management/views.py`

```python
def user_list(request):

    # 获取所有用户列表
    user_data = UserInfo.objects.all()

    # 用 python 的语法获取数据
    """
    for obj in user_data:
        # obj.get_gender_display() 表示匹配 男/女,原始字段名为gender,obj.get_字段名称_display()
        # obj.create_time.strftime("%Y-%m-%d") 表示将时间格式转换成固定格式的字符串
        # obj.depart.title 表示获取depart_id对应的部门名称,因为我们在models中定义表时与另外一张表设置了级联关系,有外键
        print(obj.id, obj.name, obj.password, obj.age, obj.account, obj.get_gender_display(), obj.depart.title, obj.create_time.strftime("%Y-%m-%d"))
    """

    return render(request, "user_list.html", {"user_data": user_data})

```

新建`myproject/employee_management/templates/user_list.html`

> 注意: HTML 中获取数据的方式与 Python 中有些不同
> 例如:
> 1.HTML中引入函数不能带括号, obj.get_gender_display()
> 2.日期类型转字符串有Django自己的格式, obj.create_time|date:“Y-m-d”

```
{% extends 'layout.html' %}

{% block content %}
<div class="container">
    <div style="margin-bottom: 10px">
        <a class="btn btn-primary" href="/depart/add/" target="_blank">新建用户</a>
    </div>
    <div>
        <div class="panel panel-default">
            <!-- Default panel contents -->
            <div class="panel-heading">
                <span class="glyphicon glyphicon-th-list" aria-hidden="true" style="margin-right: 5px;"></span>
                <span>用户列表</span>
            </div>

            <!-- Table -->
            <table class="table table-bordered">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>姓名</th>
                        <th>密码</th>
                        <th>年龄</th>
                        <th>性别</th>
```

```
                        <th>账户余额</th>
                        <th>入职时间</th>
                        <th>部门</th>
                        <th>操作</th>
                    </tr>
                </thead>
                <tbody>
                    {% for obj in user_data %}
                    <tr>
                        <th>{{ obj.id }}</th>
                        <td>{{ obj.name }}</td>
                        <td>{{ obj.password }}</td>
                        <td>{{ obj.age }}</td>
                        <td>{{ obj.get_gender_display }}</td>
                        <td>{{ obj.account }}</td>
                        <td>{{ obj.create_time|date:"Y-m-d" }}</td>
                        <td>{{ obj.depart.title }}</td>
                        <td>
                            <a class="btn btn-primary btn-xs" href="/user/{{ obj.id }}/edit/">编辑</a>
                            <a class="btn btn-danger btn-xs" href="/user/delete/?nid={{ obj.id }}">删除</a>
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    </div>
</div>

{% endblock %}
```

浏览器进行访问测试

用户添加

> 这里不演示最无脑原始的方式,漏洞百出,因为:
>
> - 数据校验较麻烦
> - 页面没有错误提示
> - 页面上每一个字段都需要重新写一遍
> - 关联的数据,需要手动获取并循环展示在页面

Django组件:

- Form组件(简便)
- ModelForm组件(最简便)

初识Form

- views.py

```
class MyForm(Form):
    user = forms.CharField(widget=forms.Input)
    pwd = forms.CharField(widget=forms.Input)
    email = forms.CharField(widget=forms.Input)

def user_add(request):
    if request.method == "GET":
    form = MyForm()
        return render(request, "user_add.html", {"form": form})
```

- user_add.html

> {{ form.xxx }} 可以自动生成前端代码

```
<form method="post">
    {{ form.user }}
    {{ form.pwd }}
    {{ form.email }}
</form>
```

> 也可以不指定,自动生成全部

```
<form method="post">
    {% for field in form %}
        {{ field }}
    {% endfor %}
</form>
```

ModelForm

- models.py

```
from django.db import models

# Create your models here.
class Department(models.Model):
    """部门表"""
    title = models.CharField(max_length=32, verbose_name='标题')


class UserInfo(models.Model):
    """员工表"""
    name = models.CharField(max_length=16, verbose_name="姓名")
    password = models.CharField(max_length=64, verbose_name="密码")
    age = models.IntegerField(verbose_name="年龄")
    account = models.DecimalField(verbose_name="账户余额", max_digits=10,
decimal_places=2, default=0)
```

```
    create_time = models.DateTimeField(verbose_name="入职时间")
    depart = models.ForeignKey(to="Department", to_field="id", on_delete=models.CASCADE, verbose_name="部门")

    gender_choices = (
        (1, "男"),
        (2, "女"),
    )
    gender = models.SmallIntegerField(choices=gender_choices,verbose_name="性别")

```

- views.py

```
class MyForm(ModelForm):
    class Meta:
        field = ["name", "password", "age"]

def user_add(request):
    if request.method == "GET":
    form = MyForm()
        return render(request, "user_add.html", {"form": form})
```

- user_add.html

> {{ form.xxx }} 可以自动生成前端代码

```
<form method="post">
    {{ form.user }}
    {{ form.pwd }}
    {{ form.email }}
</form>
```

> 也可以不指定,自动生成全部

```
<form method="post">
    {% for field in form %}
        {{ field }}
    {% endfor %}
</form>
```

> 获取标签

```
<form method="post">
    {% csrf_token %}
    {{ form.name.label }}:{{ form.name }}
    {{ form.password.label }}:{{ form.password }}
    {{ form.age.label }}:{{ form.age }}
</form>
```

用户添加(ModelForm)

修改`myproject/employee_management/views.py`

```python
########################## ModelForm 演示 ##############################

from django import forms

class UserModelForm(forms.ModelForm):

    ### 自定义数据校验
    # 例如: 用户名最小三个字符
    #name = forms.CharField(min_length=3, label="用户名")

    class Meta:
        model = UserInfo
        fields = ["name", "password", "age", "account", "create_time", "gender", "depart"]
        # 逐一控制标签的样式
        # widgets = {
        #     "name": forms.TextInput(attrs={"class": "form-control"}),
        #     "password": forms.PasswordInput(attrs={"class": "form-control"}),
        # }

        # 这里让日期可以手动点击鼠标选择,所以单独拎出来,加上日期插件
        widgets = {
            "create_time": forms.DateTimeInput(attrs={'class': 'form-control', 'id': 'myDate'}),
        }

    # 循环找到所有的插件,添加 "class": "form-control"
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        for name, field in self.fields.items():
            # 可以排除指定的字段
            if name == "create_time":
                continue
            print(name, field)
            field.widget.attrs = {"class": "form-control"}


def user_model_form_add(request):
    """添加用户(ModelForm版本)"""
    if request.method == "GET":
        form = UserModelForm()
        return render(request, "user_model_form_add.html", {"form": form})

    # 用户POST请求提交数据,需要进行数据校验
    form = UserModelForm(data=request.POST)
    if form.is_valid():
        print(form.cleaned_data)
        # 直接保存至数据库
        form.save()
        return redirect("/user/list/")

    # 校验失败(在页面上显示错误信息)
    return render(request, "user_model_form_add.html", {"form": form})
```

修改 `myproject/myproject/urls.py`

```
from django.contrib import admin
from django.urls import path
from employee_management import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('depart/list/', views.depart_list),
    path('depart/add/', views.depart_add),
    path('depart/delete/', views.depart_delete),
    path('depart/<int:nid>/edit/', views.depart_edit),
    path('user/list/', views.user_list),
    path('user/model/form/add/', views.user_model_form_add),
]
```

新建 `myproject/employee_management/templates/user_model_form_add.html`

```
{% extends 'layout.html' %}

{% block content %}
<div class="container">
    <div class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-title">添加用户</h3>
        </div>
        <div class="panel-body">
            <form action="/user/model/form/add/" method="post" novalidate>
                {% csrf_token %}

                {% for field in form %}
                    <div class="form-group">
                        <label>{{ field.label }}: </label>
                        {{ field }}
                        <!-- 数据校验,显示错误信息 -->
                        <span style="color: red;">{{ field.errors.0 }}</span>
                    </div>
                {% endfor %}

                <button type="submit" class="btn btn-primary">保存</button>
            </form>
        </div>
    </div>
</div>

{% endblock %}
</body>
</html>
```

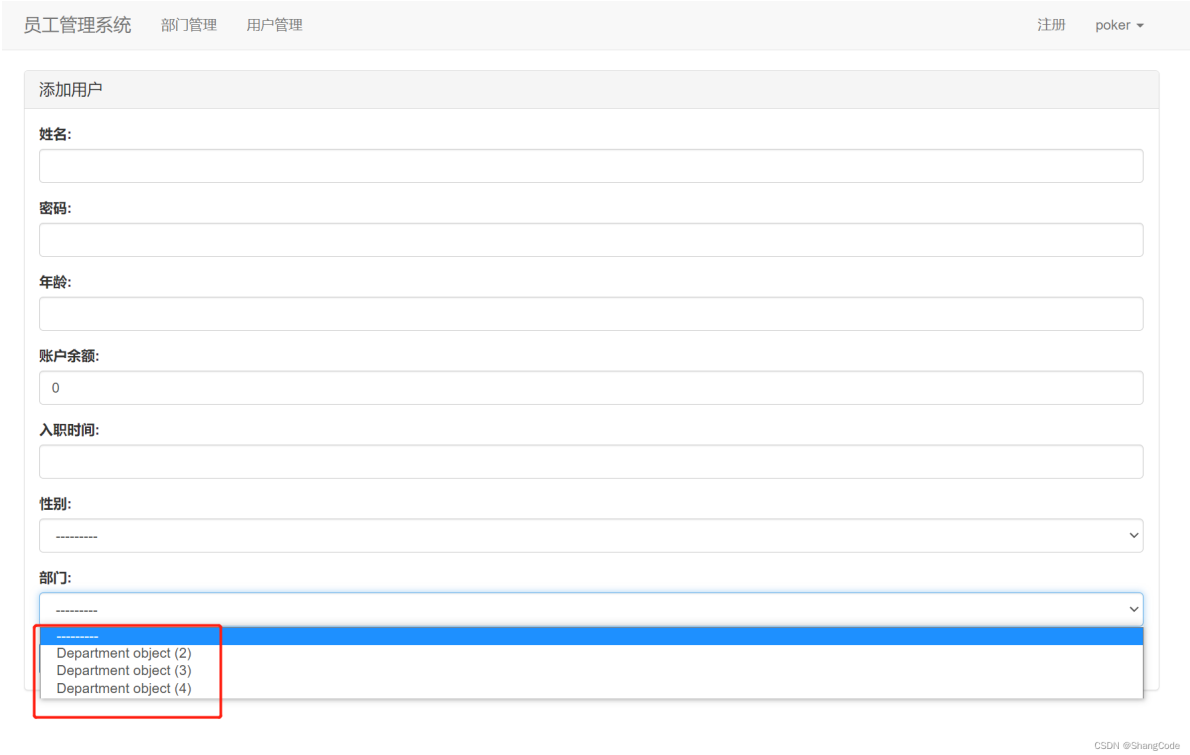
> form 默认保存的是用户输入的所有数据 如果想要再用户输入以外增加一点值

```
form.instance.字段值 = 值
```

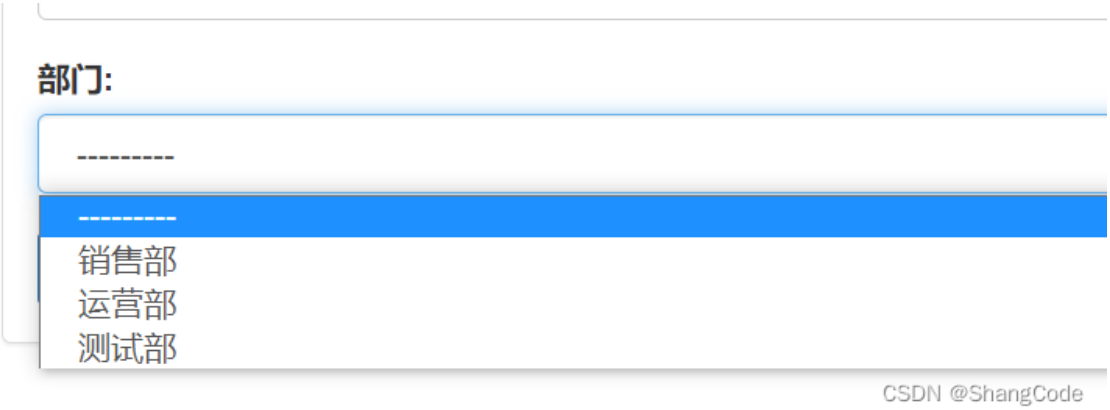
修改 `myproject/employee_management/models.py`

> 目的是让自动生成的部门字段不显示"对象"本身,显示对象对应的"title"

```python
class Department(models.Model):
    """部门表"""
    title = models.CharField(max_length=32, verbose_name='标题')

    def __str__(self):
        return self.title
123456
```



修改后再次刷新

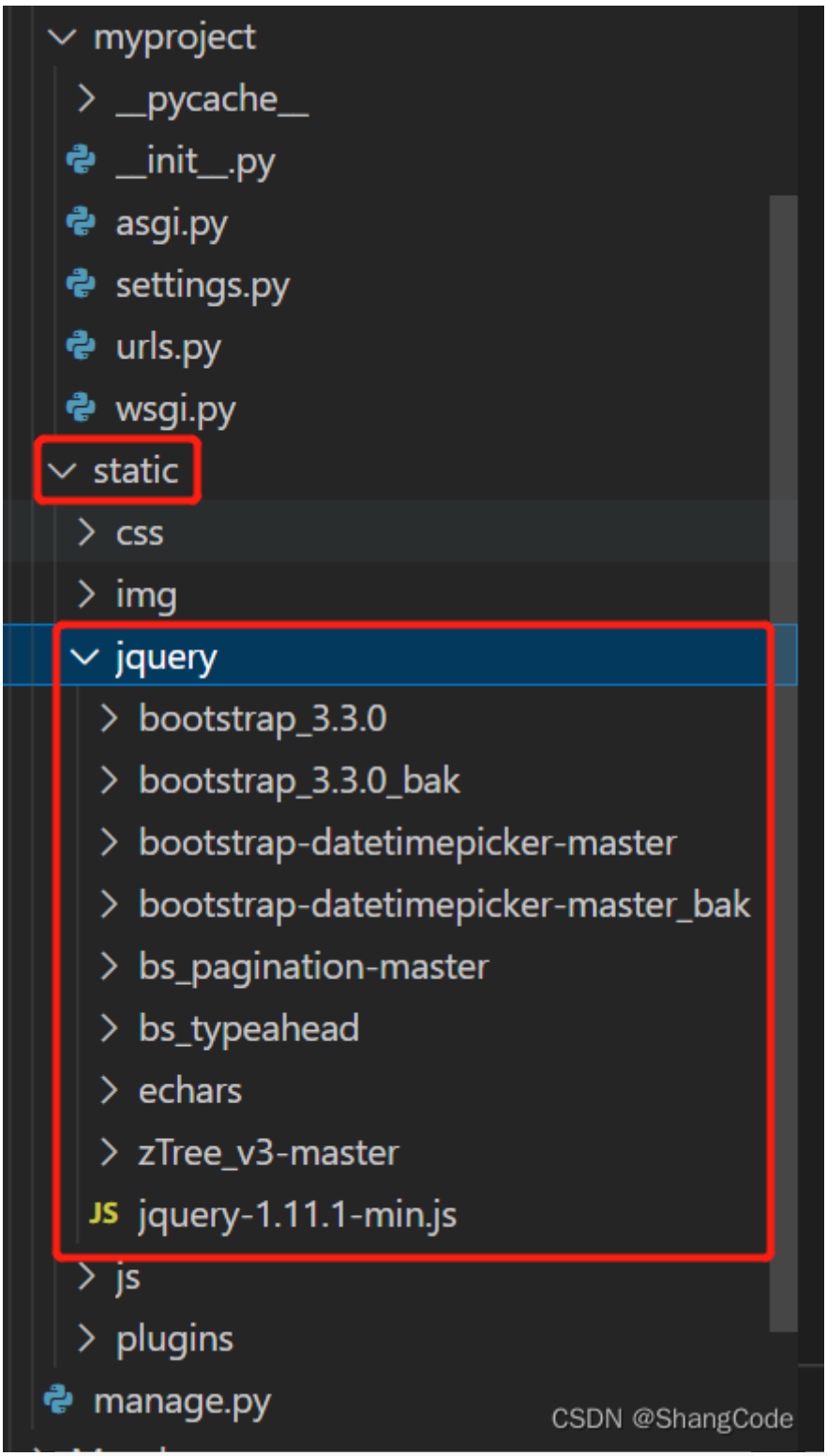


日期设置

目前日期只能手动输入,如果想要鼠标点击选择,需要调用 `datetimepicker` 插件

插件下载地址:
链接: https://pan.baidu.com/s/1yN-L7bhwdSXwfYfh2MUj2A
提取码: yyds

> 下载完成后,我将插件放在了 `/root/python/myproject/static/` 下



修改 `myproject/employee_management/templates/layout.html` 引入 `datetimepicker` 插件

> 调用方法: 在对应的标签中加入"id=myDate"

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <link rel="stylesheet" href="/static/plugins/bootstrap-
3.4.1/css/bootstrap.css">
```

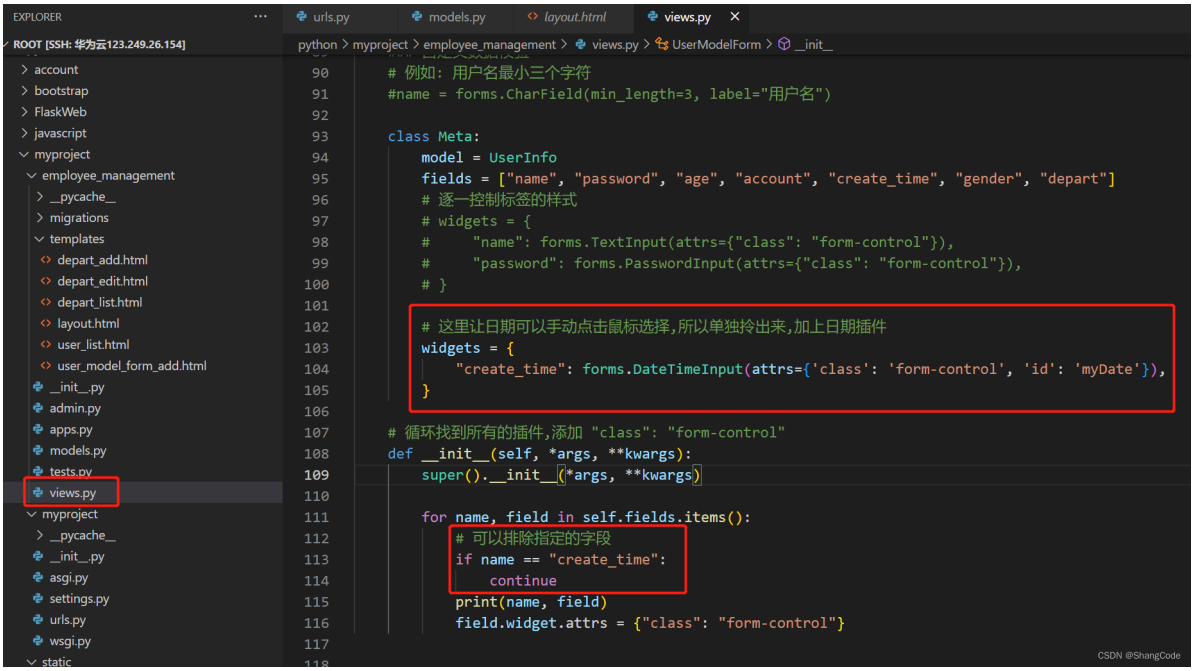
```
    <link rel="stylesheet" href="/static/plugins/font-awesome-4.7.0/css/font-awesome.css">


    <!--JQUERY-->
    <script type="text/javascript" src="/static/jquery/jquery-1.11.1-min.js"></script>
    <!--BOOTSTRAP框架-->
    <link rel="stylesheet" type="text/css" href="/static/jquery/bootstrap_3.3.0/css/bootstrap.min.css">
    <script type="text/javascript" src="/static/jquery/bootstrap_3.3.0/js/bootstrap.min.js"></script>
    <!--BOOTSTRAP_DATETIMEPICKER插件-->
    <link rel="stylesheet" type="text/css" href="/static/jquery/bootstrap-datetimepicker-master/css/bootstrap-datetimepicker.min.css">
</head>
<body>

    <!-- 此处省略一部分代码 -->

    <script type="text/javascript">
        $(function () {
            //当容器加载完成，对容器调用工具函数
            $("#myDate").datetimepicker({
                language: 'zh-CN', //语言
                format: 'yyyy-mm-dd',//日期的格式
                minView: 'month', //可以选择的最小视图
                initialDate: new Date(),//初始化显示的日期
                autoclose: true,//设置选择完日期或者时间之后，日否自动关闭日历
                todayBtn: true,//设置自动显示为今天
                clearBtn: false//设置是否清空按钮，默认为false
            });
        });
    </script>
</body>
</html>
12345678910111213141516171819202122232425262728293031323334353637383940
```

如何使用呢

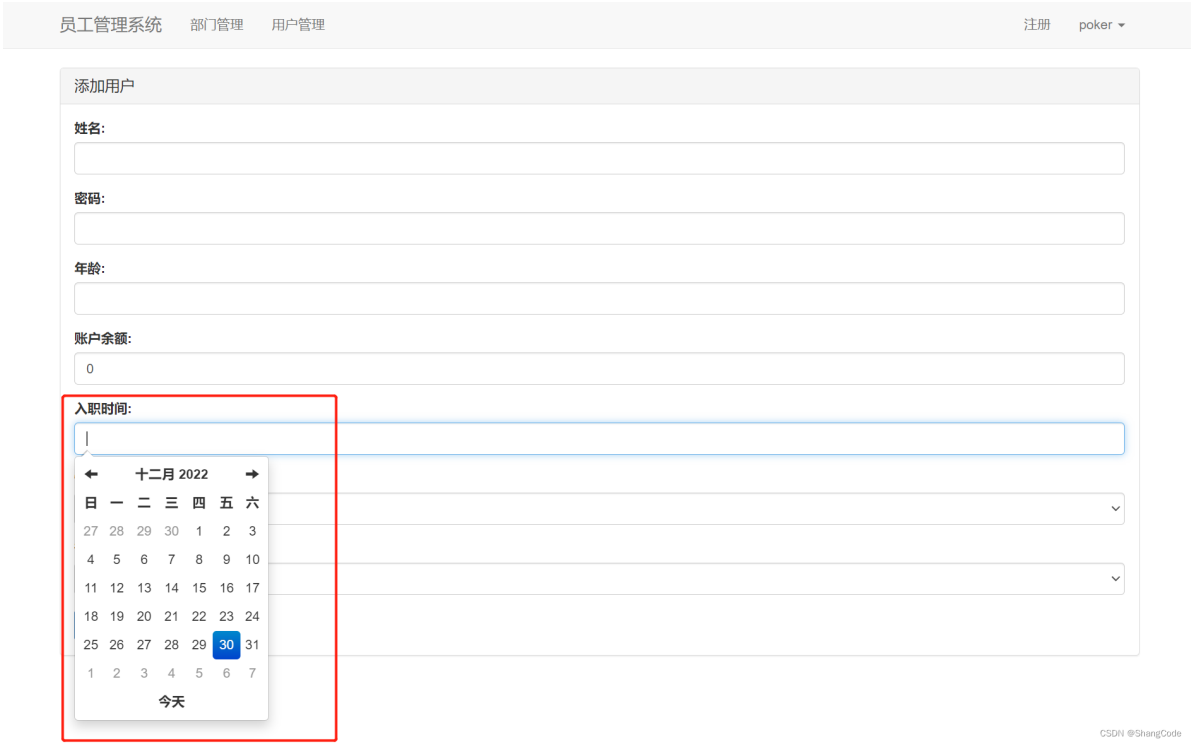
其实在上面的代码中我已经提前写上了

下图是Django中ModelForm修改标签属性的方式,在 `widgets` 字典中定义

`attrs={'class': 'form-control', 'id': 'myDate'}`



最终效果



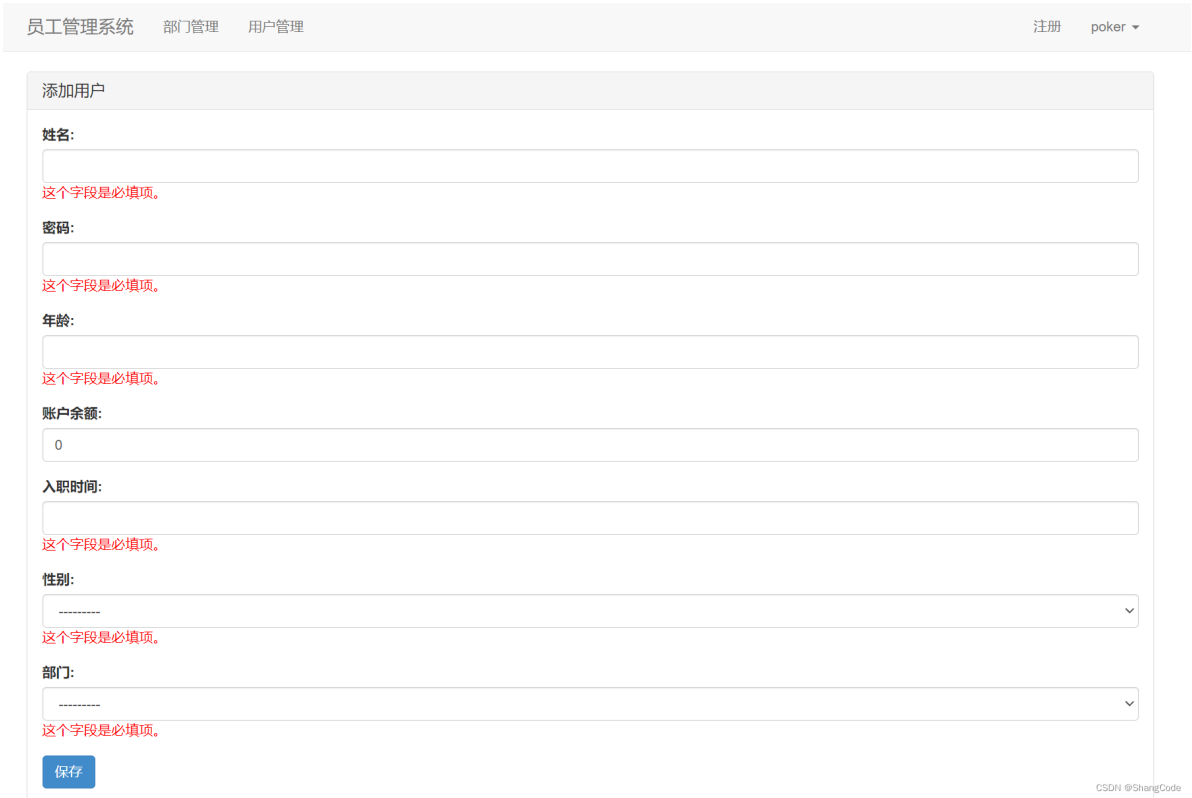
数据校验错误提示

修改 `myproject/myproject/settings.py`

> 改为中文,目的是为了让页面提示错误信息时显示中文,否则会显示英文

```
#LANGUAGE_CODE = 'en-us'
LANGUAGE_CODE = 'zh-hans'
12
```

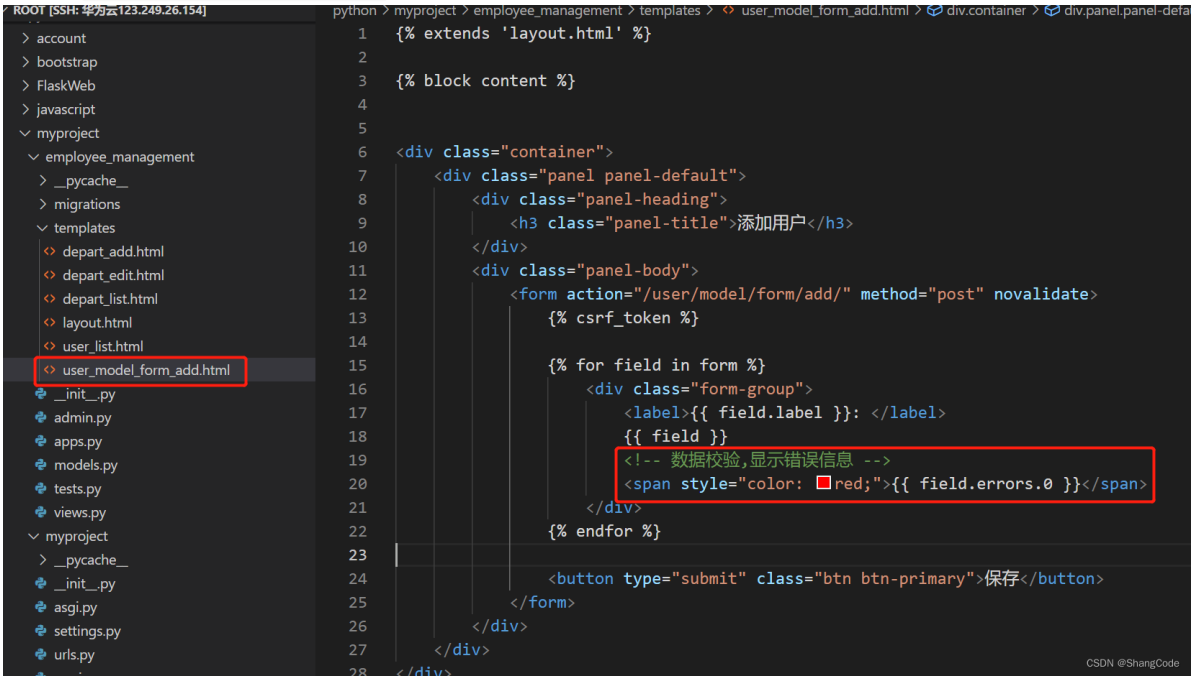
如果我们没有写入任何内容直接点击"保存",那么页面会提示错误



因为我们

在 `/root/python/myproject/employee_management/templates/user_model_form_add.html` 中

加入了 `{{ field.errors.0 }}` 字段



编辑用户

修改 `myproject/employee_management/views.py`

```
def user_edit(request, nid):
    """编辑用户"""

    row_obj = UserInfo.objects.filter(id=nid).first()

```

```python
    # GET请求
    if request.method == "GET":
        form = UserModelForm(instance=row_obj)
        return render(request, "user_edit.html", {"form": form})

    # POST请求
    form = UserModelForm(data=request.POST, instance=row_obj)
    if form.is_valid():
        form.save()
        return redirect("/user/list/")

    return render(request, "user_edit.html", {"form": form})
```

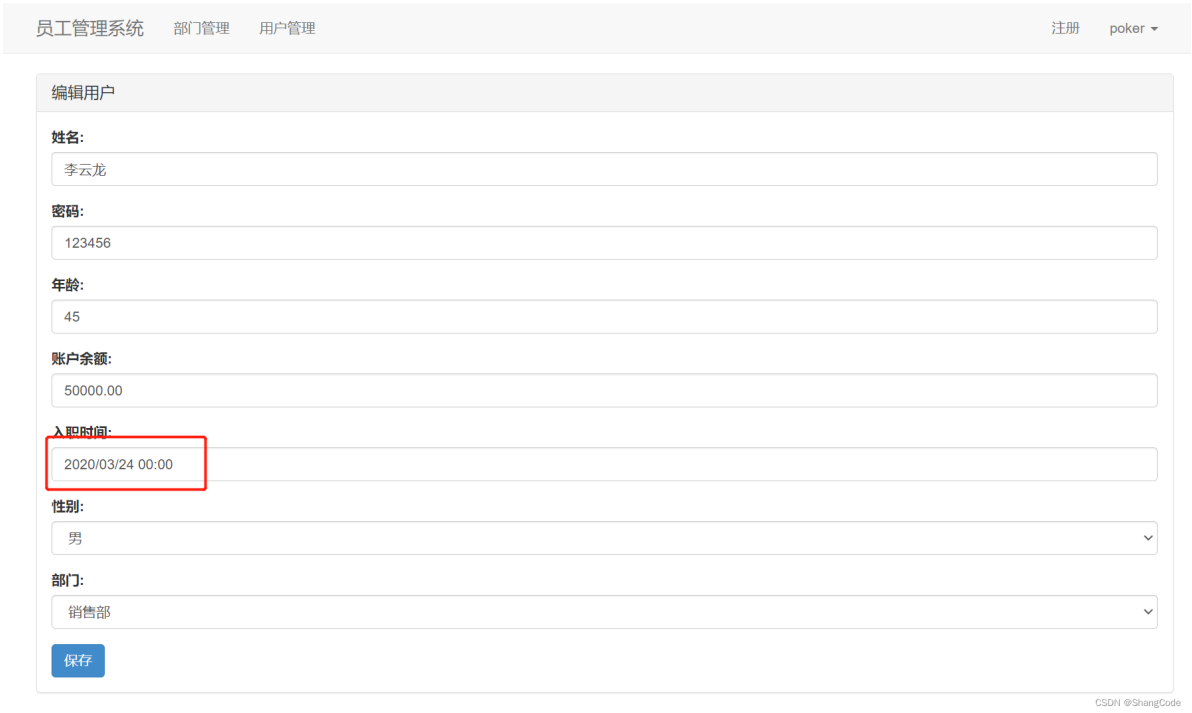
修改 `/root/python/myproject/myproject/urls.py`

```python
path('user/<int:nid>/edit/', views.user_edit),
```

加 `/root/python/myproject/employee_management/templates/user_edit.html`

```
{% extends 'layout.html' %}

{% block content %}

<div class="container">
    <div class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-title">编辑用户</h3>
        </div>
        <div class="panel-body">
            <form method="post" novalidate>
                {% csrf_token %}

                {% for field in form %}
                    <div class="form-group">
                        <label>{{ field.label }}: </label>
                        {{ field }}
                        <!-- 数据校验,显示错误信息 -->
                        <span style="color: red;">{{ field.errors.0 }}</span>
                    </div>
                {% endfor %}

                <button type="submit" class="btn btn-primary">保存</button>
            </form>
        </div>
    </div>
</div>
{% endblock %}
</body>
</html>
```

浏览器访问测试,点击"编辑"



但是发现上面的时间有些问题,应该只显示年月日就可以了,不应该显示时分秒

需要修改数据库models

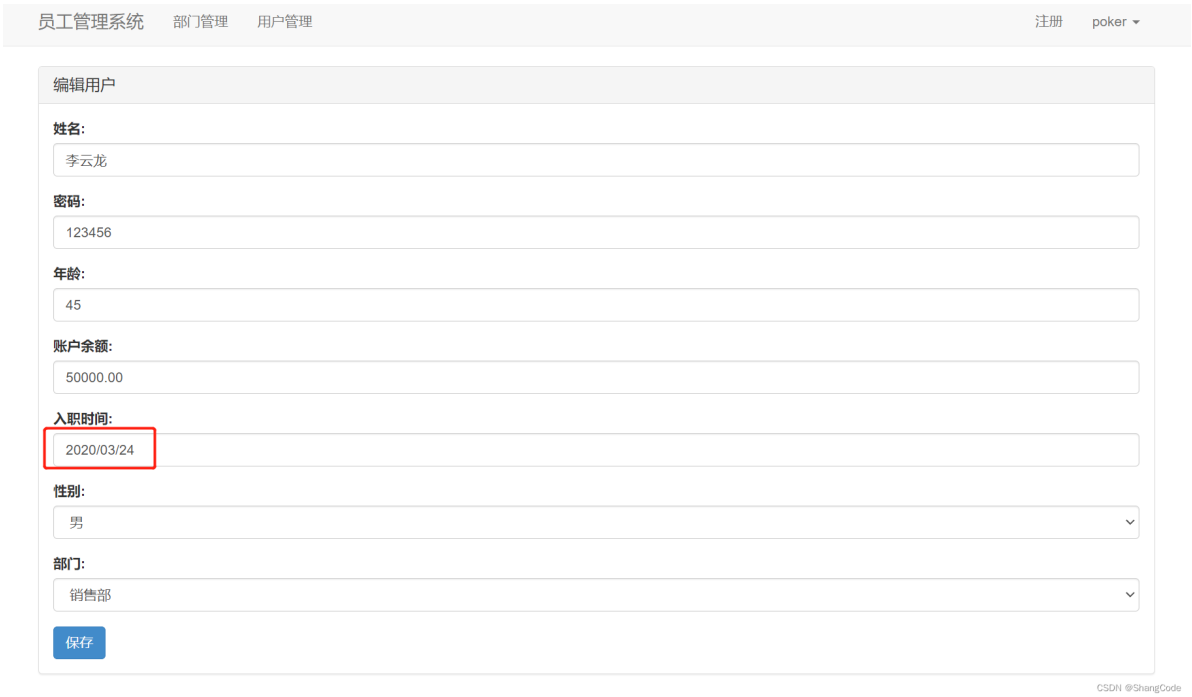
更新**数据库表结构**

```
python3 manage.py makemigrations
python3 manage.py migrate
12
```

```
[root@hecs-33592 ~]# cd python/
[root@hecs-33592 python]# cd myproject/
[root@hecs-33592 myproject]# python3 manage.py makemigrations
Migrations for 'employee_management':
  employee_management/migrations/0002_alter_userinfo_create_time.py
    - Alter field create_time on userinfo
[root@hecs-33592 myproject]# python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, employee_management, sessions
Running migrations:
  Applying employee_management.0002_alter_userinfo_create_time... OK
```


此时还需要更改一个地方

浏览器刷新



删除用户

修改`myproject/employee_management/views.py`

```python
def user_delete(request, nid):
    """用户删除"""
    UserInfo.objects.filter(id=nid).delete()
    return redirect("/user/list/")
```

修改`myproject/myproject/urls.py`

```
path('user/<int:nid>/delete/', views.user_delete),
```

修改`myproject/employee_management/templates/user_list.html`

```
<td>
    <a class="btn btn-primary btn-xs" href="/user/{{ obj.id }}/edit/">编辑</a>
    <a class="btn btn-danger btn-xs" href="/user/{{ obj.id }}/delete/">删除</a>
</td>
```

浏览器测试

靓号管理

表结构

修改`myproject/employee_management/models.py`

```
class PrettyNum(models.Model):
    """靓号表"""
    # 如果想要为空 null=True blank=True
    mobile = models.CharField(verbose_name="手机号", max_length=32)
    price = models.IntegerField(verbose_name="价格", default=0)

    level_choices = (
        (1, "1级"),
        (2, "2级"),
        (3, "3级"),
        (4, "4级"),
    )
    level = models.SmallIntegerField(verbose_name="级别", choices=level_choices, default=1)

    status_choices = (
        (1, "已占用"),
        (2, "未使用"),
    )
    status = models.SmallIntegerField(verbose_name="状态", choices=status_choices, default=2)
```

生成数据库表

```
python3 manage.py makemigrations
python3 manage.py migrate
```

```
mysql> use my_project;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+--------------------------------+
| Tables_in_my_project           |
+--------------------------------+
| auth_group                     |
| auth_group_permissions         |
| auth_permission                |
| auth_user                      |
| auth_user_groups               |
| auth_user_user_permissions     |
| django_admin_log               |
| django_content_type            |
| django_migrations              |
| django_session                 |
| employee_management_department |
| employee_management_prettynum  |
| employee_management_userinfo   |
+--------------------------------+
13 rows in set (0.00 sec)
```


手动模拟添加一些数据

```
1   insert into employee_management_prettynum(mobile, price, level, status)
    values("15811223344", 500, 2, 2);
2   insert into employee_management_prettynum(mobile, price, level, status)
    values("13846783361", 100, 3, 2);
```

```
mysql> select * from employee_management_prettynum;
+----+-------------+-------+-------+--------+
| id | mobile      | price | level | status |
+----+-------------+-------+-------+--------+
|  1 | 15811223344 |   500 |     2 |      2 |
|  2 | 13846783361 |   100 |     3 |      2 |
+----+-------------+-------+-------+--------+
2 rows in set (0.00 sec)
```


靓号列表

> 相信到这里大家可以做到这里,可以说是已经轻车熟路了

修改 `myproject/employee_management/views.py` 增加 `pretty_list` 函数

```python
def pretty_list(request):
    """靓号列表"""

    # select * from 表 by level desc;
    pretty_data = PrettyNum.objects.all().order_by("-level")

    return render(request, "pretty_list.html", {"pretty_data": pretty_data})
```

修改 `myproject/myproject/urls.py`

```python
path('pretty/list/', views.pretty_list),
```

新建 `myproject/employee_management/templates/pretty_list.html`

```
{% extends 'layout.html' %}

{% block content %}
<div class="container">
    <div style="margin-bottom: 10px">
        <a class="btn btn-primary" href="/user/model/form/add/" target="_blank">新建靓号</a>
    </div>
    <div>
        <div class="panel panel-default">
            <!-- Default panel contents -->
            <div class="panel-heading">
                <span class="glyphicon glyphicon-th-list" aria-hidden="true" style="margin-right: 5px;"></span>
                <span>靓号列表</span>
            </div>

            <!-- Table -->
            <table class="table table-bordered">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>号码</th>
                        <th>价格</th>
                        <th>级别</th>
                        <th>状态</th>
                        <th>操作</th>
                    </tr>
                </thead>
                <tbody>
                    {% for obj in pretty_data %}
                    <tr>
                        <th>{{ obj.id }}</th>
                        <td>{{ obj.mobile }}</td>
                        <td>{{ obj.price }}</td>
                        <td>{{ obj.get_level_display }}</td>
                        <td>{{ obj.get_status_display }}</td>
                        <td>
                            <a class="btn btn-primary btn-xs" href="/user/{{ obj.id }}/edit/">编辑</a>
                            <a class="btn btn-danger btn-xs" href="/user/{{ obj.id }}/delete/">删除</a>
```

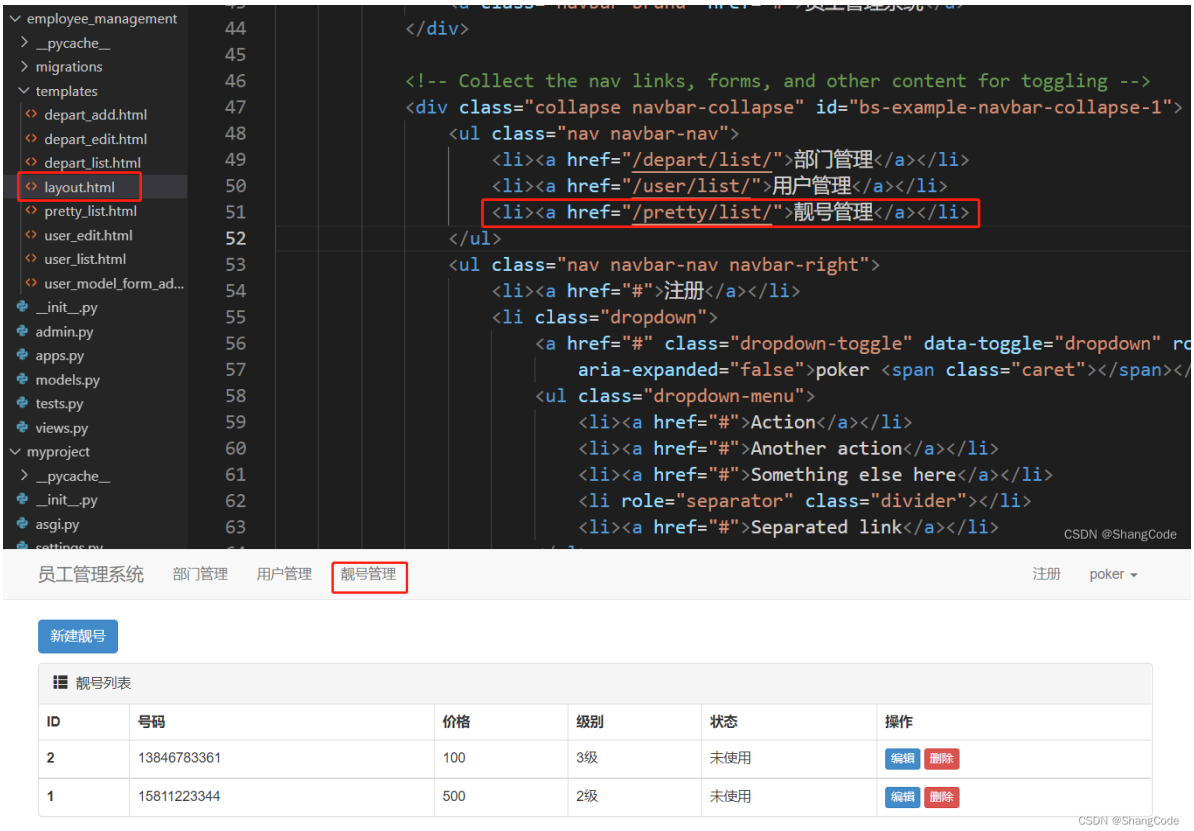
```
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    </div>
</div>

{% endblock %}
```

运行,浏览器访问测试

> 这里在导航栏中增加一个"靓号管理"的标签

修改`myproject/employee_management/templates/layout.html`

```
<li><a href="/pretty/list/">靓号管理</a></li>
```



靓号添加

修改`myproject/employee_management/views.py`

```python
from django.core.validators import RegexValidator

class PrettyModelForm(forms.ModelForm):

    # 数据校验
    mobile = forms.CharField(
        label="手机号",
        validators=[RegexValidator(r'^1[3-9]\d{9}$', '手机号格式错误'),],
    )

    class Meta:
        model = PrettyNum
        # fields = "__all__"    表示取表中所有的字段
        fields = ['mobile', 'price', 'level', 'status']
        # exclude = ['level']   表示取除了表中的某个字段的所有字段

    # 循环找到所有的插件,加入css样式,添加 "class": "form-control"
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        for name, field in self.fields.items():
            print(name, field)
            field.widget.attrs = {"class": "form-control"}

def pretty_add(request):
    """添加靓号"""

    if request.method == "GET":
        form = PrettyModelForm()
        return render(request, "pretty_add.html", {"form": form})

    # 用户POST请求提交数据,需要进行数据校验
    form = PrettyModelForm(data=request.POST)
    if form.is_valid():
        print(form.cleaned_data)
        # 直接保存至数据库
        form.save()
        return redirect("/pretty/list/")

    # 校验失败(在页面上显示错误信息)
    return render(request, "pretty_add.html", {"form": form})
```

修改 `/root/python/myproject/myproject/urls.py`

```python
path('pretty/add/', views.pretty_add),
```

修改 `myproject/employee_management/templates/pretty_list.html`

> 修改 href

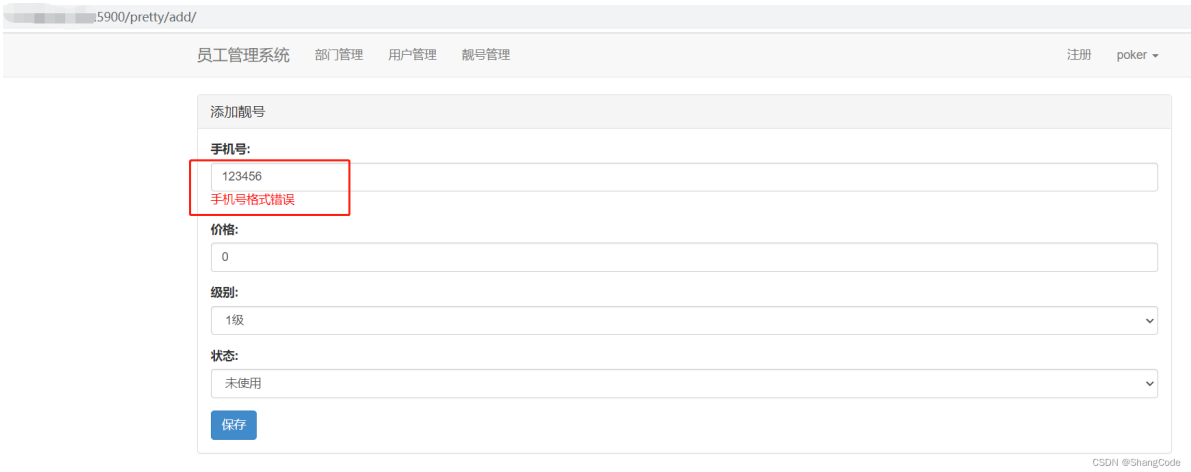
```html
<a class="btn btn-primary" href="/pretty/add/" target="_blank">新建靓号</a>
```

```
{% block content %}
<div class="container">
    <div style="margin-bottom: 10px">
        <a class="btn btn-primary" href="/pretty/add/" target="_blank">新建靓号</a>
    </div>
    <div>
        <div class="panel panel-default">
            <!-- Default panel contents -->
```

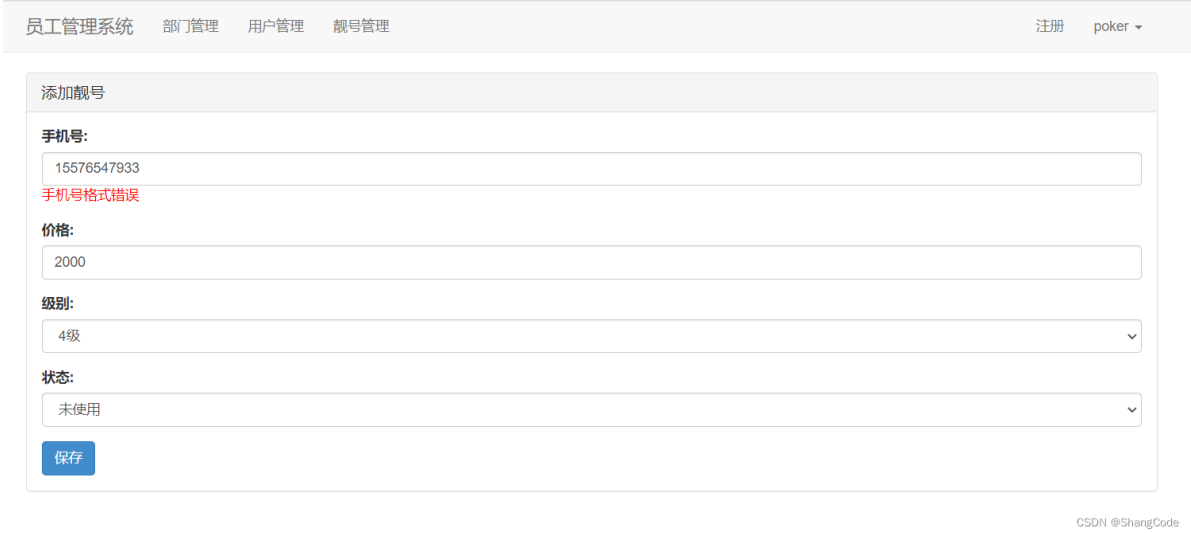

新建 `myproject/employee_management/templates/pretty_add.html`

```
{% extends 'layout.html' %}

{% block content %}


<div class="container">
    <div class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-title">添加靓号</h3>
        </div>
        <div class="panel-body">
            <form action="/pretty/add/" method="post" novalidate>
                {% csrf_token %}

                {% for field in form %}
                    <div class="form-group">
                        <label>{{ field.label }}: </label>
                        {{ field }}
                        <!-- 数据校验,显示错误信息 -->
                        <span style="color: red;">{{ field.errors.0 }}</span>
                    </div>
                {% endfor %}

                <button type="submit" class="btn btn-primary">保存</button>
            </form>
        </div>
    </div>
</div>

{% endblock %}

```

浏览器访问测试



格式只能11位手机号



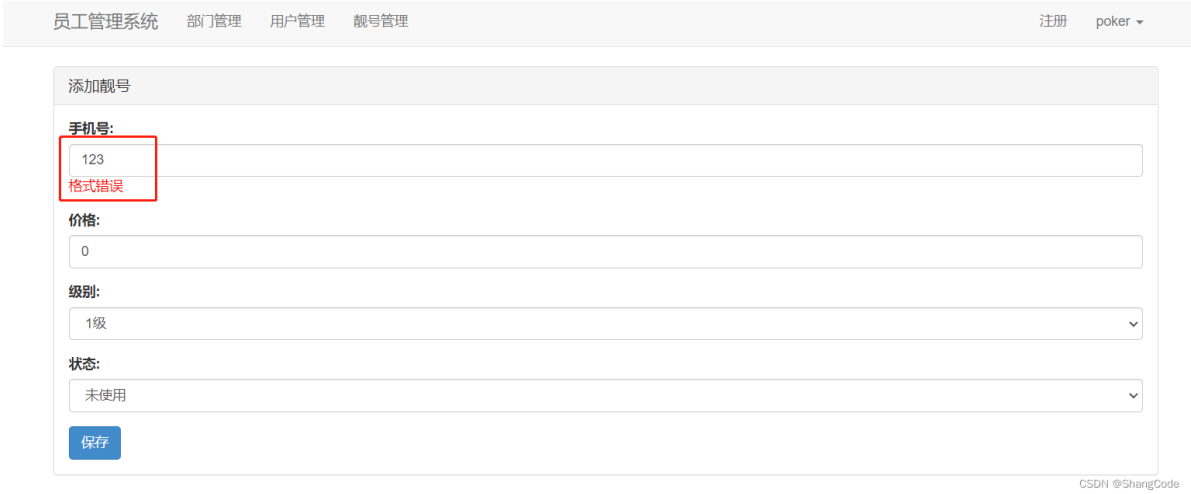
点击保存

> 数据校验的方式还有另外一种,更为复杂的

修改 `myproject/employee_management/views.py`

```python
from django.core.exceptions import ValidationError

class PrettyModelForm(forms.ModelForm):

    # 此处省略中间内容
    ...

    # 数据校验：验证方式2
    def clean_mobile(self):
        txt_mobile = self.cleaned_data['mobile']
```

```
11
12          if len(txt_mobile) != 11:
13              # 验证不通过
14              raise ValidationError('格式错误')
15
16          # 验证通过
17          return txt_mobile
```



同样的效果,选择其中一种方式即可

靓号编辑

> **重新复习一下编辑实现的逻辑:**
>
> 1. 列表页面点击"编辑"按钮
> 2. 匹配 `urls.py` 中的定义的path
> 3. `views.py` 中定义的函数接收到POST请求提交的 `nid`
> 4. 函数根据 `nid` 找到对应的数据行
> 5. `templates` 模板下的HTML文件根据Django的方法展示数据

修改 `myproject/employee_management/views.py`

```
1   def pretty_edit(request, nid):
2       """编辑靓号"""
3       row_obj = PrettyNum.objects.filter(id=nid).first()
4
5       # GET请求
6       if request.method == "GET":
7           form = PrettyModelForm(instance=row_obj)
8           return render(request, "pretty_edit.html", {"form": form})
9
10      # POST请求
11      form = PrettyModelForm(data=request.POST, instance=row_obj)
12      if form.is_valid():
13          form.save()
14          return redirect("/pretty/list/")
15
16      return render(request, "pretty_edit.html", {"form": form})
```

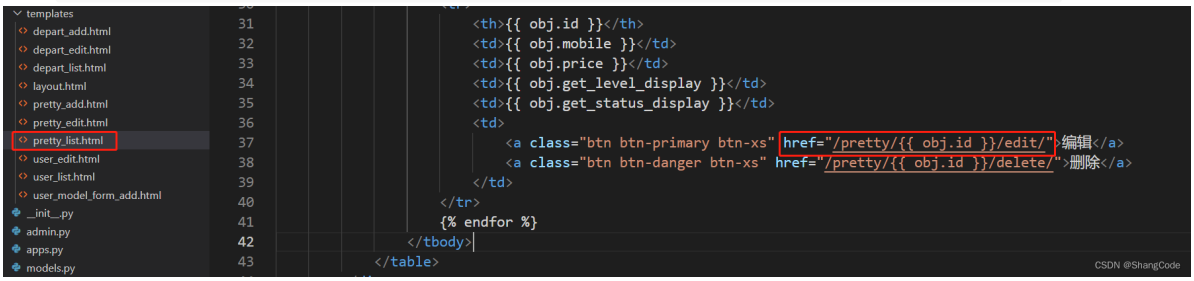
修改 `myproject/myproject/urls.py`

```
path('pretty/<int:nid>/edit/', views.pretty_edit),
```

新建 `/root/python/myproject/employee_management/templates/pretty_edit.html`

```
{% extends 'layout.html' %}

{% block content %}


<div class="container">
    <div class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-title">编辑靓号</h3>
        </div>
        <div class="panel-body">
            <form method="post" novalidate>
                {% csrf_token %}

                {% for field in form %}
                    <div class="form-group">
                        <label>{{ field.label }}: </label>
                        {{ field }}
                        <!-- 数据校验,显示错误信息 -->
                        <span style="color: red;">{{ field.errors.0 }}</span>
                    </div>
                {% endfor %}

                <button type="submit" class="btn btn-primary">保存</button>
            </form>
        </div>
    </div>
</div>

{% endblock %}

```

修改 `/root/python/myproject/employee_management/templates/pretty_list.html`



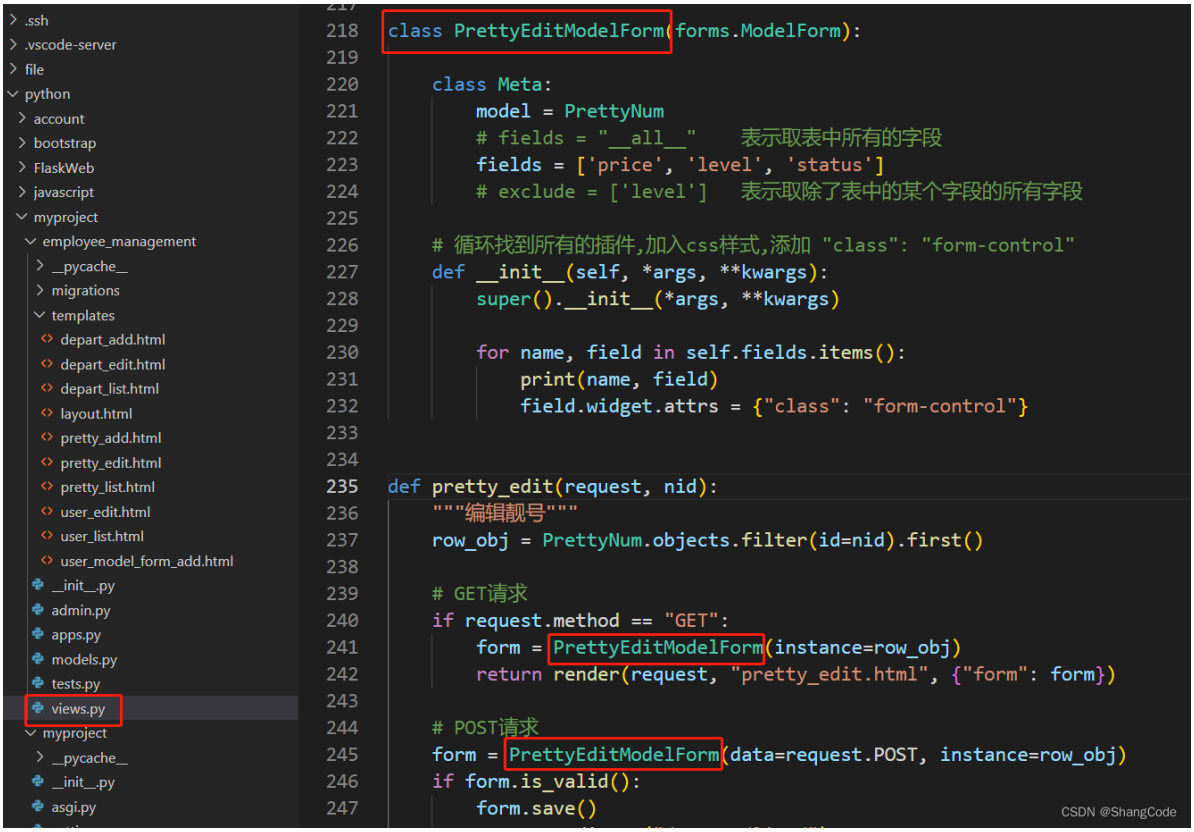
浏览器点击"编辑"更改**数据测试**即可

> 现在有一个新的需求,不想让用户编辑手机号字段,该怎么做?
> 其实我们可以单独新增一个class,去掉 `mobile` 字段

修改 `myproject/employee_management/views.py`

```
class PrettyEditModelForm(forms.ModelForm):

    class Meta:
        model = PrettyNum
```

```python
        # fields = "__all__"    表示取表中所有的字段
        fields = ['price', 'level', 'status']
        # exclude = ['level']   表示取除了表中的某个字段的所有字段

    # 循环找到所有的插件,加入css样式,添加 "class": "form-control"
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        for name, field in self.fields.items():
            print(name, field)
            field.widget.attrs = {"class": "form-control"}


def pretty_edit(request, nid):
    """编辑靓号"""
    row_obj = PrettyNum.objects.filter(id=nid).first()

    # GET请求
    if request.method == "GET":
        form = PrettyEditModelForm(instance=row_obj)
        return render(request, "pretty_edit.html", {"form": form})

    # POST请求
    form = PrettyEditModelForm(data=request.POST, instance=row_obj)
    if form.is_valid():
        form.save()
        return redirect("/pretty/list/")

    return render(request, "pretty_edit.html", {"form": form})
```



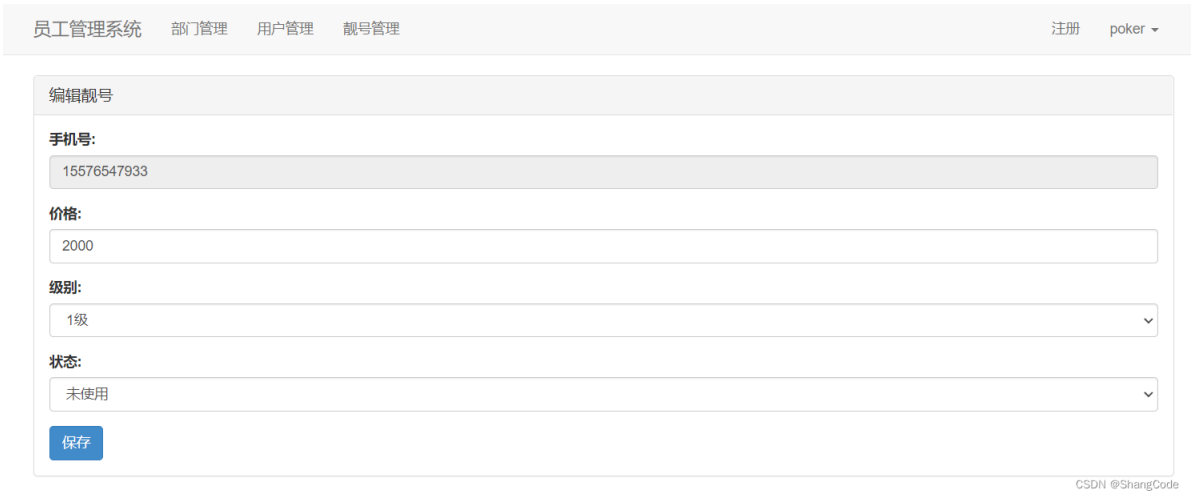
浏览器访问,可以发现没有手机号了

> 这时可能有人会说,我就是想要加上手机号字段,但是不想让他可编辑
> 这个当然也可以实现哈

修改 `myproject/employee_management/views.py`

```
1    mobile = forms.CharField(disabled=True, label="手机号")
```

浏览器刷新



> 那如果也允许编辑手机号字段呢
> 那么这样会有一个问题,如果手机号已经存在怎么办?
> 这里就涉及到了手机号的重复性问题

不允许手机号重复

- 添加: 如果手机号已存在,提示"手机号已存在"
- 编辑: 如果手机号除了当前手机号以外已存在,提示"手机号已存在"

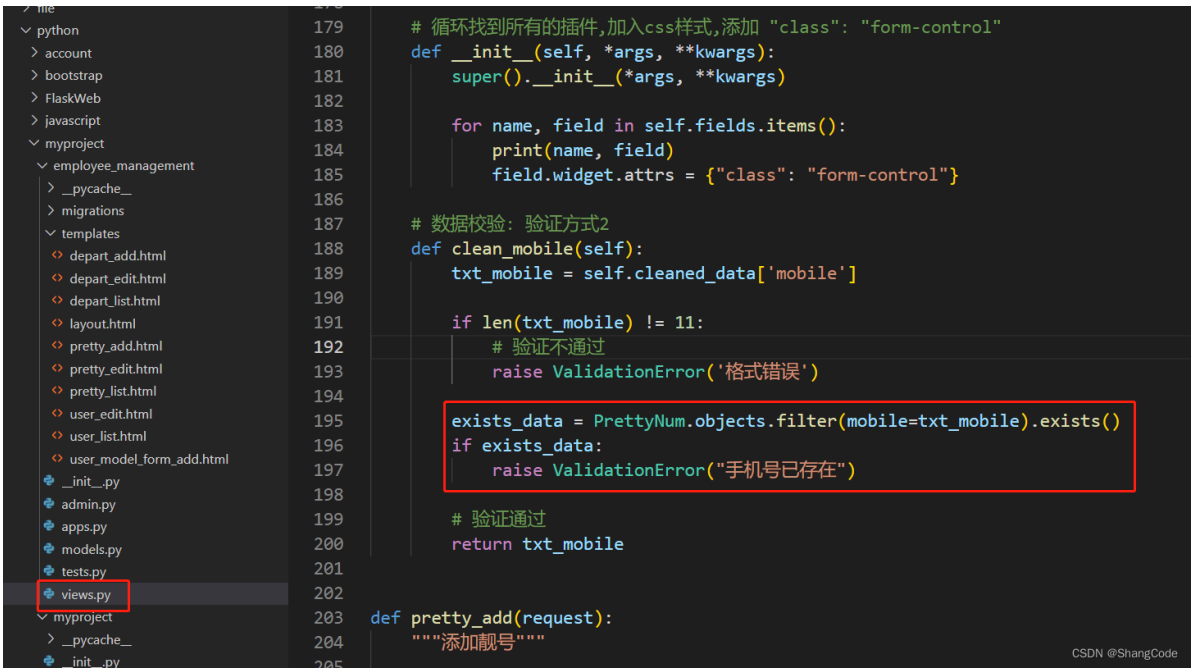
> 添加和编辑的逻辑不太一样,编辑需要将它自己先排除然后再做判断

首先实现"添加"功能的手机号重复验证

修改 `myproject/employee_management/views.py` 中 `class PrettyModelForm` 的 `clean_mobile` 函数

```
# 数据校验：验证方式1
    mobile = forms.CharFIeld(
        label="手机号",
        validators=[RegexValidator(r'^1[3-9]\d{9}$',"手机号格式错误"),],
    )
# 数据校验：验证方式2
def clean_mobile(self):
    txt_mobile = self.cleaned_data['mobile']

    if len(txt_mobile) != 11:
        # 验证不通过
        raise ValidationError('格式错误')
    exists_data = PrettyNum.objects.filter(mobile=txt_mobile).exists()
```

```
        if exists_data:
            raise ValidationError("手机号已存在")

        # 验证通过
        return txt_mobile
```



浏览器访问测试

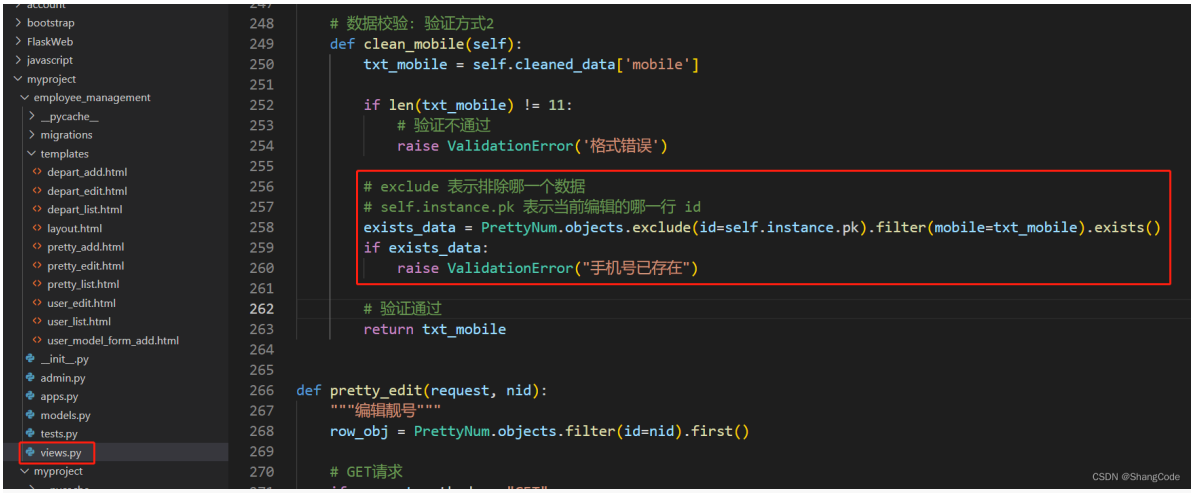
实现"编辑"功能的手机号重复验证

> 注意: 下面修改的是 `PrettyEditModelForm` 类下面的函数

修改 `myproject/employee_management/views.py` 中 `class PrettyEditModelForm` 的 `clean_mobile` 函数

```
    # 数据校验：验证方式2
    def clean_mobile(self):
        txt_mobile = self.cleaned_data['mobile']

        if len(txt_mobile) != 11:
            # 验证不通过
            raise ValidationError('格式错误')

        # exclude 表示排除哪一个数据
        # self.instance.pk 表示当前编辑的哪一行 id
```

```
        exists_data = PrettyNum.objects.exclude(id=self.instance.pk).filter(mobile=txt_mobile).exists()
        if exists_data:
            raise ValidationError("手机号已存在")

        # 验证通过
        return txt_mobile
```



不更改直接保存,正常

将手机号更改为已经存在的其他电话号码,保存,则提示"手机号已存在"

靓号删除

修改 `myproject/employee_management/views.py`

```python
def pretty_delete(request, nid):
    """删除靓号"""
    PrettyNum.objects.filter(id=nid).delete()
    return redirect('/pretty/list/')
```

修改 `myproject/myproject/urls.py`

```python
path('pretty/<int:nid>/delete/', views.pretty_delete),
```

修改 `/root/python/myproject/employee_management/templates/pretty_list.html`

浏览器测试即可

手机号搜索

```python
query1 = PrettyNum.objects.filter(mobile=15576547933, id=4)
print(query1)

# 如果是空字典,表示获取所有
query_dict = {"mobile": "15576547933", "id": 4}
query2 = PrettyNum.objects.filter(**query_dict)
print(query2)
PrettyNum.objects.filter(id=4)          # 等于4
PrettyNum.objects.filter(id__gt=4)      # 大于4
PrettyNum.objects.filter(id__gte=4)     # 大于等于4
PrettyNum.objects.filter(id__lt=4)      # 小于4
PrettyNum.objects.filter(id__lte=4)     # 小于等于4

PrettyNum.objects.filter(mobile__startswith="1999")      # 筛选出以"1999"开头的
PrettyNum.objects.filter(mobile__endswith="1999")        # 筛选出以"1999"结尾的
PrettyNum.objects.filter(mobile__contains="1999")        # 筛选出包含"1999"开头的
```

> 接下来我们来实现这个功能

修改 `myproject/employee_management/views.py`

```python
def pretty_list(request):
```

```
    """靓号列表"""

    data_dict = {}

    search_data = request.GET.get('query', "")  # 不加后面的 "", 首次访问浏览器,搜索框中不会显示前端页面中的 placeholder="Search for..." 属性
    if search_data:
        # mobile_contains  表示筛选出字段 mobile 包含 search_data 数据的行
        data_dict["mobile__contains"] = search_data

    # select * from 表 by level desc;
    # data_dict 如果是空字典,表示获取所有
    pretty_data = PrettyNum.objects.filter(**data_dict).order_by("-level")

    # 加入search_data的目的是,当搜索后,搜索框内的值不会置为空
    return render(request, "pretty_list.html", {"pretty_data": pretty_data, "search_data": search_data})
```

修改`myproject/employee_management/templates/pretty_list.html`

```
{% extends 'layout.html' %}

{% block content %}
<div class="container">

    <div>
        <div style="margin-bottom: 10px; ">
            <a class="btn btn-primary" href="/pretty/add/" target="_blank">新建靓号</a>

            <div style="float: right; width: 300px;">
                <form method="get">
                    <div class="input-group">
                        <input type="text" name="query" class="form-control" placeholder="Search for..."
                               value="{{ search_data }}">
                        <span class="input-group-btn">
                            <button class="btn btn-default" type="submit">
                                <span class="glyphicon glyphicon-search" aria-hidden="true"></span>
                            </button>
                        </span>
                    </div>
                </form>
            </div>
        </div>
    </div>

    <div>
        <div class="panel panel-default">
            <!-- Default panel contents -->
            <div class="panel-heading">
                <span class="glyphicon glyphicon-th-list" aria-hidden="true" style="margin-right: 5px;"></span>
                <span>靓号列表</span>
            </div>
```

```
33
34              <!-- Table -->
35              <table class="table table-bordered">
36                  <thead>
37                      <tr>
38                          <th>ID</th>
39                          <th>号码</th>
40                          <th>价格</th>
41                          <th>级别</th>
42                          <th>状态</th>
43                          <th>操作</th>
44                      </tr>
45                  </thead>
46                  <tbody>
47                      {% for obj in pretty_data %}
48                      <tr>
49                          <th>{{ obj.id }}</th>
50                          <td>{{ obj.mobile }}</td>
51                          <td>{{ obj.price }}</td>
52                          <td>{{ obj.get_level_display }}</td>
53                          <td>{{ obj.get_status_display }}</td>
54                          <td>
55                              <a class="btn btn-primary btn-xs" href="/pretty/{{ obj.id }}/edit/">编辑</a>
56                              <a class="btn btn-danger btn-xs" href="/pretty/{{ obj.id }}/delete/">删除</a>
57                          </td>
58                      </tr>
59                      {% endfor %}
60                  </tbody>
61              </table>
62          </div>
63      </div>
64  </div>
65
66  {% endblock %}
```

浏览器访问测试

进行搜索

分页

```
quertset = PrettyNum.objects.filter(id=1)[0:10]
quertset = PrettyNum.objects.all()[0:10]            # 取0到10行数据
```

> 先使用最原始的代码方式实现分页
> 内容较多,仔细看,有很多的细节

修改`myproject/employee_management/views.py`

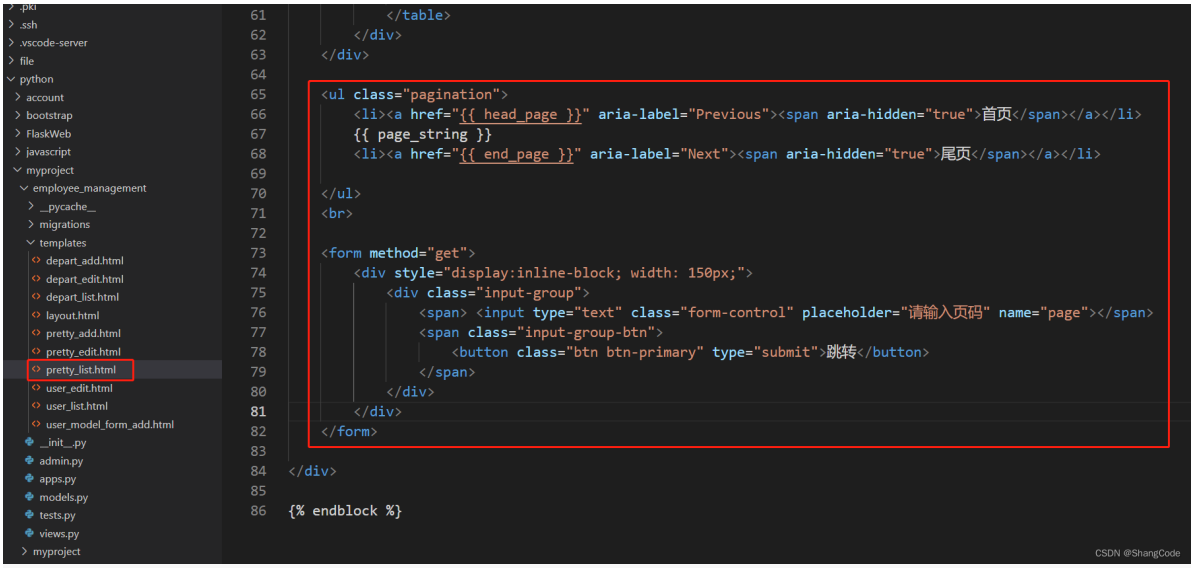
```
from django.utils.safestring import mark_safe

def pretty_list(request):
    """靓号列表"""

    ################### 纯代码实现分页 ###################
    # 插入一些演示数据,使用完注释掉
    # for i in range(300):
    #     PrettyNum.objects.create(mobile="13811223344", price=100, level=3, status=1)

    page_now = int(request.GET.get('page', 1))

    page_range = 5  # 当前页面前后留几个页码
    start = (page_now - 1) * 10
    end = page_now * 10 + 2

    # 总数据行数
    data_num = PrettyNum.objects.all().count()

    # 总页码
    page_num, div = divmod(data_num, 10)
```

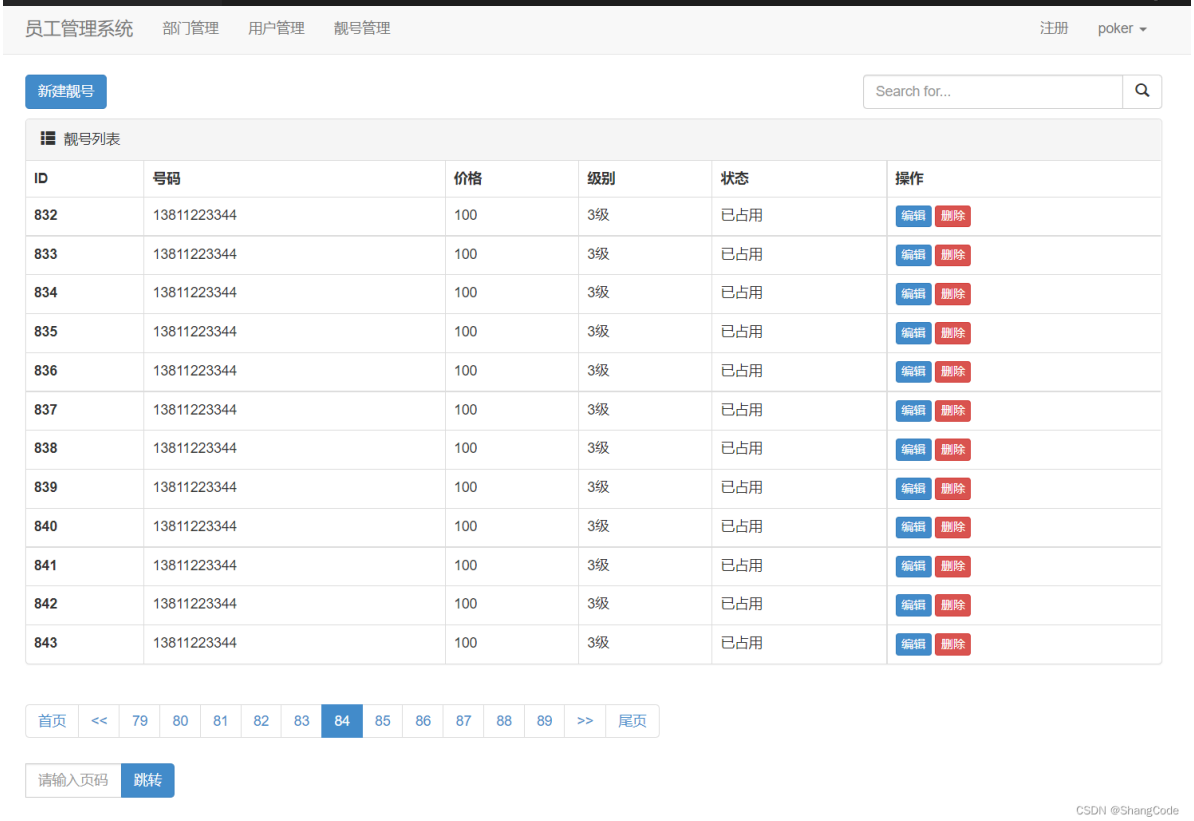
```
    if div:
        page_num += 1

    page_show = 5
    # 如果总页码数量大于 11
    if page_num > page_show * 2 + 1:
        # 如果当前页面页码位置小于等于5
        if page_now <= 5:
            start_page = 1
            end_page = page_show * 2 + 2
        # 否则,当前页面页码位置大于5时
        else:
            # 防止页码超出范围
            if page_now >= page_num - page_show:
                start_page = page_num - page_show * 2
                end_page = page_num + 1
            else:
                # 计算出当前页的前5页和后5页
                start_page = page_now - page_show
                end_page = page_now + page_show + 1

    else:
        start_page = 1
        end_page = page_num + 1


    ######## 创建页码 ########
    # 页码
    page_str_list = []

    # 跳到首页
    head_page = '?page={}'.format(1)

    # 跳到上10页
    # 如果当前页面小于 11, 防止超过最小页数
    if page_now < page_show * 2 + 1:
        prev = '<li><a href="?page={}">{}</a></li>'.format(1, "<<")
        page_str_list.append(prev)
    else:
        prev = '<li><a href="?page={}">{}</a></li>'.format(page_now - 10, "<<")
        page_str_list.append(prev)

    for i in range(start_page, end_page):

        # 如果是当前页,高亮显示页码颜色
        if page_now == i:
            ele = '<li class="active"><a href="?page={}">{}</a></li>'.format(i,
i)
        else:
            ele = '<li><a href="?page={}">{}</a></li>'.format(i, i)
        page_str_list.append(ele)

    ### 跳到下10页
    # 如果当前页面页数 大于 最大页面数量减去(page_show*2+1),则直接跳到最后一页,防止超过最大
页数
    if page_now >= page_num - page_show * 2 + 1:
```

```
            next = '<li><a href="?page={}">{}</a></li>'.format(page_num, ">>")
            page_str_list.append(next)
        else:
            next = '<li><a href="?page={}">{}</a></li>'.format(page_now + 10, ">>")
            page_str_list.append(next)

        page_string = mark_safe("".join(page_str_list))

        # 跳到尾页
        end_page = '?page={}'.format(page_num)

        ################### 代码实现分页结束 ###################

        data_dict = {}
        # 如果是空字典,表示获取所有
        search_data = request.GET.get('query', "")  # 不加后面的 "", 首次访问浏览器,搜索框中不会显示前端页面中的 placeholder="Search for..." 属性
        if search_data:
            data_dict["mobile__contains"] = search_data

        # select * from 表 by level desc;
        pretty_data = PrettyNum.objects.filter(**data_dict).order_by("-level")[start:end]

        return render(request, "pretty_list.html", {"pretty_data": pretty_data, "search_data": search_data, "page_string": page_string, "head_page": head_page, "end_page": end_page})

```

修改 `myproject/employee_management/templates/pretty_list.html`

```
<ul class="pagination">
    <li><a href="{{ head_page }}" aria-label="Previous"><span aria-hidden="true">首页</span></a></li>
    {{ page_string }}
    <li><a href="{{ end_page }}" aria-label="Next"><span aria-hidden="true">尾页</span></a></li>

</ul>
<br>

<form method="get">
    <div style="display:inline-block; width: 150px;">
        <div class="input-group">
            <span> <input type="text" class="form-control" placeholder="请输入页码" name="page"></span>
            <span class="input-group-btn">
                <button class="btn btn-primary" type="submit">跳转</button>
            </span>
        </div>
    </div>
</form>
```





封装分页

> 实现分页代码的封装,方便以后其他代码的调用

在 `myproject/employee_management` 目录下新建 `utils` 目录
在 `utils` 下新建 `pagination.py` 文件

```
"""
自定义的分页组件
"""
from django.utils.safestring import mark_safe

class Pagination(object):
    def __init__(self, request, queryset, page_size=10, page_param="page",
page_show=5):
```

```
        """
        :param request: 请求的对象
        :param queryset: 符合条件的数据(根据此数据进行分页处理)
        :param page_size: 每页显示多少条数据
        :param page_param: 获取在URL中传递的分页参数, 例如: /pretty/list/?page=21
        :param page_show: 页码显示前几页后几页
        """
        page = int(request.GET.get(page_param, 1))

        # 如果不是整数
        if type(page) != int:
            # 强制让页码为1
            page = 1

        self.page = page

        self.start = (page - 1) * page_size
        self.end = page * page_size

        # 每页展示的数据行数
        self.page_queryset = queryset[self.start:self.end]

        total_data_count = queryset.count()     # 数据行数
        total_page_count, div = divmod(total_data_count, page_size)
        if div:
            total_page_count += 1
        self.total_page_count = total_page_count    # 总页码数量
        self.page_show = page_show  # 当前页前后展示的页码数量
        self.request = request


    def html(self):
        # 如果总页码数量大于 11
        if self.total_page_count > self.page_show * 2 + 1:
            # 如果当前页面页码位置小于等于5
            if self.page <= 5:
                start_page = 1
                end_page = self.page_show * 2 + 2
            # 否则,当前页面页码位置大于5时
            else:
                # 防止页码超出范围
                if self.page >= self.total_page_count - self.page_show:
                    start_page = self.total_page_count - self.page_show * 2
                    end_page = self.total_page_count + 1
                else:
                    # 计算出当前页的前5页和后5页
                    start_page = self.page - self.page_show
                    end_page = self.page + self.page_show + 1

        else:
            start_page = 1
            end_page = self.total_page_count + 1

        ######## 创建页码 ########
        # 页码
        page_str_list = []
```

```
64
65          # 跳到首页
66          self.head_page = '?page={}'.format(1)
67
68          # 跳到上10页
69          # 如果当前页面小于 11, 防止超过最小页数
70          if self.page < self.page_show * 2 + 1:
71              prev = '<li><a href="?page={}">{}</a></li>'.format(1, "<<")
72              page_str_list.append(prev)
73          else:
74              prev = '<li><a href="?page={}">{}</a></li>'.format(self.page - 10, "
    <<")
75              page_str_list.append(prev)
76
77          for i in range(start_page, end_page):
78
79              # 如果是当前页,高亮显示页码颜色
80              if self.page == i:
81                  ele = '<li class="active"><a href="?page={}">{}</a>
    </li>'.format(i, i)
82              else:
83                  ele = '<li><a href="?page={}">{}</a></li>'.format(i, i)
84              page_str_list.append(ele)
85
86          # 跳到下10页
87          # 如果当前页面页数 大于 最大页面数量减去(page_show*2+1),则直接跳到最后一页,防止超过
    最大页数
88          if self.page >= self.total_page_count - self.page_show * 2 + 1:
89              next = '<li><a href="?page={}">{}</a>
    </li>'.format(self.total_page_count, ">>")
90              page_str_list.append(next)
91          else:
92              next = '<li><a href="?page={}">{}</a></li>'.format(self.page + 10,
    ">>")
93              page_str_list.append(next)
94
95          self.page_string = mark_safe("".join(page_str_list))
96
97          # 跳到尾页
98          self.end_page = '?page={}'.format(self.total_page_count)
```

修改 `myproject/employee_management/views.py`

```
1   def pretty_list(request):
2       """靓号列表"""
3
4       data_dict = {}
5       # 如果是空字典,表示获取所有
6       # 不加后面的 "", 首次访问浏览器,搜索框中不会显示前端页面中的 placeholder="Search
    for..." 属性
7       search_data = request.GET.get('query', "")
8       if search_data:
9           data_dict["mobile__contains"] = search_data
10
11      queryset = PrettyNum.objects.filter(**data_dict).order_by("-level")
12
```

```
    ### 引入封装的 Pagination 类并初始化
    # 初始化
    page_object = Pagination(request, queryset, page_size=10, page_param="page")
    page_queryset = page_object.page_queryset

    # 调用对象的html方法,生成页码
    page_object.html()

    page_string = page_object.page_string
    head_page = page_object.head_page
    end_page = page_object.end_page

    context = {
        "pretty_data": page_queryset,    # 分页的数据
        "search_data": search_data,      # 搜索的内容
        "page_string": page_string,      # 页码
        "head_page": head_page,          # 首页
        "end_page": end_page,            # 尾页
    }

    return render(request, "pretty_list.html", context)
```

浏览器刷新访问,你会发现效果和之前一样

解决小BUG

> 当我们进行搜索后,可以展示出搜索的对应数据
> 但是如果此时我们进行翻页,搜索数据将清空,重新展示所有的数据
> 这样是不符合逻辑的

下面的代码对之前的代码进行了优化:

- 将首页和尾页加入到了 `page_str_list` 列表中,方便后续其他页面进行调用

修改 `myproject/employee_management/utils/pagination.py`

```
"""
自定义的分页组件

"""

from django.utils.safestring import mark_safe
import copy


class Pagination(object):
    def __init__(self, request, queryset, page_size=10, page_param="page", page_show=5):
        """
        :param request: 请求的对象
        :param queryset: 符合条件的数据(根据此数据进行分页处理)
        :param page_size: 每页显示多少条数据
        :param page_param: 获取在URL中传递的分页参数, 例如: /pretty/list/?page=21
        :param page_show: 页码显示前几页后几页
```

```
        """

        # 防止搜索出结果进行翻页时,URL参数没有了搜索参数
        query_dict = copy.deepcopy(request.GET)
        query_dict._mutable = True
        self.query_dict = query_dict

        self.page_param = page_param

        page = int(request.GET.get(page_param, 1))

        # 如果不是整数
        if type(page) != int:
            # 强制让页码为1
            page = 1

        self.page = page

        self.start = (page - 1) * page_size
        self.end = page * page_size

        # 每页展示的数据行数
        self.page_queryset = queryset[self.start:self.end]

        total_data_count = queryset.count()     # 数据行数
        total_page_count, div = divmod(total_data_count, page_size)
        if div:
            total_page_count += 1
        self.total_page_count = total_page_count    # 总页码数量
        self.page_show = page_show  # 当前页前后展示的页码数量
        self.request = request

    def html(self):
        # 如果总页码数量大于 11
        if self.total_page_count > self.page_show * 2 + 1:
            # 如果当前页面页码位置小于等于5
            if self.page <= 5:
                start_page = 1
                end_page = self.page_show * 2 + 2
            # 否则,当前页面页码位置大于5时
            else:
                # 防止页码超出范围
                if self.page >= self.total_page_count - self.page_show:
                    start_page = self.total_page_count - self.page_show * 2
                    end_page = self.total_page_count + 1
                else:
                    # 计算出当前页的前5页和后5页
                    start_page = self.page - self.page_show
                    end_page = self.page + self.page_show + 1

        else:
            start_page = 1
            end_page = self.total_page_count + 1

        ######## 创建页码 ########
        # 页码
```

```
        page_str_list = []

        # self.query_dict.setlist(self.page_param, [1])
        # page_str_list.append('<li><a href="?page={}">{}</a></li>'.format(self.query_dict.urlencode()))

        # 跳到首页
        self.query_dict.setlist(self.page_param, [1])
        self.head_page = '<li><a href="?{}" aria-label="Previous"><span aria-hidden="true">首页</span></a></li>'.format(self.query_dict.urlencode())
        page_str_list.append(self.head_page)

        # 跳到上10页
        # 如果当前页面小于 11, 防止超过最小页数
        if self.page < self.page_show * 2 + 1:
            self.query_dict.setlist(self.page_param, [1])
            prev = '<li><a href="?{}">{}</a></li>'.format(
                self.query_dict.urlencode(), "<<")
            page_str_list.append(prev)
        else:
            self.query_dict.setlist(self.page_param, [self.page - 10])
            prev = '<li><a href="?{}">{}</a></li>'.format(
                self.query_dict.urlencode(), "<<")
            page_str_list.append(prev)

        for i in range(start_page, end_page):
            # 如果是当前页,高亮显示页码颜色
            if self.page == i:
                self.query_dict.setlist(self.page_param, [i])
                ele = '<li class="active"><a href="?{}">{}</a></li>'.format(
                    self.query_dict.urlencode(), i)
            else:
                self.query_dict.setlist(self.page_param, [i])
                ele = '<li><a href="?{}">{}</a></li>'.format(
                    self.query_dict.urlencode(), i)
            page_str_list.append(ele)

        # 跳到下10页
        # 如果当前页面页数 大于 最大页面数量减去(page_show*2+1),则直接跳到最后一页,防止超过最大页数
        if self.page >= self.total_page_count - self.page_show * 2 + 1:
            self.query_dict.setlist(self.page_param, [self.total_page_count])
            next = '<li><a href="?{}">{}</a></li>'.format(
                self.query_dict.urlencode(), ">>")
            page_str_list.append(next)
        else:
            self.query_dict.setlist(self.page_param, [self.page + 10])
            next = '<li><a href="?page={}">{}</a></li>'.format(
                self.query_dict.urlencode(), ">>")
            page_str_list.append(next)

        # 跳到尾页
        self.query_dict.setlist(self.page_param, [self.total_page_count])
        self.end_page = '<li><a href="?{}" aria-label="Next"><span aria-hidden="true">尾页</span></a></li>'.format(self.query_dict.urlencode())
        page_str_list.append(self.end_page)
```

```
126
127             self.page_string = mark_safe("".join(page_str_list))
```

对上面一段代码的解析:

```
query_dict = copy.deepcopy(request.GET)
query_dict._mutable = True
self.query_dict = query_dict
```

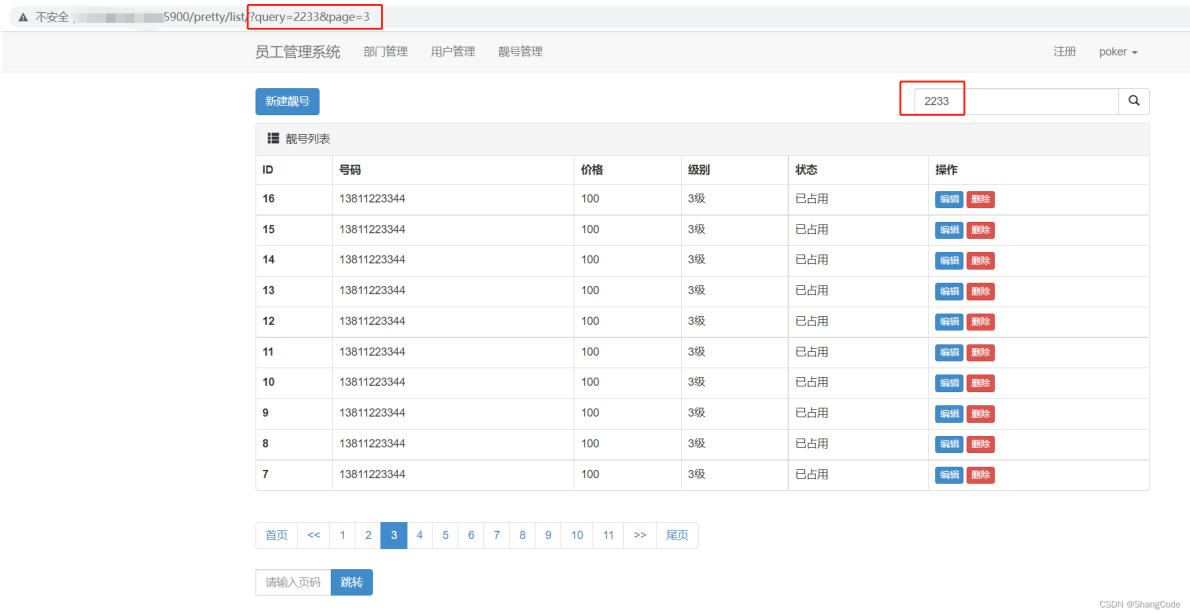
上面这一段代码是获取保留目前已有的URL参数
`query_dict._mutable = True` 表示将URL改为参数可拼接

```
self.query_dict.setlist(self.page_param, [self.page + 10])
    next = '<li><a href="?page={}">{}</a>
</li>'.format(self.query_dict.urlencode(), ">>")
```

上面这一段代码表示在原来URL参数的基础上进行拼接,不是像原来一样直接覆盖URL参数
`setlist` 表示增加URL参数
`self.query_dict.urlencode()` 表示将 `self.page_param` 与数值 `self.page + 10` 拼接起来,不会讲原来已有的URL参数覆盖
例如 `setlist(page, [10])` 表示 `page=10`
修改 `myproject/employee_management/templates/pretty_list.html`

```
<ul class="pagination">
    {{ page_string }}
</ul>
```

> 浏览器访问测试



但是目前 输入页码跳转到指定页 的功能还是有这个问题,现在还无法解决,需要使用到ajax

用户列表增加分页

修改 `myproject/employee_management/views.py`

```
def user_list(request):
    """用户列表"""
    # 获取所有用户列表
    queryset = UserInfo.objects.all()

    page_object = Pagination(request, queryset, page_size=2)

    # 用 python 的语法获取数据
    """
    for obj in user_data:
        # obj.get_gender_display() 表示匹配 男/女,原始字段名为gender,obj.get_字段名称_display()
        # obj.create_time.strftime("%Y-%m-%d") 表示将时间格式转换成固定格式的字符串
        # obj.depart.title 表示获取depart_id对应的部门名称,因为我们在models中定义表时与另外一张表设置了级联关系,有外键
        print(obj.id, obj.name, obj.password, obj.age, obj.account, obj.get_gender_display(), obj.depart.title, obj.create_time.strftime("%Y-%m-%d"))
    """

    page_object.html()

    context = {
        "queryset": page_object.page_queryset,
        "page_string": page_object.page_string,
    }

    return render(request, "user_list.html", context)
```

修改 `myproject/employee_management/templates/user_list.html`

```
{% for obj in queryset %}

...

<ul class="pagination">
    {{ page_string }}
</ul>
```

浏览器访问测试

部门列表增加分页

修改 `myproject/employee_management/views.py`

```
def depart_list(request):
    """部门列表"""

    queryset = Department.objects.all()
    page_object = Pagination(request, queryset, page_size=2)

    page_object.html()

    context = {
        "queryset": page_object.page_queryset,
        "page_string": page_object.page_string,
    }

    return render(request, "depart_list.html", context)
```

修改 `myproject/employee_management/templates/depart_list.html`

```
{% for obj in queryset %}

...

<ul class="pagination">
    {{ page_string }}
</ul>
```

浏览器访问

项目代码优化

ModelForm优化

新建`myproject/employee_management/utils/modelform.py`

```
from django import forms

class BootStrapModelForm(forms.ModelForm):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        # 循环ModelForm中的所有字段,给每个字段的插件设置
        for _, field in self.fields.items():
            # 字段中有属性,保留原来的属性,没有属性,才增加
            if field.widget.attrs:
                field.widget.attrs["class"] = "form-control"
            else:
                field.widget.attrs = {
                    "class": "form-control",
                }
```

修改`myproject/employee_management/views.py`

> 将目前的三个class类改为继承 `BootStrapModelForm` 自定义类

```
class UserModelForm(BootStrapModelForm):
class PrettyModelForm(BootStrapModelForm):
class PrettyEditModelForm(BootStrapModelForm):
```

> 删除三个class的 `init函数`

刷新浏览器查看效果,你会发现和以前一样,没有变化

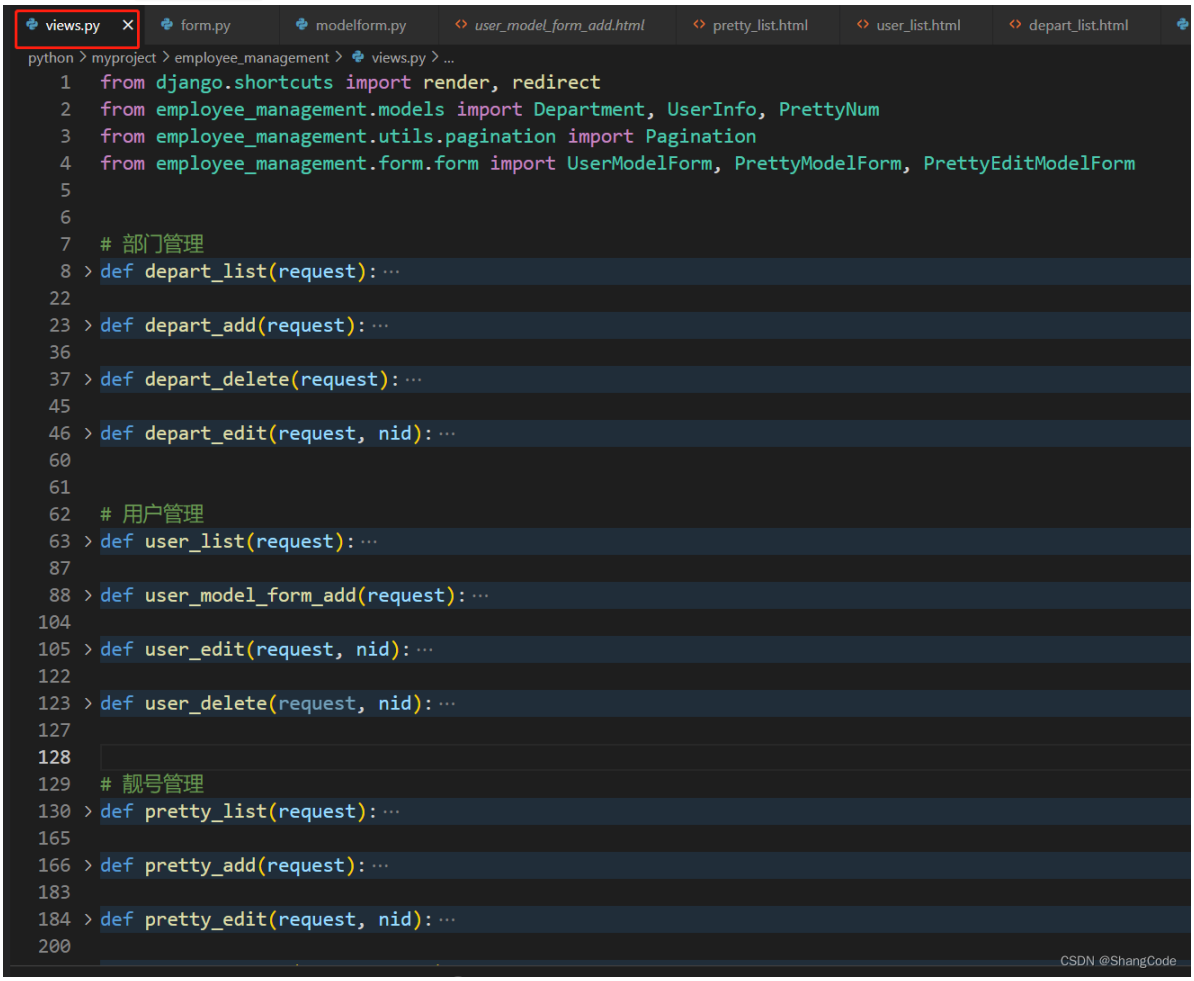
整合form

新建 `myproject/employee_management/form` 目录
新建 `myproject/employee_management/form/form.py` 文件,
将 `/root/python/myproject/employee_management/views.py` 文件中的相关class移动到此文件中

```
from employee_management.utils.modelform import BootStrapModelForm
from employee_management.models import UserInfo, PrettyNum
from django.core.exceptions import ValidationError
from django import forms

class UserModelForm(BootStrapModelForm):

    ### 自定义数据校验
    # 例如: 用户名最小三个字符
    #name = forms.CharField(min_length=3, label="用户名")

    class Meta:
        model = UserInfo
        fields = ["name", "password", "age", "account", "create_time", "gender", "depart"]
        # 逐一控制标签的样式
        # widgets = {
        #     "name": forms.TextInput(attrs={"class": "form-control"}),
        #     "password": forms.PasswordInput(attrs={"class": "form-control"}),
        # }

        # 这里让日期可以手动点击鼠标选择,所以单独拎出来,加上日期插件
        widgets = {
            # "create_time": forms.DateInput(attrs={'class': 'form-control', 'id': 'myDate'}),
            "create_time": forms.DateInput(attrs={'id': 'myDate'}),
        }


class PrettyModelForm(BootStrapModelForm):

    # 数据校验: 验证方式1
    # mobile = forms.CharField(
    #     label="手机号",
    #     validators=[RegexValidator(r'^1[3-9]\d{9}$', '手机号格式错误'),],
    # )
```

```python

    class Meta:
        model = PrettyNum
        # fields = "__all__"    表示取表中所有的字段
        fields = ['mobile', 'price', 'level', 'status']
        # exclude = ['level']   表示取除了表中的某个字段的所有字段

    # 数据校验: 验证方式2
    def clean_mobile(self):
        txt_mobile = self.cleaned_data['mobile']

        if len(txt_mobile) != 11:
            # 验证不通过
            raise ValidationError('格式错误')

        exists_data = PrettyNum.objects.filter(mobile=txt_mobile).exists()
        if exists_data:
            raise ValidationError("手机号已存在")

        # 验证通过
        return txt_mobile


class PrettyEditModelForm(BootStrapModelForm):

    # mobile = forms.CharField(disabled=True, label="手机号")

    # 数据校验: 验证方式1
    # mobile = forms.CharField(
    #     label="手机号",
    #     validators=[RegexValidator(r'^1[3-9]\d{9}$', '手机号格式错误'),],
    # )

    class Meta:
        model = PrettyNum
        # fields = "__all__"    表示取表中所有的字段
        fields = ['mobile', 'price', 'level', 'status']
        # exclude = ['level']   表示取除了表中的某个字段的所有字段

    # 数据校验: 验证方式2
    def clean_mobile(self):
        txt_mobile = self.cleaned_data['mobile']

        if len(txt_mobile) != 11:
            # 验证不通过
            raise ValidationError('格式错误')

        # exclude 表示排除哪一个数据
        # self.instance.pk 表示当前编辑的哪一行 id
        exists_data = PrettyNum.objects.exclude(id=self.instance.pk).filter(mobile=txt_mobile).exists()
        if exists_data:
            raise ValidationError("手机号已存在")

        # 验证通过
        return txt_mobile
```

整理之后的 `views.py` 文件是这样的



优化views.py

> 我们也可以将 `/root/python/myproject/employee_management/views.py` 进行拆分
> 按照部门进行拆分

新建 `/root/python/myproject/employee_management/views` 目录
然后在其下依次新建 `user.py` `depart.py` `pretty.py`

> depart.py

```python
from django.shortcuts import render, redirect
from employee_management.models import Department
from employee_management.utils.pagination import Pagination


def depart_list(request):
    """部门列表"""

    queryset = Department.objects.all()
    page_object = Pagination(request, queryset, page_size=2)

    page_object.html()

    context = {
```

```
        "queryset": page_object.page_queryset,
        "page_string": page_object.page_string,
    }

    return render(request, "depart_list.html", context)

def depart_add(request):
    """部门添加"""
    if request.method == "GET":
        return render(request, "depart_add.html")

    # 获取用户提交的部门数据
    depart_title = request.POST.get("depart_title")

    # 保存到数据库
    Department.objects.create(title=depart_title)

    # 重定向回部门列表
    return redirect("/depart/list/")

def depart_delete(request):
    """部门删除"""

    nid = request.GET.get('nid')
    Department.objects.filter(id=nid).delete()

    # 重定向回部门列表
    return redirect("/depart/list/")

def depart_edit(request, nid):
    """部门编辑"""

    if request.method == "GET":
        # 根据nid,获取数据
        row_object = Department.objects.filter(id=nid).first()
        return render(request, 'depart_edit.html', {"row_object": row_object})

    # 如果是POST请求,保存修改
    depart_title = request.POST.get('depart_title')
    Department.objects.filter(id=nid).update(title=depart_title)

    # 重定向回部门列表
    return redirect('/depart/list/')
```

user.py

```
from django.shortcuts import render, redirect
from employee_management.models import UserInfo
from employee_management.utils.pagination import Pagination
from employee_management.form.form import UserModelForm


def user_list(request):
    """用户列表"""
    # 获取所有用户列表
```

```
    queryset = UserInfo.objects.all()

    page_object = Pagination(request, queryset, page_size=2)

    # 用 python 的语法获取数据
    """
    for obj in user_data:
        # obj.get_gender_display() 表示匹配 男/女,原始字段名为gender,obj.get_字段名称_display()
        # obj.create_time.strftime("%Y-%m-%d") 表示将时间格式转换成固定格式的字符串
        # obj.depart.title 表示获取depart_id对应的部门名称,因为我们在models中定义表时与另外一张表设置了级联关系,有外键
        print(obj.id, obj.name, obj.password, obj.age, obj.account, obj.get_gender_display(), obj.depart.title, obj.create_time.strftime("%Y-%m-%d"))
    """

    page_object.html()

    context = {
        "queryset": page_object.page_queryset,
        "page_string": page_object.page_string,
    }

    return render(request, "user_list.html", context)

def user_model_form_add(request):
    """添加用户(ModelForm版本)"""
    if request.method == "GET":
        form = UserModelForm()
        return render(request, "user_model_form_add.html", {"form": form})

    # 用户POST请求提交数据,需要进行数据校验
    form = UserModelForm(data=request.POST)
    if form.is_valid():
        print(form.cleaned_data)
        # 直接保存至数据库
        form.save()
        return redirect("/user/list/")

    # 校验失败(在页面上显示错误信息)
    return render(request, "user_model_form_add.html", {"form": form})

def user_edit(request, nid):
    """编辑用户"""

    row_obj = UserInfo.objects.filter(id=nid).first()

    # GET请求
    if request.method == "GET":
        form = UserModelForm(instance=row_obj)
        return render(request, "user_edit.html", {"form": form})

    # POST请求
    form = UserModelForm(data=request.POST, instance=row_obj)
    if form.is_valid():
        form.save()
```

```
63          return redirect("/user/list/")
64
65      return render(request, "user_edit.html", {"form": form})
66
67  def user_delete(request, nid):
68      """用户删除"""
69      UserInfo.objects.filter(id=nid).delete()
70      return redirect("/user/list/")
```

pretty.py

```
1   from django.shortcuts import render, redirect
2   from employee_management.models import PrettyNum
3   from employee_management.utils.pagination import Pagination
4   from employee_management.form.form import PrettyModelForm, PrettyEditModelForm
5
6
7   def pretty_list(request):
8       """靓号列表"""
9
10
11      print(request.GET)
12      print(request.GET.urlencode())
13
14      data_dict = {}
15      # 如果是空字典,表示获取所有
16      # 不加后面的 "", 首次访问浏览器,搜索框中不会显示前端页面中的 placeholder="Search
    for..." 属性
17      search_data = request.GET.get('query', "")
18      if search_data:
19          data_dict["mobile__contains"] = search_data
20
21      queryset = PrettyNum.objects.filter(**data_dict).order_by("-level")
22
23      ### 引入封装的 Pagination 类并初始化
24      # 初始化
25      page_object = Pagination(request, queryset, page_size=10, page_param="page")
26      page_queryset = page_object.page_queryset
27
28      # 调用对象的html方法,生成页码
29      page_object.html()
30
31      page_string = page_object.page_string
32
33      context = {
34          "pretty_data": page_queryset,    # 分页的数据
35          "search_data": search_data,      # 搜索的内容
36          "page_string": page_string,      # 页码
37      }
38
39
40
41      return render(request, "pretty_list.html", context)
42
43  def pretty_add(request):
```

```
    """添加靓号"""

    if request.method == "GET":
        form = PrettyModelForm()
        return render(request, "pretty_add.html", {"form": form})

    # 用户POST请求提交数据,需要进行数据校验
    form = PrettyModelForm(data=request.POST)
    if form.is_valid():
        print(form.cleaned_data)
        # 直接保存至数据库
        form.save()
        return redirect("/pretty/list/")

    # 校验失败(在页面上显示错误信息)
    return render(request, "pretty_add.html", {"form": form})

def pretty_edit(request, nid):
    """编辑靓号"""
    row_obj = PrettyNum.objects.filter(id=nid).first()

    # GET请求
    if request.method == "GET":
        form = PrettyEditModelForm(instance=row_obj)
        return render(request, "pretty_edit.html", {"form": form})

    # POST请求
    form = PrettyEditModelForm(data=request.POST, instance=row_obj)
    if form.is_valid():
        form.save()
        return redirect("/pretty/list/")

    return render(request, "pretty_edit.html", {"form": form})

def pretty_delete(request, nid):
    """删除靓号"""
    PrettyNum.objects.filter(id=nid).delete()
    return redirect('/pretty/list/')
```

重命名原来的`views.py`文件为`views_old.py`,不然会跟目前的views目录冲突

修改`myproject/myproject/urls.py`

```
from django.contrib import admin
from django.urls import path
from employee_management.views import depart,user,pretty

urlpatterns = [
    path('admin/', admin.site.urls),
    path('depart/list/', depart.depart_list),
    path('depart/add/', depart.depart_add),
    path('depart/delete/', depart.depart_delete),
    path('depart/<int:nid>/edit/', depart.depart_edit),
    path('user/list/', user.user_list),
    path('user/model/form/add/', user.user_model_form_add),
    path('user/<int:nid>/edit/', user.user_edit),
    path('user/<int:nid>/delete/', user.user_delete),
```

```
    path('pretty/list/', pretty.pretty_list),
    path('pretty/add/', pretty.pretty_add),
    path('pretty/<int:nid>/edit/', pretty.pretty_edit),
    path('pretty/<int:nid>/delete/', pretty.pretty_delete),
]
```

刷新浏览器,效果还是一样,不同的是我们的代码更加规整

管理员管理

创建数据库表

```
class Admin(models.Model):
    """管理员"""
    username = models.CharField(verbose_name="用户名", max_length=32)
    password = models.CharField(verbose_name="密码", max_length=64)

```

执行命令

```
python3 manage.py makemigrations
python3 manage.py migrate
```

```
[root@hecs-33592 ~]# cd python/myproject/
[root@hecs-33592 myproject]# python3 manage.py makemigrations
Migrations for 'employee_management':
  employee_management/migrations/0004_admin.py
    - Create model Admin
[root@hecs-33592 myproject]# python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, employee_management, sessions
Running migrations:
  Applying employee_management.0004_admin... OK
[root@hecs-33592 myproject]#
```


像素数据库表中插入一条数据

```
mysql> use my_project;

mysql> insert into employee_management_admin(username, password) values("张三", "123456");
Query OK, 1 row affected (0.00 sec)

mysql> insert into employee_management_admin(username, password) values("李四", "456789");
Query OK, 1 row affected (0.00 sec)

mysql> insert into employee_management_admin(username, password) values("楚云飞", "qwe123");
Query OK, 1 row affected (0.01 sec)

mysql> insert into employee_management_admin(username, password) values("小日子", "123456");
Query OK, 1 row affected (0.00 sec)

```

```
mysql> insert into employee_management_admin(username, password) values("小瘪三",
"123456");
Query OK, 1 row affected (0.00 sec)

mysql> insert into employee_management_admin(username, password) values("大狗",
"123456");
Query OK, 1 row affected (0.00 sec)

mysql> insert into employee_management_admin(username, password) values("孙悟空",
"123456");
Query OK, 1 row affected (0.00 sec)

12345678910111213141516171819202122 23
```

增加管理员菜单

修改 `/root/python/myproject/employee_management/templates/layout.html`

```
<ul class="nav navbar-nav">
    <li><a href="/admin/list/">管理员账户</a></li>
    <li><a href="/depart/list/">部门管理</a></li>
    <li><a href="/user/list/">用户管理</a></li>
    <li><a href="/pretty/list/">靓号管理</a></li>
</ul>
123456
```

管理员列表

新建 `myproject/employee_management/templates/admin_list.html`

```
{% extends 'layout.html' %}

{% block content %}
<div class="container">

    <div>
        <div style="margin-bottom: 10px; ">
            <a class="btn btn-primary" href="/pretty/add/" target="_blank">新建管
理员</a>

            <div style="float: right; width: 300px;">
                <form method="get">
                    <div class="input-group">
```

```
                        <input type="text" name="query" class="form-control" placeholder="Search for..."
                               value="{{ search_data }}">
                        <span class="input-group-btn">
                            <button class="btn btn-default" type="submit">
                                <span class="glyphicon glyphicon-search" aria-hidden="true"></span>
                            </button>
                        </span>
                    </div>
                </form>
            </div>
        </div>
    </div>

    <div>
        <div class="panel panel-default">
            <!-- Default panel contents -->
            <div class="panel-heading">
                <span class="glyphicon glyphicon-th-list" aria-hidden="true" style="margin-right: 5px;"></span>
                <span>管理员列表</span>
            </div>

            <!-- Table -->
            <table class="table table-bordered">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>姓名</th>
                        <th>密码</th>
                        <th>操作</th>
                    </tr>
                </thead>
                <tbody>
                    {% for obj in page_queryset %}
                    <tr>
                        <th>{{ obj.id }}</th>
                        <td>{{ obj.username }}</td>
                        <td>{{ obj.password }}</td>
                        <td>
                            <a class="btn btn-primary btn-xs" href="/admin/{{ obj.id }}/edit/">编辑</a>
                            <a class="btn btn-danger btn-xs" href="/admin/{{ obj.id }}/delete/">删除</a>
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    </div>

    <ul class="pagination">
        {{ page_string }}
    </ul>
```

```
    <br>

    <form method="get">
        <div style="display:inline-block; width: 150px;">
            <div class="input-group">
                <span> <input type="text" class="form-control" placeholder="请输入页码" name="page"></span>
                <span class="input-group-btn">
                    <button class="btn btn-primary" type="submit">跳转</button>
                </span>
            </div>
        </div>
    </form>
</div>

{% endblock %}
123456789101112131415161718192021222324252627282930313233343536373839404142434445464748495051525354555657585960616263646566676869707172737475767778
```

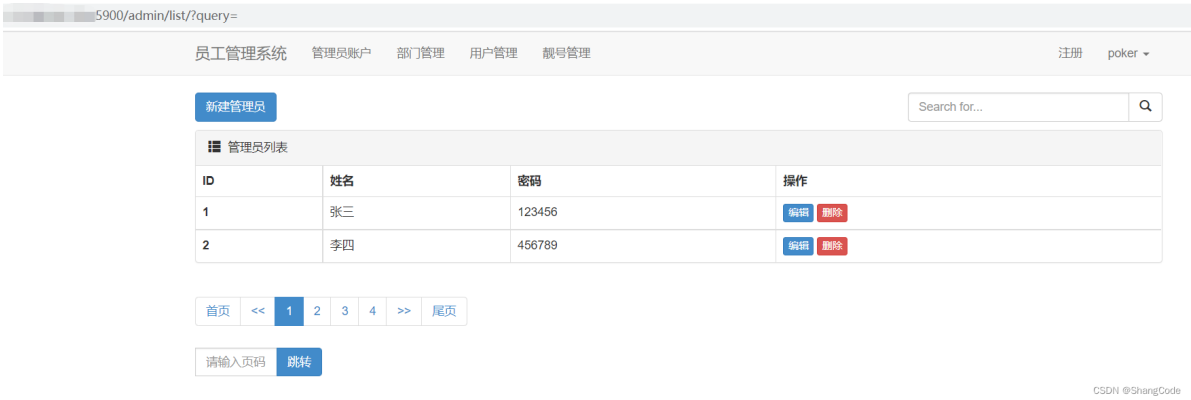
新建 `myproject/employee_management/views/admin.py`

```
from django.shortcuts import render
from employee_management.models import Admin
from employee_management.utils.pagination import Pagination


def admin_list(request):
    """管理员列表"""

    data_dict = {}
    # 不加后面的 "", 首次访问浏览器,搜索框中不会显示前端页面中的 placeholder="Search for..." 属性
    search_data = request.GET.get('query', "")
    if search_data:
        data_dict["username__contains"] = search_data

    queryset = Admin.objects.filter(**data_dict).order_by("id")

    # queryset = Admin.objects.all()
    page_object = Pagination(request, queryset, page_size=2)
    page_queryset = page_object.page_queryset
    page_object.html()
    page_string = page_object.page_string

    context = {
        "page_queryset": page_queryset,
        "page_string": page_string,
        "search_data": search_data,
    }
123456789101112131415161718192021222324252627
```

修改 `/root/python/myproject/myproject/urls.py`

```
from django.contrib import admin
from django.urls import path
from employee_management.views import depart,user,pretty,admin
```

```python

urlpatterns = [

    # path('admin/', admin.site.urls),

    # 部门管理
    path('depart/list/', depart.depart_list),
    path('depart/add/', depart.depart_add),
    path('depart/delete/', depart.depart_delete),
    path('depart/<int:nid>/edit/', depart.depart_edit),

    # 用户管理
    path('user/list/', user.user_list),
    path('user/model/form/add/', user.user_model_form_add),
    path('user/<int:nid>/edit/', user.user_edit),
    path('user/<int:nid>/delete/', user.user_delete),

    # 靓号管理
    path('pretty/list/', pretty.pretty_list),
    path('pretty/add/', pretty.pretty_add),
    path('pretty/<int:nid>/edit/', pretty.pretty_edit),
    path('pretty/<int:nid>/delete/', pretty.pretty_delete),

    # 管理员管理
    path('admin/list/', admin.admin_list),
]
12345678910111213141516171819202122232425262728 29 30
```



管理员添加

修改`myproject/employee_management/views/admin.py`

```python
from employee_management.utils.modelform import BootStrapModelForm
from django import forms


class AdminModelForm(BootStrapModelForm):
    confirm_password = forms.CharField(
        label="确认密码",
        widget=forms.PasswordInput,
    )

```

```
    class Meta:
        model = Admin
        fields = ["username", "password", "confirm_password"]
        widgets = {
            "password": forms.PasswordInput
        }

def admin_add(request):
    """添加管理员"""

    form = AdminModelForm

    context = {
        "form": form,
        "title": "新建管理员",
    }

    return render(request, "change.html", context)
```

> 按照之前的套路,我应该会新建一个名为 `admin_add.html` 的HTML文件来展示添加界面
> 这次我们优化一下
> 不管是 `pretty_add.html` 或者 `user_add.html` ,除了标题不一样,其他的都是一样的
> 没有必要在创建一个单独的HTML页面,我们只需要在一个公共的HTML页面中传入各自的标题即可

新建 `myproject/employee_management/templates/change.html`

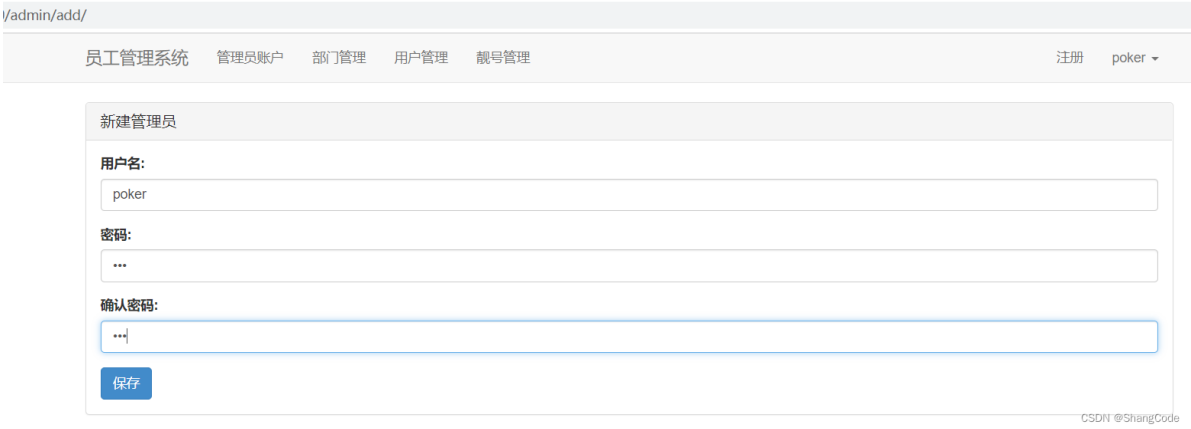
```
{% extends 'layout.html' %}

{% block content %}

<div class="container">
    <div class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-title">{{ title }}</h3>
        </div>
        <div class="panel-body">
            <form method="post" novalidate>
                {% csrf_token %}

                {% for field in form %}
                    <div class="form-group">
                        <label>{{ field.label }}: </label>
                        {{ field }}
                        <!-- 数据校验,显示错误信息 -->
                        <span style="color: red;">{{ field.errors.0 }}</span>
                    </div>
                {% endfor %}

                <button type="submit" class="btn btn-primary">保存</button>
            </form>
        </div>
    </div>
</div>

```

```
29    {% endblock %}
30    </body>
31
32    </html>
33    12345678910111213141516171819202122232425262728293031321
```

修改 `myproject/myproject/urls.py`

```
1    path('admin/add/', admin.admin_add),
2    1
```

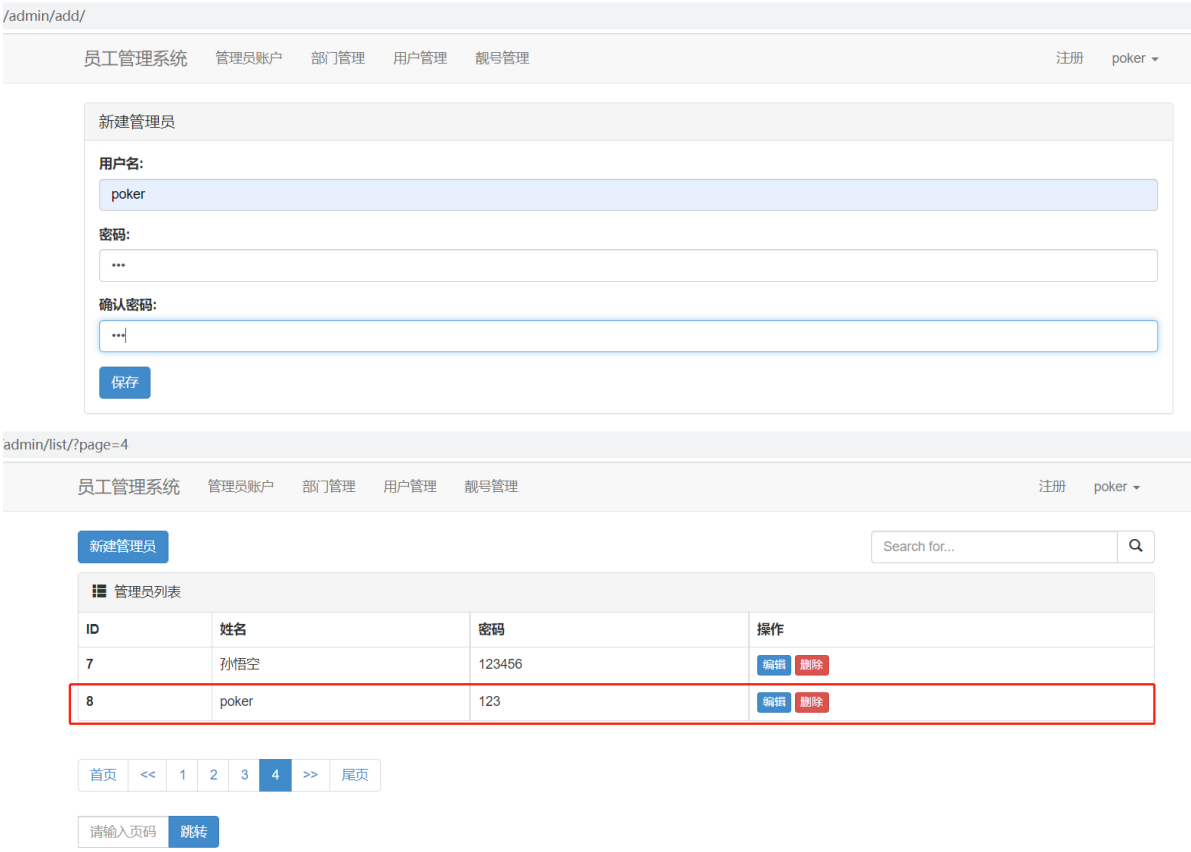
浏览器访问



> 将数据保存到数据库

修改 `myproject/employee_management/views/admin.py`

```python
def admin_add(request):
    """添加管理员"""

    title = "新建管理员"

    if request.method == "GET":
        form = AdminModelForm()
        return render(request, "change.html", {"form": form, "title": title})

    # 如果是POST请求
    form = AdminModelForm(data=request.POST)

    context = {
        "form": form,
        "title": title,
    }

    if form.is_valid():
        form.save()
        return redirect("/admin/list")


    return render(request, "change.html", context)
12345678910111213141516171819202122231
```



> 但是现在还有一个问题,如果两次输入的密码不一致呢?

修改`myproject/employee_management/views/admin.py`,增加**钩子函数**`clean_confirm_password`

```python
class AdminModelForm(BootStrapModelForm):
    confirm_password = forms.CharField(
        label="确认密码",
        widget=forms.PasswordInput,
    )

    class Meta:
        model = Admin
        fields = ["username", "password", "confirm_password"]
        widgets = {
            "password": forms.PasswordInput
        }

    # 钩子函数
    def clean_confirm_password(self):
        pwd = self.cleaned_data.get("password")
        confirm = self.cleaned_data.get("confirm_password")
        if confirm != pwd:
            raise ValidationError("密码不一致!")
        return confirm
```

浏览器测试

可以发现,**数据校验**成功了
但是还有个问题,当我们点击保存后,密码和密码确认框中的数据清空了
这样不是太友好,因此我们需要添加一个属性

修改`myproject/employee_management/views/admin.py`

```
render_value=True
1
```

```python
class AdminModelForm(BootStrapModelForm):
    confirm_password = forms.CharField(
        label="确认密码",
        widget=forms.PasswordInput(render_value=True),
    )

    class Meta:
        model = Admin
        fields = ["username", "password", "confirm_password"]
        widgets = {
            "password": forms.PasswordInput(render_value=True),
        }

    # 钩子函数
    def clean_confirm_password(self):
        pwd = self.cleaned_data.get("password")
        confirm = self.cleaned_data.get("confirm_password")
        if confirm != pwd:
            raise ValidationError("密码不一致!")

        # return返回什么,字段 confirm_password 保存至数据库的值就是什么
        return confirm
```

然后测试,就不会出现自动清空的情况了

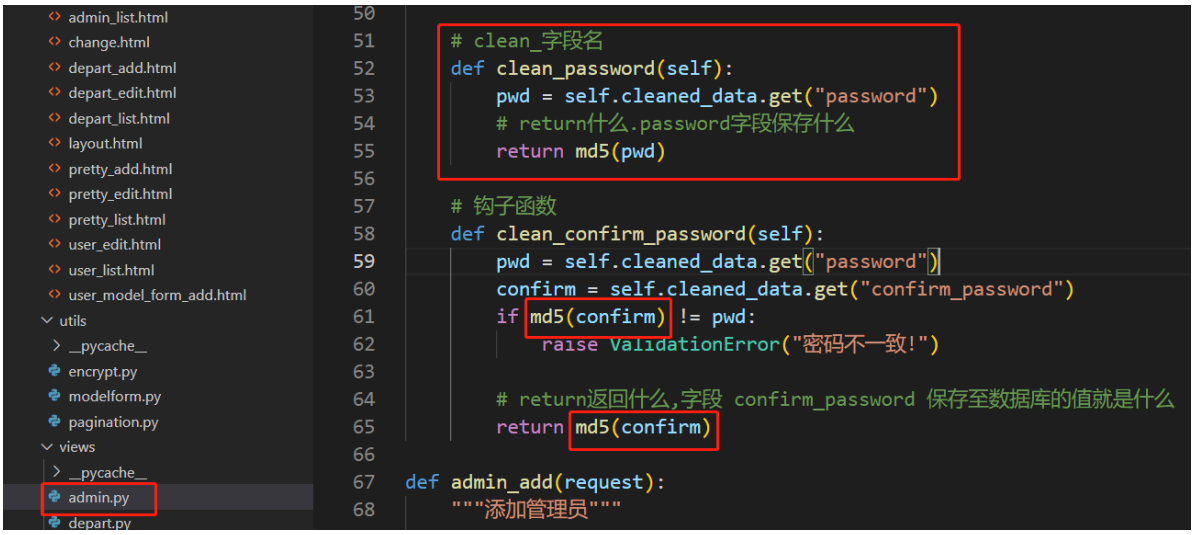
> 现在又出现一个问题
> 密码这样存储进数据库,很不安全
> 我们最好对其进行加密后,再进行保存

这里我们使用`md5`方式进行加密
新建`/root/python/myproject/employee_management/utils/encrypt.py`文件

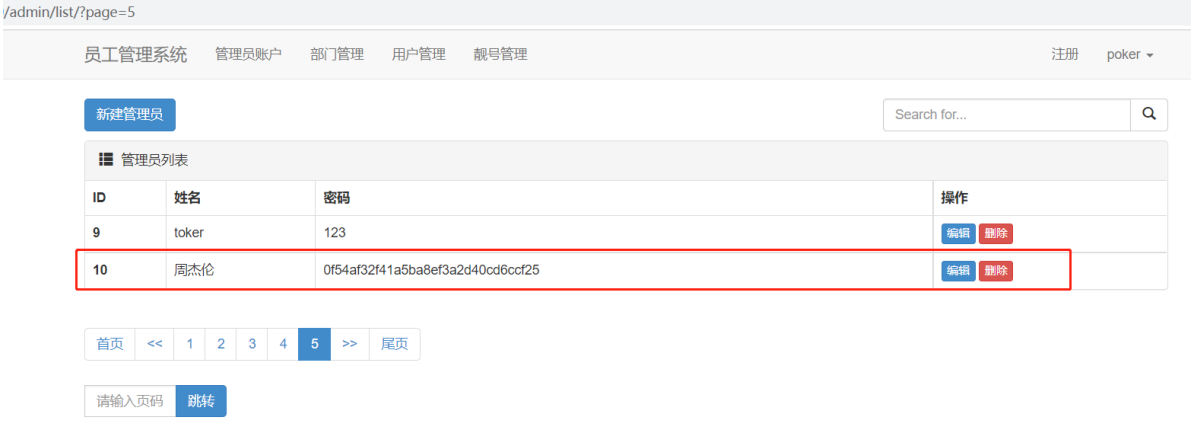
```python
import hashlib
from django.conf import settings

def md5(data_string):
    obj = hashlib.md5(settings.SECRET_KEY.encode('utf-8'))
    obj.update(data_string.encode('utf-8'))
    return obj.hexdigest()
```

修改 `myproject/employee_management/views/admin.py`

```python
# clean_字段名
def clean_password(self):
    pwd = self.cleaned_data.get("password")
    # return什么.password字段保存什么
    return md5(pwd)

# 钩子函数
def clean_confirm_password(self):
    pwd = self.cleaned_data.get("password")
    confirm = self.cleaned_data.get("confirm_password")
    if md5(confirm) != pwd:
        raise ValidationError("密码不一致!")

    # return返回什么,字段 confirm_password 保存至数据库的值就是什么
    return md5(confirm)
```



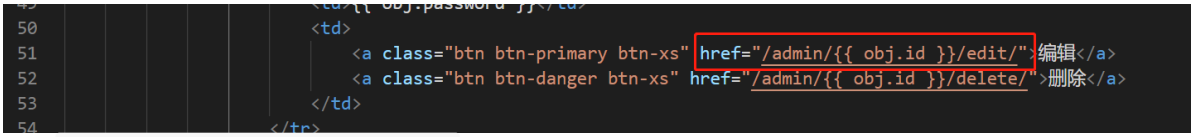
浏览器添加新用户测试



管理员编辑

修改 `myproject/employee_management/templates/admin_list.html`

```
<a class="btn btn-primary btn-xs" href="/admin/{{ obj.id }}/edit/">编辑</a>
```



修改 `myproject/myproject/urls.py`

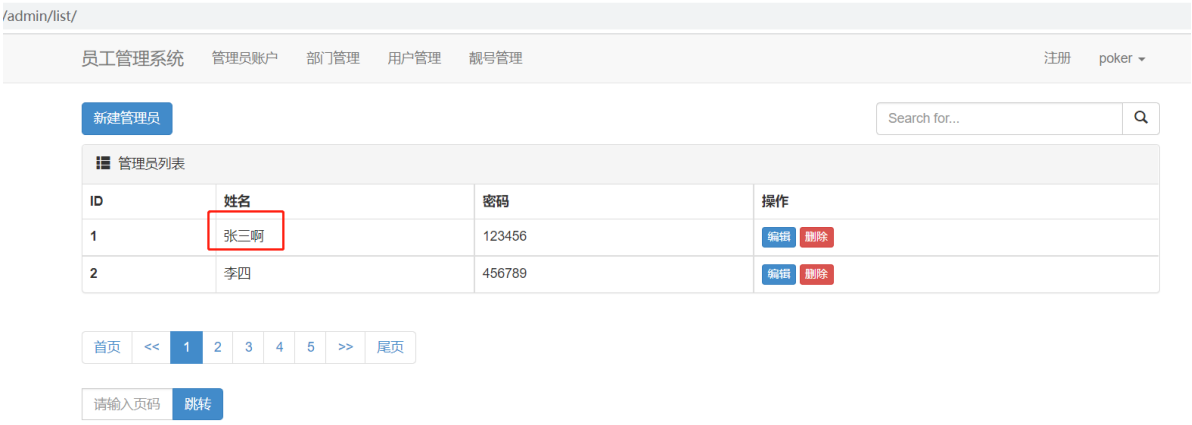
```
path('admin/<int:nid>/edit/', admin.admin_edit),
```

修改 `myproject/employee_management/views/admin.py`

```
# 如果不想让用户修改密码,只能修改用户名,那么使用下面这个AdminEditModelForm
# 如果都可以修改,直接用上面的AdminModelForm即可
class AdminEditModelForm(BootStrapModelForm):
    class Meta:
        model = Admin
        fields = ["username"]

def admin_edit(request, nid):

    # 判断 nid 是否存在
    row_object = Admin.objects.filter(id=nid).first()
    if not row_object:
        return render(request, "error.html", {"msg": "数据不存在!"})

    """编辑管理员"""

    title = "编辑管理员"

    if request.method == "GET":
        form = AdminEditModelForm(instance=row_object)
        return render(request, "change.html", {"form": form, "title": title})

    form = AdminEditModelForm(data=request.POST, instance=row_object)
    if form.is_valid():
        form.save()
        return redirect('/admin/list/')

    return render(request, "change.html", {"form": form, "title": title})
```

浏览器访问,修改 张三 为 张三啊

点击"保存"



管理员删除

修改 myproject/employee_management/templates/admin_list.html

```
<a class="btn btn-danger btn-xs" href="/admin/{{ obj.id }}/delete/">删除</a>
```

修改 myproject/myproject/urls.py

```
path('admin/<int:nid>/delete/', admin.admin_delete),
```

修改 myproject/employee_management/views/admin.py

```
def admin_delete(request, nid):
    """删除管理员"""
    Admin.objects.filter(id=nid).delete()
    return redirect("/admin/list/")
```

管理员重置密码

修改 myproject/employee_management/templates/admin_list.html

```
<th>重置密码</th>

...

<td>
    <a class="btn btn-primary btn-xs" href="/admin/{{ obj.id }}/edit/">编辑</a>
    <a class="btn btn-danger btn-xs" href="/admin/{{ obj.id }}/delete/">删除</a>
</td>
```

修改`myproject/myproject/urls.py`

```
path('admin/<int:nid>/reset/', admin.admin_reset),
1
```

修改`myproject/employee_management/views/admin.py`

```
class AdminResetModelForm(BootStrapModelForm):

    confirm_password = forms.CharField(
        label = "确认密码",
        widget = forms.PasswordInput(render_value=True),
    )

    class Meta:
        model = Admin
        fields = ["password", "confirm_password"]
        widgets = {
            "password": forms.PasswordInput(render_value=True)
        }

    # clean_字段名
    def clean_password(self):
        pwd = self.cleaned_data.get("password")

        # 校验当前数据库中的密码与用户输入的新密码是否一致
        exists = Admin.objects.filter(id=self.instance.pk, password=md5(pwd))
        if exists:
            raise ValidationError("密码不能与当前密码一致!")

        # return什么.password字段保存什么
        return md5(pwd)

    # 钩子函数
    def clean_confirm_password(self):
        pwd = self.cleaned_data.get("password")
        confirm = self.cleaned_data.get("confirm_password")
        if md5(confirm) != pwd:
            raise ValidationError("密码不一致!")

```

```python
            # return返回什么,字段 confirm_password 保存至数据库的值就是什么
            return md5(confirm)

    def admin_reset(request, nid):
        """重置管理员密码"""

        # 判断 nid 是否存在
        row_object = Admin.objects.filter(id=nid).first()
        if not row_object:
            return render(request, "error.html", {"msg": "数据不存在!"})

        title = "重置密码 - {}".format(row_object.username)

        if request.method == "GET":
            form = AdminResetModelForm(instance=row_object)

            return render(request, "change.html", {"title": title, "form": form})

        form = AdminResetModelForm(data=request.POST, instance=row_object)
        if form.is_valid():
            form.save()
            return redirect("/admin/list/")

        return render(request, "change.html", {"title": title, "form": form})
```

> 不知道为什么下面的"密码不一致错误也出来了",后面待解决

账户登录

登录界面

修改`myproject/employee_management/templates/layout.html`

```
{% block css %}
<style>

    .navbar {
        border-radius: 0;
    }

</style>

{% endblock %}
```

新建 `myproject/employee_management/templates/login.html`

```
{% extends 'layout.html' %}


{% block css %}
<style>
    .account {
        width: 400px;
        border: 1px solid #dddddd;
        border-radius: 5px;
        box-shadow: 5px 5px 20px #aaa;

        margin-left: auto;
        margin-right: auto;
        margin-top: 100px;
        padding: 20px 40px;
    }

    .account h2 {
        margin-top: 10px;
        text-align: center;
    }
</style>
{% endblock %}


{% block content %}
<div class="account">
    <h2>用户登录</h2>
    <div class="panel-body">
        <form method="post">
```

```
            {% csrf_token %}
            <div class="form-group">
                <label>用户名</label>
                <input type="text" class="form-control" placeholder="用户名" name="username">
            </div>
            <div class="form-group">
                <label>密码</label>
                <input type="password" class="form-control" placeholder="密码" name="password">
            </div>
            <button type="submit" class="btn btn-primary center-block" style="width: 80px;">登录</button>
        </form>
    </div>
</div>
{% endblock %}
```

修改 `myproject/myproject/urls.py`

```
# 登录
path('login/', account.login),
```

新建 `myproject/employee_management/views/account.py`

```
from django.shortcuts import render

def login(request):
    """登录"""

    return render(request, 'login.html')
```

浏览器简单访问

用户名密码验证

修改 `myproject/employee_management/templates/login.html`

```
{% extends 'layout.html' %}


```

```
{% block css %}
<style>
    .account {
        width: 400px;
        border: 1px solid #dddddd;
        border-radius: 5px;
        box-shadow: 5px 5px 20px #aaa;

        margin-left: auto;
        margin-right: auto;
        margin-top: 100px;
        padding: 20px 40px;
    }

    .account h2 {
        margin-top: 10px;
        text-align: center;
    }
</style>
{% endblock %}


{% block content %}
<div class="account">
    <h2>用户登录</h2>
    <div class="panel-body">
        <form method="post" novalidate>
            {% csrf_token %}
            <div class="form-group">
                <label>用户名</label>
                {{ form.username }}
                <span style="color: red;">{{ form.errors.username.0 }}</span>

            </div>
            <div class="form-group">
                <label>密码</label>
                {{ form.password }}
                <span style="color: red;">{{ form.errors.password.0 }}</span>
            </div>

            <button type="submit" class="btn btn-primary center-block"
style="width: 80px;">登录</button>
        </form>
    </div>
</div>
{% endblock %}
```

修改 `myproject/employee_management/views/account.py`

```
from django.shortcuts import render, HttpResponse
from django import forms
from employee_management.utils.modelform import BootStrapForm
from employee_management.utils.encrypt import md5
from employee_management.models import Admin

# 这一次不使用ModelForm,使用Form来实现
```

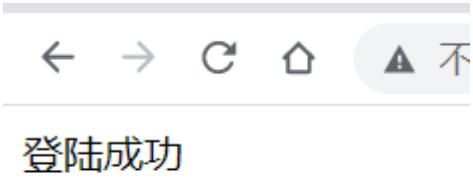
```python
class LoginForm(BootStrapForm):
    username = forms.CharField(
        label="用户名",
        widget=forms.TextInput(attrs={"class": "form-control"}),
        required=True,
    )
    password = forms.CharField(
        label="用户名",
        # render_value=True 表示当提交后,如果密码输入错误,不会自动清空密码输入框的内容
        widget=forms.PasswordInput(attrs={"class": "form-control"}, ),
        required=True,
    )

    def clean_password(self):
        pwd = self.cleaned_data.get("password")
        return md5(pwd)


def login(request):
    """登录"""
    if request.method == "GET":
        form = LoginForm()
        return render(request, 'login.html', {"form": form})

    form = LoginForm(data=request.POST)
    if form.is_valid():
        # 验证成功, 获取到的用户名和密码
        # print(form.cleaned_data)
        # {'username': '123', 'password': '123'}
        # {'username': '456', 'password': '0f54af32f41a5ba8ef3a2d40cd6ccf25'}

        # 去数据库校验用户名和密码是否正确
        admin_object = Admin.objects.filter(**form.cleaned_data).first()
        # 如果数据库中没有查询到数据
        if not admin_object:
            # 手动抛出错误显示在"password"字段下
            form.add_error("password", "用户名或密码错误")
            return render(request, 'login.html', {"form": form})
        return HttpResponse("登陆成功")

    return render(request, 'login.html', {"form": form})
```

浏览器进行测试

新建用户 `toker` (必须为后来使用 **md5加密** 过密码的用户,不能是明文密码的用户)

输入错误的密码

输入正确的密码



配置cookie与session

修改 `myproject/employee_management/views/account.py`

```
def login(request):
    """登录"""
    if request.method == "GET":
        form = LoginForm()
        return render(request, 'login.html', {"form": form})

    form = LoginForm(data=request.POST)
    if form.is_valid():
        # 验证成功, 获取到的用户名和密码
        print(form.cleaned_data)
        # {'username': '123', 'password': '123'}
        # {'username': '456', 'password': '0f54af32f41a5ba8ef3a2d40cd6ccf25'}

        # 去数据库校验用户名和密码是否正确
        admin_object = Admin.objects.filter(**form.cleaned_data).first()
        if not admin_object:
            form.add_error("password", "用户名或密码错误")
            return render(request, 'login.html', {"form": form})

        # 如果用户名密码正确
        # 网站生成随机字符创,写到用户浏览器的cookie中,再写入到服务器的session中
        request.session["info"] = {'id': admin_object.id, 'name':
admin_object.username}
        return redirect("/admin/list/")

    return render(request, 'login.html', {"form": form})
```

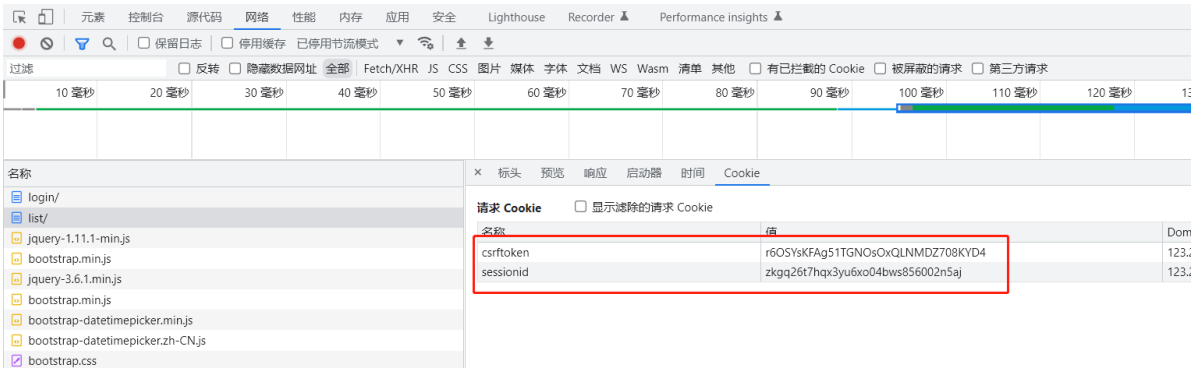
```python
def login(request):
    """登录"""
    if request.method == "GET":
        form = LoginForm()
        return render(request, 'login.html', {"form": form})

    form = LoginForm(data=request.POST)
    if form.is_valid():
        # 验证成功，获取到的用户名和密码
        print(form.cleaned_data)
        # {'username': '123', 'password': '123'}
        # {'username': '456', 'password': '0f54af32f41a5ba8ef3a2d40cd6ccf25'}

        # 去数据库校验用户名和密码是否正确
        admin_object = Admin.objects.filter(**form.cleaned_data).first()
        if not admin_object:
            form.add_error("password", "用户名或密码错误")
            return render(request, 'login.html', {"form": form})

        # 如果用户名密码正确
        # 网站生成随机字符创,写到用户浏览器的cookie中,再写入到服务器的session中
        request.session["info"] = {'id': admin_object.id, 'name': admin_object.username}
        return redirect("/admin/list/")

    return render(request, 'login.html', {"form": form})

```

登录成功后,跳转到 管理员列表 界面

session 信息保存在了服务器中的Mysql数据库 django_session 表中

```
mysql> use my_project;

mysql> show tables;
+--------------------------------+
| Tables_in_my_project           |
+--------------------------------+
| auth_group                     |
| auth_group_permissions         |
| auth_permission                |
| auth_user                      |
| auth_user_groups               |
| auth_user_user_permissions     |
| django_admin_log               |
| django_content_type            |
| django_migrations              |
| django_session                 |
| employee_management_admin      |
| employee_management_department |
| employee_management_prettynum  |
| employee_management_userinfo   |
+--------------------------------+

mysql> select * from django_session;
+----------------------------------+-----------------------------------------
----------------------------------------------------+----------------------------
+
| session_key                      | session_data
                                                   | expire_date                |
+----------------------------------+-----------------------------------------
----------------------------------------------------+----------------------------
+
```

```
| zkgq26t7hqx3yu6xo04bws856002n5aj |
eyJpbmZvIjp7ImlkIjoxMiwibmFtZSI6InRva2VyIn19:1pElim:Tus2mHaJUTNTfzhppuah8N0FVdLXQ
xyvRk_4n-4fP6g | 2023-01-23 06:33:24.373104 |
+----------------------------------+-----------------------------------------
----------------------------------------------------+--------------------------
+
```



> 当实现了这个功能后,我们需要对每个页面做验证,只有登陆的人才可以访问这些页面,这个功能在下一节的中间件中来实现

中间件实现登录验证

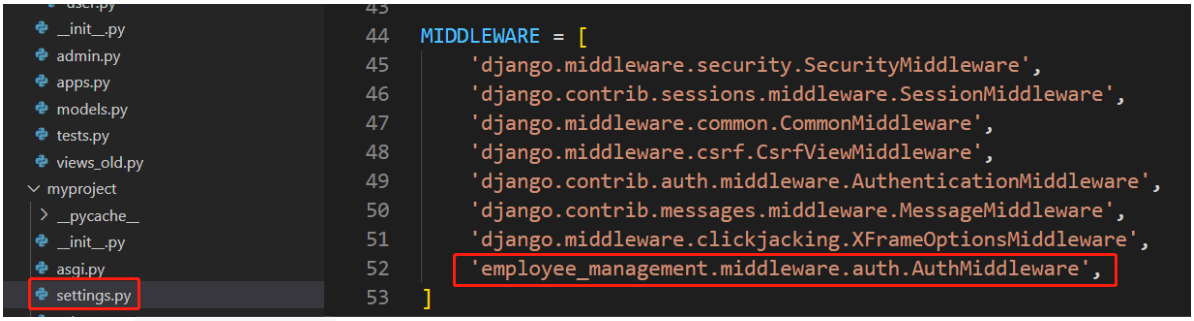
> 中间件会在视图函数下的每个方法执行前调用,不用在每个方法下面进行判断,不然函数太多,过于繁琐。

新建`myproject/employee_management/middleware`目录
在其下新建`myproject/employee_management/middleware/auth.py`文件

```python
from django.utils.deprecation import MiddlewareMixin
from django.shortcuts import HttpResponse, redirect


class AuthMiddleware(MiddlewareMixin):

    def process_request(self, request):

        # 0.排除不需要的页面
        if request.path_info == "/login/":
            return

        # 1.读取当前访问的用户的session信息,如果能读到,说明已登录过,就可以继续向后走
        info_dict = request.session.get("info")
        if info_dict:
            return

        # 2.如果没有登录信息
        return redirect("/login/")
```

修改`myproject/myproject/urls.py`

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'employee_management.middleware.auth.AuthMiddleware',
]
```



浏览器访问 `/pretty/list/` 进行测试

你会发现只要你没登录过,不管你访问什么页面,都会跳转到 `login` 登录界面

> 登录校验的功能算是实现了,但是登录的用户目前无法退出,下一节进行介绍

用户注销

> 小插曲: 我刚才整理代码的时候才发现,我因为之前使用了 `datetimepicker` 的日期插件,粘贴代码的时候,将我原来css样式给覆盖掉了,其实是我粘贴多了,需要先改一下,不然右上角的按钮无法展开,抱歉抱歉

修改 `myproject/employee_management/templates/layout.html` ,将下面的几行注释掉

```
thon > myproject > employee_management > templates > layout.html > html > head > style > .navbar
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>

    <link rel="stylesheet" href="/static/plugins/bootstrap-3.4.1/css/bootstrap.css">
    <link rel="stylesheet" href="/static/plugins/font-awesome-4.7.0/css/font-awesome.css">


    <!--JQUERY-->
    <!-- <script type="text/javascript" src="/static/jquery/jquery-1.11.1-min.js"></script> -->
    <!--BOOTSTRAP框架-->
    <!-- <link rel="stylesheet" type="text/css" href="/static/jquery/bootstrap_3.3.0/css/bootstrap.min.css">
    <script type="text/javascript" src="/static/jquery/bootstrap_3.3.0/js/bootstrap.min.js"></script> -->
    <!--BOOTSTRAP_DATETIMEPICKER插件-->
    <link rel="stylesheet" type="text/css" href="/static/jquery/bootstrap-datetimepicker-master/css/bootstrap-datetimepicker.min.css">
    {% block css %}
    <style>

        .navbar {
            border-radius: 0;
        }

    </style>

    {% endblock %}
</head>

<body>
```

配置注销按钮的跳转URL地址

然后保存就可以了

> 接下来实现注销功能

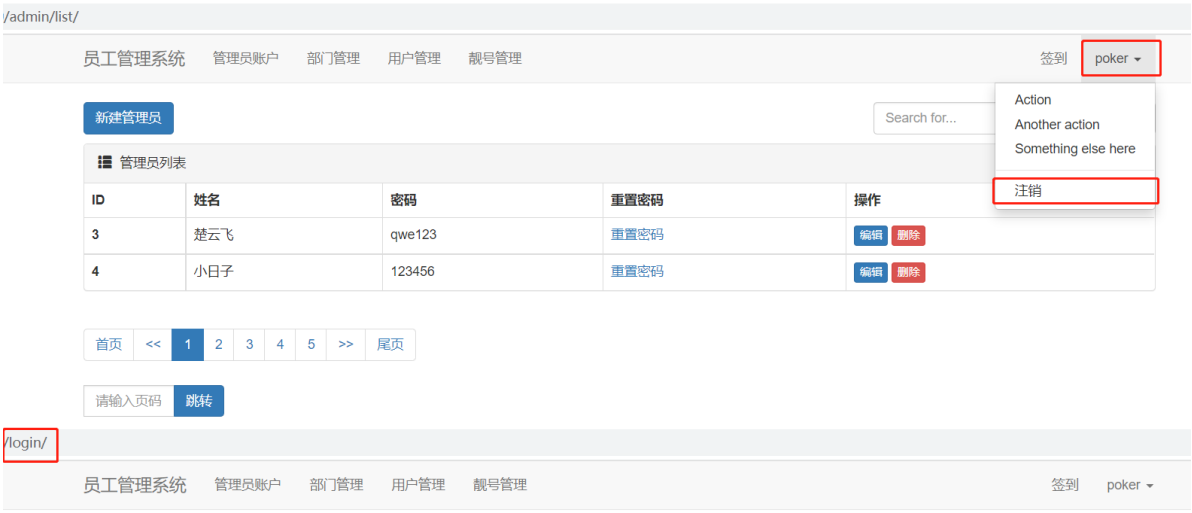
修改 `myproject/employee_management/views/account.py`

```python
def logout(request):
    """ 注销 """

    # 清楚当前session
    request.session.clear()

    return redirect("/login/")
```

修改 `myproject/myproject/urls.py`

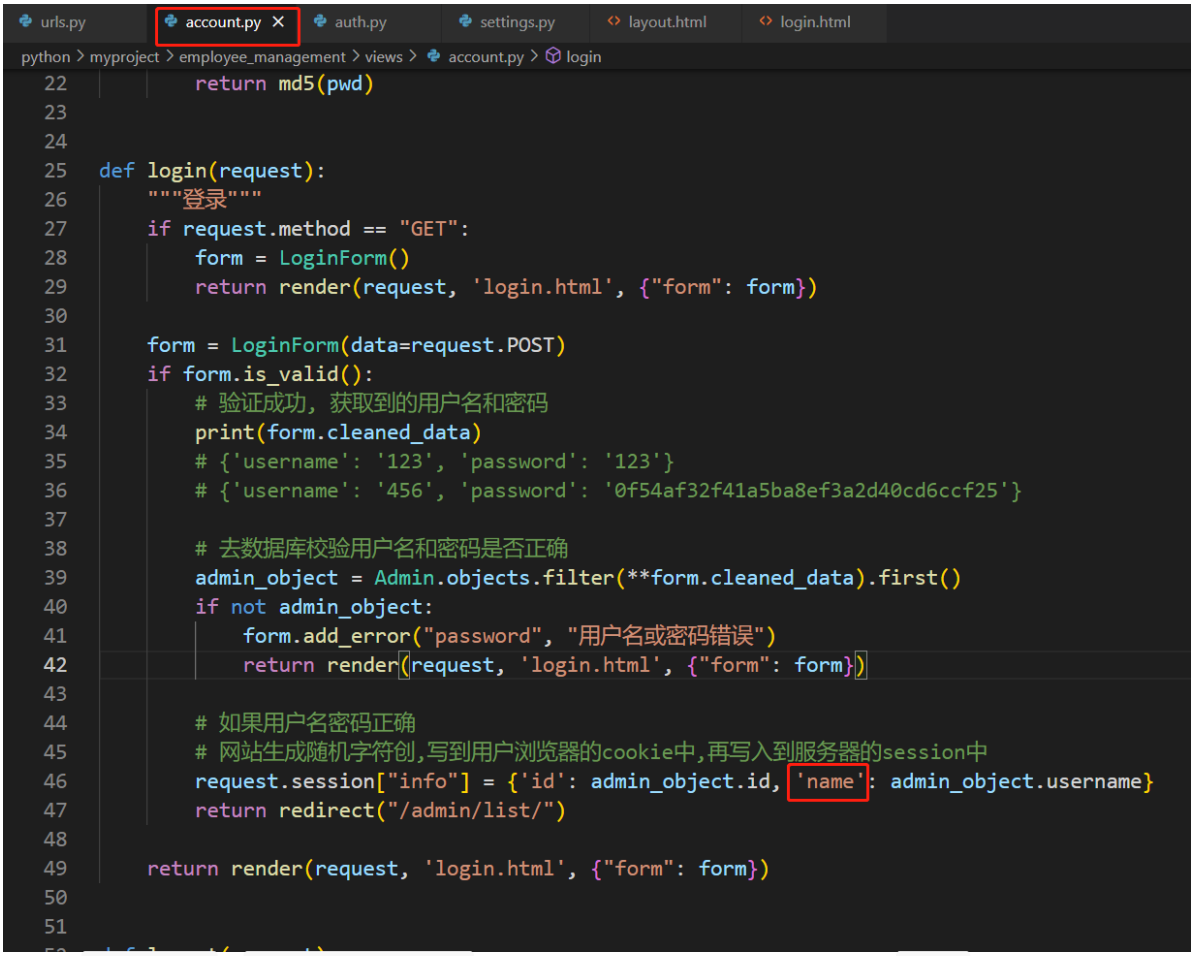
```python
path('logout/', account.logout),

```

浏览器测试



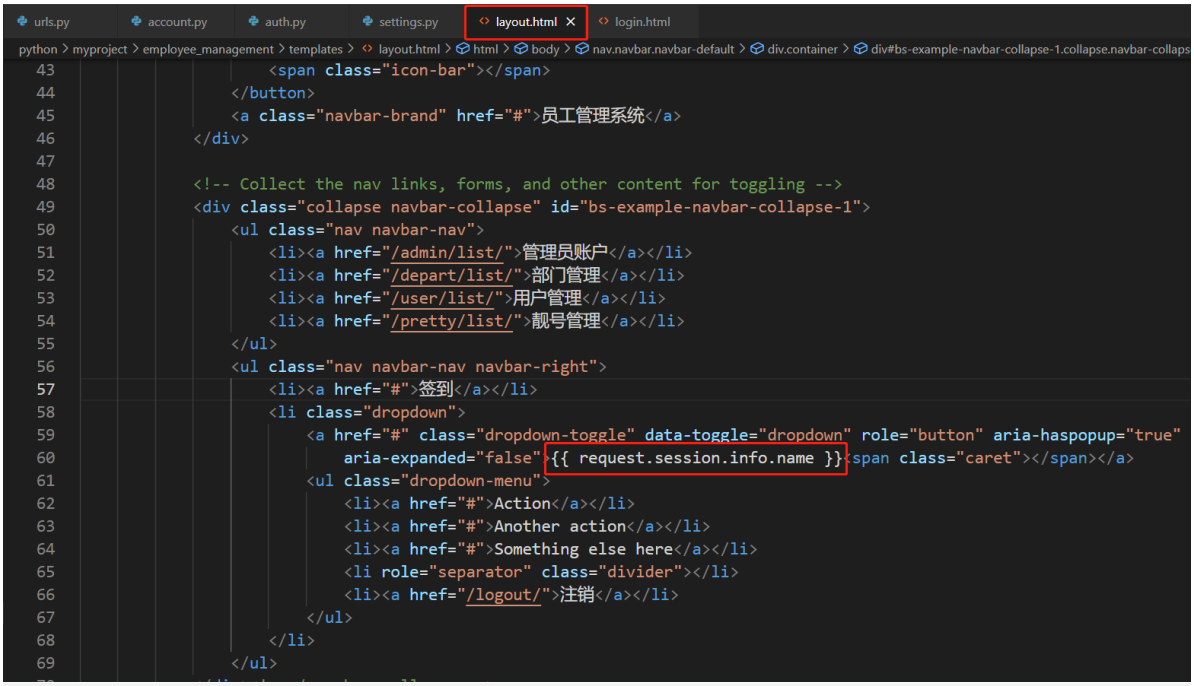
自动跳至login

登录成功后,右上角的用户名还不是目前登录的用户,所以我们还需要再做一次修改



因为在 `login函数` 中 `request.session` 中定义的username字段对应的名称为 `name` ,所以我们可以在前端代码中直接调用

```
<a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true"
aria-expanded="false">{{ request.session.info.name }}<span class="caret"></span></a>
```



图片验证码

> 生成验证码参考链接: https://www.cnblogs.com/wupeiqi/articles/5812291.html

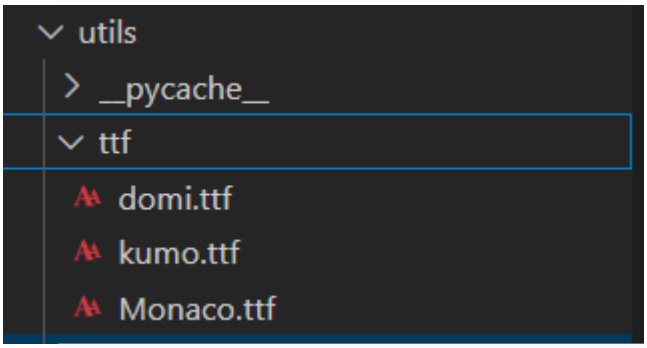
下载必要的软件

```
pip3 install pillow
```

下载字体

> 点击这里: **验证码字体下载**

新建 `myproject/employee_management/utils/ttf` 目录,将字体放在其下



新建 `myproject/employee_management/utils/code.py` ,编辑验证码生成函数

```python
from PIL import Image, ImageDraw, ImageFilter, ImageFont
import random


def check_code(width=120, height=30, char_length=5, font_file='employee_management/utils/ttf/Monaco.ttf', font_size=28):
    code = []
    img = Image.new(mode='RGB', size=(width, height), color=(255, 255, 255))
    draw = ImageDraw.Draw(img, mode='RGB')

    def rndChar():
        """
        生成随机字母
        :return:
        """
        return chr(random.randint(65, 90))

    def rndColor():
        """
        生成随机颜色
        :return:
        """
        return (random.randint(0, 255), random.randint(10, 255), random.randint(64, 255))

    # 写文字
    font = ImageFont.truetype(font_file, font_size)
    for i in range(char_length):
        char = rndChar()
        code.append(char)
        h = random.randint(0, 4)
        draw.text([i * width / char_length, h], char, font=font, fill=rndColor())

    # 写干扰点
    for i in range(40):
        draw.point([random.randint(0, width), random.randint(0, height)], fill=rndColor())

    # 写干扰圆圈
    for i in range(40):
        draw.point([random.randint(0, width), random.randint(0, height)], fill=rndColor())
        x = random.randint(0, width)
        y = random.randint(0, height)
```

```
            draw.arc((x, y, x + 4, y + 4), 0, 90, fill=rndColor())

        # 画干扰线
        for i in range(5):
            x1 = random.randint(0, width)
            y1 = random.randint(0, height)
            x2 = random.randint(0, width)
            y2 = random.randint(0, height)

            draw.line((x1, y1, x2, y2), fill=rndColor())

        img = img.filter(ImageFilter.EDGE_ENHANCE_MORE)
        return img,''.join(code)


    if __name__ == '__main__':
        img, code_str = check_code()
        print(code_str)

        with open('code.png', 'wb') as f:
            img.save(f, format='png')
```

修改`myproject/employee_management/views/account.py`,引入调用刚刚写的验证码生成函数

```
    from employee_management.utils.code import check_code
    from django.shortcuts import HttpResponse
    from io import BytesIO

    def image_code(request):
        """ 生成图片验证码 """
        # 调用pillow函数,生成图片
        img, code_string = check_code()

        # 将图片保存到内存
        stream = BytesIO()
        img.save(stream, 'png')
        return HttpResponse(stream.getvalue())
```

修改`myproject/myproject/urls.py`

```
    path('image/code/', account.image_code),
```

还需要将该函数的URL加入访问白名单,使验证码链接可用

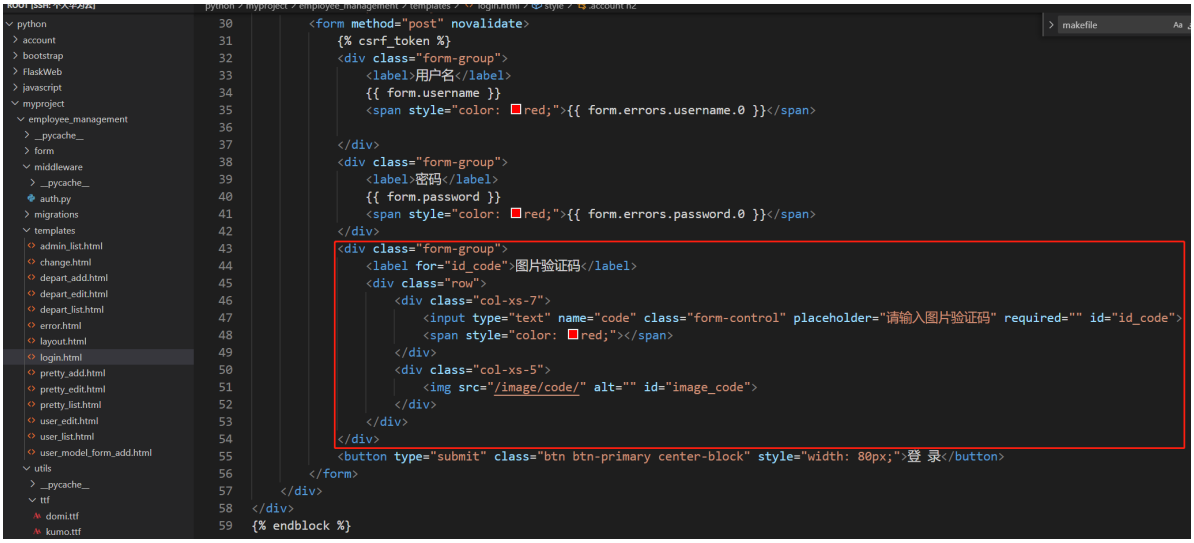
编辑`myproject/employee_management/middleware/auth.py`

```
    # 0.排除不需要的页面
    if request.path_info in ["/login/", "/image/code/"]:
        return
```

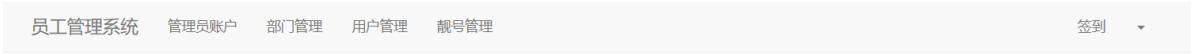
修改`myproject/employee_management/templates/login.html`

```
    {% extends 'layout.html' %}


    {% block css %}
    <style>
```

```
    .account {
        width: 400px;
        border: 1px solid #dddddd;
        border-radius: 5px;
        box-shadow: 5px 5px 20px #aaa;

        margin-left: auto;
        margin-right: auto;
        margin-top: 100px;
        padding: 20px 40px;
    }

    .account h2 {
        margin-top: 10px;
        text-align: center;
    }
</style>
{% endblock %}


{% block content %}
<div class="account">
    <h2>用户登录</h2>
    <div class="panel-body">
        <form method="post" novalidate>
            {% csrf_token %}
            <div class="form-group">
                <label>用户名</label>
                {{ form.username }}
                <span style="color: red;">{{ form.errors.username.0 }}</span>

            </div>
            <div class="form-group">
                <label>密码</label>
                {{ form.password }}
                <span style="color: red;">{{ form.errors.password.0 }}</span>
            </div>
            <div class="form-group">
                <label for="id_code">图片验证码</label>
                <div class="row">
                    <div class="col-xs-7">
                        <input type="text" name="code" class="form-control" placeholder="请输入图片验证码" required="" id="id_code">
                        <span style="color: red;"></span>
                    </div>
                    <div class="col-xs-5">
                        <img src="/image/code/" alt="" id="image_code">
                    </div>
                </div>
            </div>
            <button type="submit" class="btn btn-primary center-block" style="width: 80px;">登 录</button>
        </form>
    </div>
</div>
{% endblock %}
```



浏览器访问



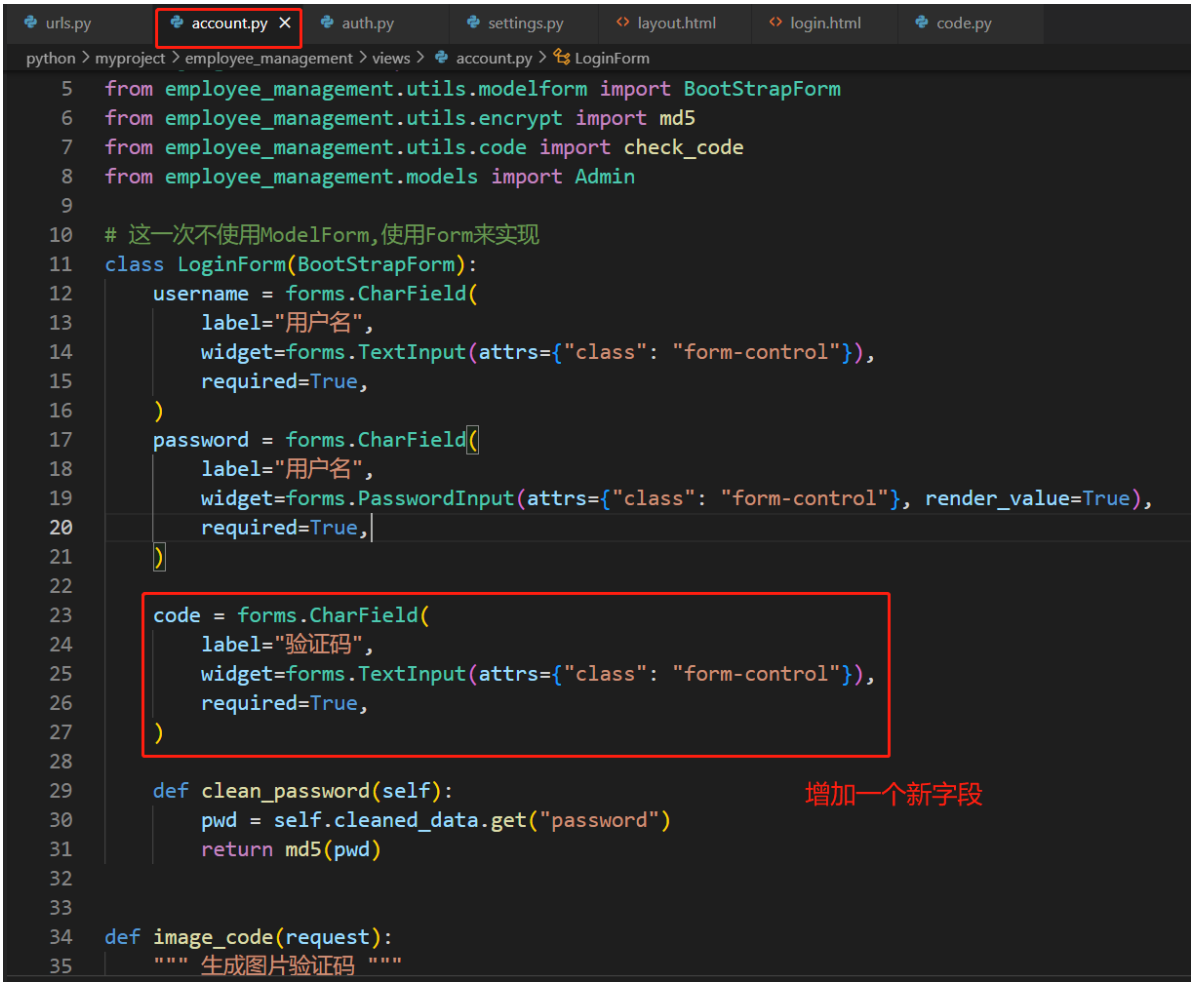
验证码的校验

> 接下来需要验证用户输入的验证码与生成的验证码是否一致

修改 `myproject/employee_management/views/account.py`

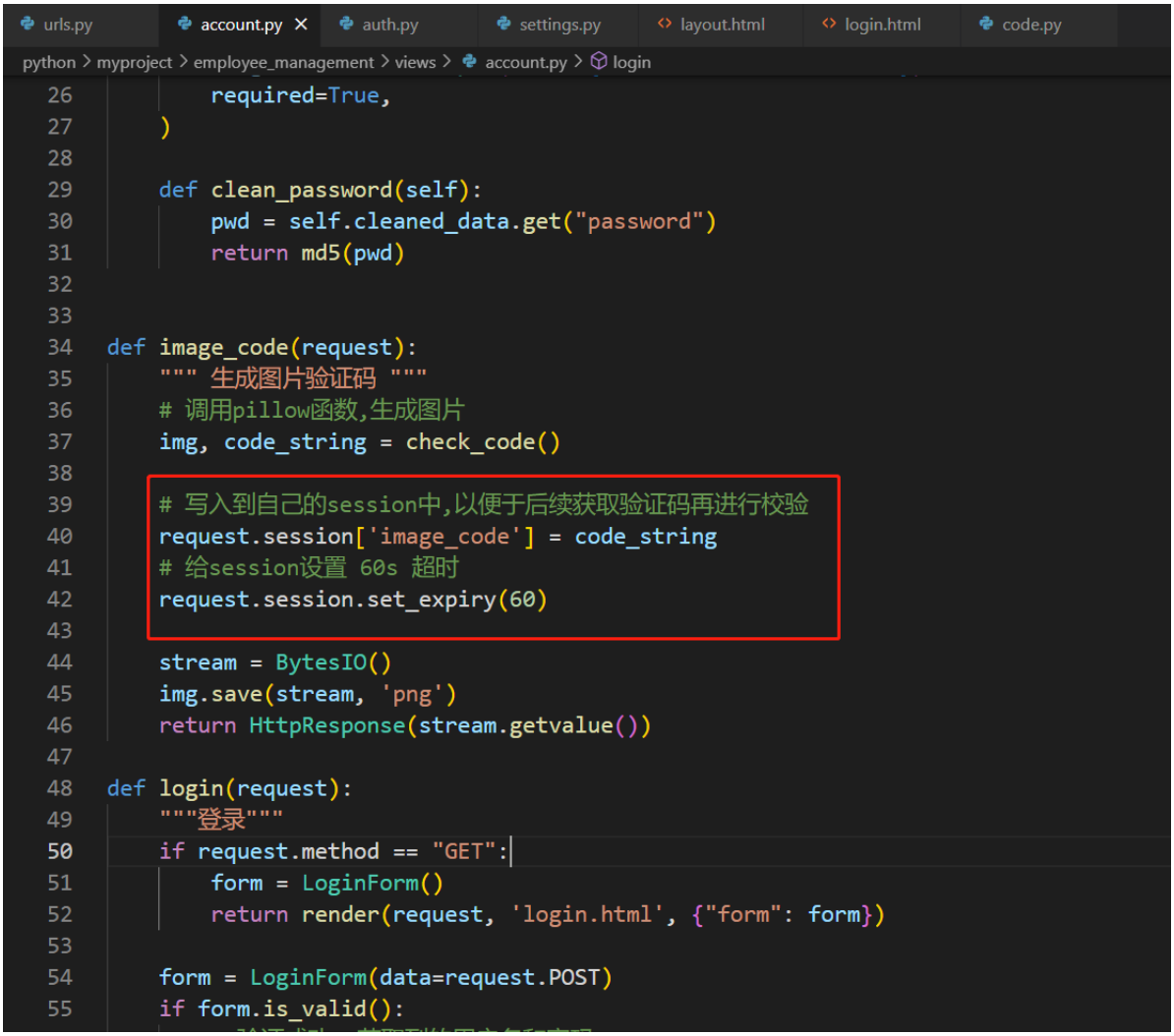
- 第一处

```
code = forms.CharField(
    label="验证码",
    widget=forms.TextInput(attrs={"class": "form-control"}),
    required=True,
)
```



- 第二处

```python
def image_code(request):
    """ 生成图片验证码 """
    # 调用pillow函数,生成图片
    img, code_string = check_code()

    # 写入到自己的session中,以便于后续获取验证码再进行校验
    request.session['image_code'] = code_string
    # 给session设置 60s 超时
    request.session.set_expiry(60)

    stream = BytesIO()
    img.save(stream, 'png')
    return HttpResponse(stream.getvalue())
```



- 第三处

```
def login(request):
    """登录"""
    if request.method == "GET":
        form = LoginForm()
        return render(request, 'login.html', {"form": form})

    form = LoginForm(data=request.POST)
    if form.is_valid():
        # 验证成功, 获取到的用户名和密码
        print(form.cleaned_data)
        # {'username': '123', 'password': '123'}
        # {'username': '456', 'password': '0f54af32f41a5ba8ef3a2d40cd6ccf25'}

        # 验证码的校验
        user_input_code = form.cleaned_data.pop('code')
        image_code = request.session.get('image_code', "")
        print("user_input_code={}, image_code={}".format(user_input_code,
image_code))
        if image_code.upper() != user_input_code.upper():
            form.add_error("code", "验证码错误")
            return render(request, 'login.html', {"form": form})

        # 去数据库校验用户名和密码是否正确
        admin_object = Admin.objects.filter(**form.cleaned_data).first()
        if not admin_object:
```

```
            form.add_error("password", "用户名或密码错误")
            return render(request, 'login.html', {"form": form})

        # 如果用户名密码和验证码正确
        # 网站生成随机字符创,写到用户浏览器的cookie中,再写入到服务器的session中
        request.session["info"] = {'id': admin_object.id, 'name':
admin_object.username}
        # 重新设置session的超时时间,因为之前设置的session的超时时间的 60s
        request.session.set_expiry(60 * 60 * 24)

        return redirect("/admin/list/")

    return render(request, 'login.html', {"form": form})
```

urls.py | account.py | auth.py | settings.py | layout.html | login.html | code.py

python > myproject > employee_management > views > account.py > login

```python

def login(request):
    """登录"""
    if request.method == "GET":
        form = LoginForm()
        return render(request, 'login.html', {"form": form})

    form = LoginForm(data=request.POST)
    if form.is_valid():
        # 验证成功，获取到的用户名和密码
        print(form.cleaned_data)
        # {'username': '123', 'password': '123'}
        # {'username': '456', 'password': '0f54af32f41a5ba8ef3a2d40cd6ccf25'}

        # 验证码的校验
        user_input_code = form.cleaned_data.pop('code')
        image_code = request.session.get('image_code', "")
        print("user_input_code={}, image_code={}".format(user_input_code, image_code))
        if image_code.upper() != user_input_code.upper():
            form.add_error("code", "验证码错误")
            return render(request, 'login.html', {"form": form})

        # 去数据库校验用户名和密码是否正确
        admin_object = Admin.objects.filter(**form.cleaned_data).first()
        if not admin_object:
            form.add_error("password", "用户名或密码错误")
            return render(request, 'login.html', {"form": form})

        # 如果用户名密码和验证码正确
        # 网站生成随机字符创,写到用户浏览器的cookie中,再写入到服务器的session中
        request.session["info"] = {'id': admin_object.id, 'name': admin_object.username}
        # 重新设置session的超时时间,因为之前设置的session的超时时间的 60s
        request.session.set_expiry(60 * 60 * 24)

        return redirect("/admin/list/")

    return render(request, 'login.html', {"form": form})

```

浏览器登录测试

首页 << 1 2 3 4 5 >> 尾页

请输入页码 跳转

> 现在已经可以正常登录,但是还可以进行优化
> 我们想要点击验证码图片进行刷新验证码,很简单

修改`myproject/employee_management/templates/login.html`

```
1    <img src="/image/code/" alt="" id="image_code"
     onclick="this.setAttribute('src','/image/code/?random='+Math.random())">
```

点击验证码图片后,验证码可刷新