

页面美化及时间戳转换

页面美化及时间戳转换

1. Flask-Bootstrap扩展库使用
 - 1.1. 安装扩展库
 - 1.2. 初始化扩展库
2. 原页面美化改造
 - 2.1. base页面改造
 - 2.2. 登录页面改造
 - 2.3. 注册页面改造
 - 2.4. 首页改造
 - 2.5. 用户资料页面改造
 - 2.6. 用户资料修改页面改造
 - 2.7. 申请重置密码、重置密码、错误页面改造
3. 日期时间转换
 - 3.1. 安装扩展库Flask-Moment
 - 3.2. 注册扩展库
 - 3.3. 注册moment.js库
 - 3.4. 渲染用户资料页的时间戳
 - 3.5. 渲染帖子模板的时间戳
 - 3.6. 渲染效果

1. Flask-Bootstrap扩展库使用

1.1. 安装扩展库

Flask的插件Flask-Bootstrap(<https://pythonhosted.org/Flask-Bootstrap/>), 提供了一个已准备好的基础模板, 该模板引入了Bootstrap框架。通过安装该扩展库, 快速实现页面美化。

```
pip install flask-bootstrap
```

1.2. 初始化扩展库

修改 app/__init__.py 脚本, 初始化 flask-bootstrap 扩展库。

```

.....
from flask_bootstrap import Bootstrap
.....
# 实例化flask_bootstrap
bootstrap = Bootstrap()

def create_app(test_config=None):
    .....
    # 初始化flask_bootstrap
    bootstrap.init_app(application)

```

2. 原页面美化改造

2.1. base页面改造

原base页面 `app/templates/base.html`，需要修改为继承 `bootstrap/base.html`。改造之后，其他继承至 `app/templates/base.html` 的页面均需要修改为将内容填充至 `app_content` 块中。

新增 `scripts` 块用于添加 `js` 代码，同时通过 `super()` 函数首先继承 `bootstrap/base.html` 父模板中的 `js` 代码。

改造后的base页面如下：

```

{% extends 'bootstrap/base.html' %}

{% block title %}
    {% if title %}
        {{ title }} - 博客
    {% else %}
        博客
    {% endif %}
{% endblock %}

{% block navbar %}
    <nav class="navbar navbar-default">
        <div class="container">
            <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="{{ url_for('index') }}">博客</a>
            </div>
            <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                <ul class="nav navbar-nav">
                    <li><a href="{{ url_for('index') }}">首页</a></li>
                    <li><a href="{{ url_for('explore') }}">发现</a></li>
                </ul>
                <ul class="nav navbar-nav navbar-right">

```

```

        {% if current_user.is_anonymous %}
            <li><a href="{{ url_for('login') }}">登录</a></li>
        {% else %}
            <li><a href="{{ url_for('user.user_info',
username=current_user.username) }}">个人资料</a></li>
            <li><a href="{{ url_for('logout') }}">退出</a></li>
        {% endif %}
    </ul>
</div>
</div>
</nav>
{% endblock %}

{% block content %}
    <div class="container">
        {% with messages = get_flashed_messages() %}
            {% if messages %}
                {% for message in messages %}
                    <div class="alert alert-info" role="alert">{{ message }}</div>
                {% endfor %}
            {% endif %}
        {% endwith %}

        {# 其他页面内容均填充至此块中 #}
        {% block app_content %}{% endblock %}
    </div>
{% endblock %}

{% block scripts %}
    {{ super() }}
{% endblock %}

```

2.2. 登录页面改造

修改 `app/templates/login/login.html` 模板文件，修改内容如下：

- 通过 `import 'bootstrap/wtf.html' as wtf` 方式可以导入扩展库中的表单快速生成的宏，来快速实现表单的处理；
- 将原来的 `content` 块修改为 `app_content` 块。

```

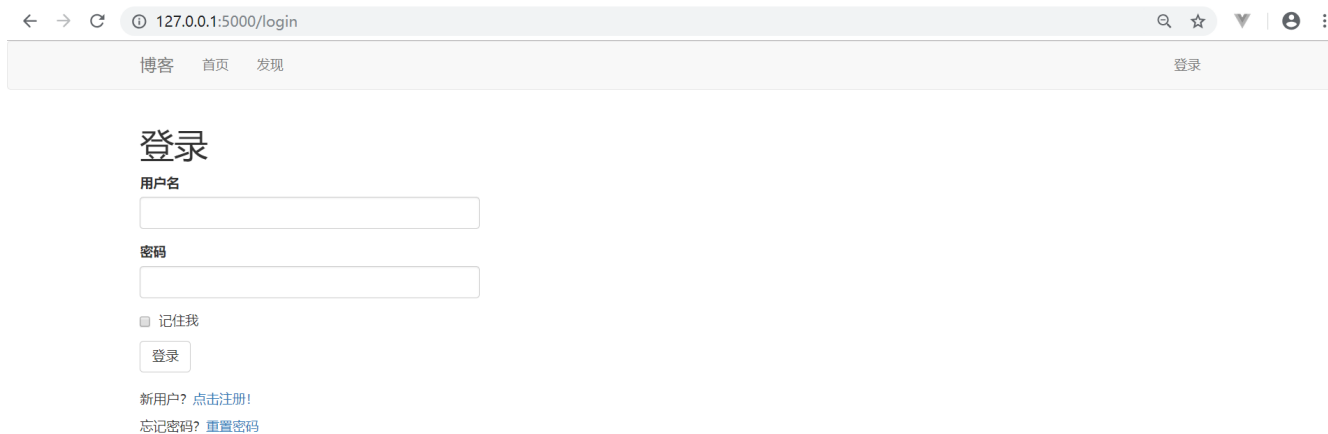
{% extends 'base.html' %}
{% import 'bootstrap/wtf.html' as wtf %}

{% block app_content %}
    <h1>登录</h1>
    <div class="row">
        <div class="col-md-4">
            {{ wtf.quick_form(form) }}
        </div>
    </div>
    <br>
    <p>新用户? <a href="{{ url_for('register') }}">点击注册! </a></p>

```

```
<p>忘记密码? <a href="{{ url_for('reset_password_request') }}">重置密码</a></p>
{% endblock %}
```

实现效果如下：



浏览器地址栏显示: 127.0.0.1:5000/login

页面顶部导航栏: 博客 首页 发现 登录

登录页面内容:

登录

用户名

密码

☐ 记住我

登录

新用户? [点击注册!](#)

忘记密码? [重置密码](#)

2.3. 注册页面改造

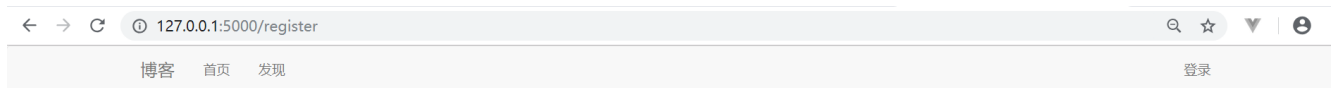
修改 `app/templates/login/register.html` 模板文件，修改内容如下：

- 通过 `import 'bootstrap/wtf.html' as wtf` 方式可以导入扩展库中的表单快速生成的宏，来快速实现表单的处理；
- 将原来的 `content` 块修改为 `app_content` 块。

```
{% extends 'base.html' %}
{% import 'bootstrap/wtf.html' as wtf %}

{% block app_content %}
    <h1>注册</h1>
    <div class="row">
        <div class="col-md-4">
            {{ wtf.quick_form(form) }}
        </div>
    </div>
{% endblock %}
```

实现效果如下：



注册

用户名

邮箱

密码

确认密码

注册

2.4. 首页改造

修改 `app/templates/index/index.html` 文件，修改内容如下：

- 通过 `import 'bootstrap/wtf.html' as wtf` 方式可以导入扩展库中的表单快速生成的宏，来快速实现表单的处理；
- 将原来的 `content` 块修改为 `app_content` 块；
- 给分页增加特定样式，若无上一页或下一页，按钮设置为禁用状态。

```
{% extends 'base.html' %}
{% import 'bootstrap/wtf.html' as wtf %}

{% block app_content %}
    <h1>Hi, {{ current_user.username }}!</h1>
    {% if form %}
        {{ wtf.quick_form(form) }}
    {% endif %}
    <br>
    {% for post in posts %}
        {% include 'common/post.html' %}
    {% endfor %}
    <br>
    <nav aria-label="...">
        <ul class="pager">
            <li class="previous{% if not prev_url %} disabled {% endif %}">
                <a href="{{ prev_url or '#' }}">
                    <span aria-hidden="true">&larr;</span> 上一页
                </a>
            </li>
            <li>第{{ page }}页</li>
            <li class="next{% if not next_url %} disabled {% endif %}">
                <a href="{{ next_url or '#' }}">
                    下一页 <span aria-hidden="true">&rarr;</span>
                </a>
            </li>
        </ul>
    </nav>
```

```
{% endblock %}
```

修改帖子模板 `app/templates/common/post.html` 文件，修改内容如下：

- 增加样式

```
<table class="table table-hover">
  <tr>
    <td width="70px">
      <a href="{{ url_for('user.user_info', username=post.author.username) }}">
        
      </a>
    </td>
    <td>
      <a href="{{ url_for('user.user_info', username=post.author.username) }}">
        {{ post.author.username }}
      </a>
      说:<br>{{ post.body }}
    </td>
  </tr>
</table>
```

实现效果如下：

博客 首页 发现 个人资料 退出

Hi, john!

内容

提交

 john 说:
111

← 上一页 第1页 下一页 →

2.5. 用户资料页面改造

修改 `app/templates/user/user_info.html` 文件，修改内容如下：

- 将原来的 `content` 块修改为 `app_content` 块；
- 给表格、分页增加特定样式。

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

{% block app_content %}
  <table class="table table-hover">
    <tr>
      <td width="256px"></td>
      ....
    </tr>
  </table>
{% endblock %}
```

```

        </tr>
    </table>
    <br>
    {% for post in posts %}
        {% include 'common/post.html' %}
    {% endfor %}
    <br>
    <nav aria-label="...">
        <ul class="pager">
            <li class="previous{% if not prev_url %} disabled {% endif %}">
                <a href="{% if prev_url or '#' %}">
                    <span aria-hidden="true">&larr;</span> 上一页
                </a>
            </li>
            <li>第{{ page }}页</li>
            <li class="next{% if not next_url %} disabled {% endif %}">
                <a href="{% if next_url or '#' %}">
                    下一页 <span aria-hidden="true">&rarr;</span>
                </a>
            </li>
        </ul>
    </nav>
{% endblock %}

```

实现效果如下：



2.6. 用户资料修改页面改造

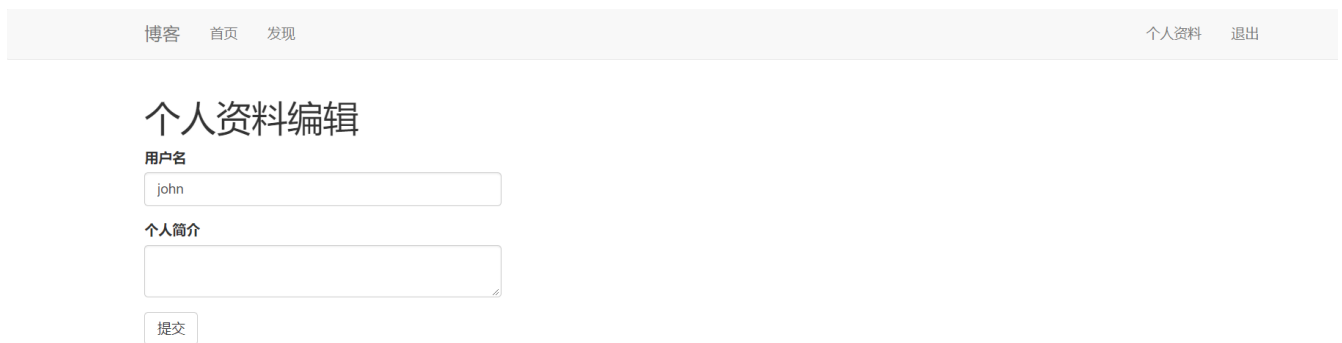
修改 `app/templates/user/user_info_edit.html` 文件，修改内容如下：

- 通过 `import 'bootstrap/wtf.html' as wtf` 方式可以导入扩展库中的表单快速生成的宏，来快速实现表单的处理；
- 将原来的 `content` 块修改为 `app_content` 块。

```
{% extends 'base.html' %}
{% import 'bootstrap/wtf.html' as wtf %}

{% block app_content %}
    <h1>个人资料编辑</h1>
    <div class="row">
        <div class="col-md-4">
            {{ wtf.quick_form(form) }}
        </div>
    </div>
{% endblock %}
```

实现效果如下：



2.7. 申请重置密码、重置密码、错误页面改造

改造方式同上。

3. 日期时间转换

由于该博客服务可能会被全球不同地区的用户访问，因此每个用户访问系统是，系统都应该显示其对应区域的日期时间。所以服务端记录的时间均为UTC时区标准时间，然后通过 `Flask-Moment` 扩展库实现浏览器端的日期时间转换。

3.1. 安装扩展库Flask-Moment

```
pip install flask-moment
```

3.2. 注册扩展库

修改 `app/__init__.py` 脚本，增加 `flask-moment` 扩展库的注册


```

.....
from flask_moment import Moment
.....
# 实例化flask_moment
moment = Moment()

def create_app(test_config=None):
    .....
    # 初始化flask_moment
    moment.init_app(application)

```

3.3. 注册moment.js库

修改 `app/templates/base.html` 文件，增加 `moment.js` 开源库的引入，便于所有页面均有效。Flask-Moment的 `moment.include_moment()` 函数可以生成了一个 `<script>` 标签并在其中包含 `moment.js`。

```

.....
{% block scripts %}
    {{ super() }}
    {{ moment.include_moment() }}
{% endblock %}

```

3.4. 渲染用户资料页的时间戳

修改 `app/templates/user/user_info.html` 文件，进行用户资料页的时间戳转换。其中，`moment()` 函数用于时间戳转换，转换格式如下：

```

moment('2017-09-28T21:45:23Z').format('L')
"09/28/2017"
moment('2017-09-28T21:45:23Z').format('LL')
"September 28, 2017"
moment('2017-09-28T21:45:23Z').format('LLL')
"September 28, 2017 2:45 PM"
moment('2017-09-28T21:45:23Z').format('LLLL')
"Thursday, September 28, 2017 2:45 PM"
moment('2017-09-28T21:45:23Z').format('dddd')
"Thursday"
moment('2017-09-28T21:45:23Z').fromNow()
"7 hours ago"
moment('2017-09-28T21:45:23Z').calendar()
"Today at 2:45 PM"

```

修改代码如下：

```

.....
        {% if user.last_seen %}
            最后访问于: <p>{{ moment(user.last_seen).format('LLL') }}</p>
        {% endif %}
.....

```

3.5. 渲染帖子模板的时间戳

修改 `app/templates/common/post.html` 文件，进行帖子模板的时间戳转换。

```
.....
    <td>
        <a href="{{ url_for('user.user_info', username=post.author.username) }}">
            {{ post.author.username }}
        </a>
        说 {{ moment(post.timestamp).fromNow() }}:
        <br>
        {{ post.body }}
    </td>
.....
```

3.6. 渲染效果

用户资料页时间戳转换



用户帖子页时间戳转换

