

用户信息弹窗及私信功能实现

用户信息弹窗及私信功能实现

1. 用户信息弹窗
 - 1.1. 编写用户弹窗信息视图
 - 1.2. 编写用户信息弹窗模板
 - 1.3. 新增弹出窗口关联的DOM元素
 - 1.4. 悬停事件实现
 - 1.5. 启动服务测试效果
2. 发送私信功能
 - 2.1. 新增私信数据库模型
 - 2.2. 编写发送私信表单类
 - 2.3. 新增发送私信模板
 - 2.4. 编写发送私信视图类
 - 2.5. 个人主页增加发送私信链接
 - 2.6. 启动服务测试效果
3. 查看私信功能
 - 3.1. 编写查看私信视图类
 - 3.2. 新增查看私信模板
 - 3.3. 导航栏增加查看私信链接
 - 3.4. 更新翻译文件
 - 3.5. 启动服务测试效果
4. 消息通知徽章功能
 - 4.1. 静态消息通知徽章
 - 4.2. 动态消息通知徽章模板修改
 - 4.3. 新增通知模型
 - 4.4. 更新用户通知
 - 4.5. 编写通知视图类
 - 4.6. 客户端轮询通知
 - 4.7. 启动服务测试效果

1. 用户信息弹窗

1.1. 编写用户弹窗信息视图

修改 `app/main/views.py` 脚本，编写用户弹窗信息视图。

```
1 class UserPopupView(view):
2     """用户资料弹出框"""
3     methods = ['GET']
4     decorators = [login_required]
5
6     def dispatch_request(self, username):
7         user = User.query.filter_by(username=username).first_or_404()
8         return render_template('user_popup.html', user=user)
```

修改 `app/main/__init__.py` 脚本，注册视图。

```
1 def register_views(bp):
2     # 在函数中引入可以避免循环依赖问题
3     from app.main.views import IndexView, ExploreView, UserInfoView, UserInfoEditView,
4     FollowView, \
5     UnfollowView, SearchView, UserPopupView
6     # .....
7     # 用户资料弹出框
8     bp.add_url_rule('/info/<username>/popup',
9                     view_func=UserPopupView.as_view('user_popup'))
```

1.2. 编写用户信息弹窗模板

新增 `app/templates/user_popup.html` 模板文件，编写用户信息弹窗展示。

```
1 <table class="table">
2     <tr>
3         <td width="64px" style="border: 0px;"></td>
5         <td style="border: 0px;">
6             <p>
7                 <a href="{{ url_for('main.user_info', username=user.username) }}">
8                     {{ user.username }}
9                 </a>
10            </p>
11            <small>
12                {% if user.about_me %}
13                    <p>{{ user.about_me }}</p>
14                {% endif %}
15                {% if user.last_seen %}
16                    {{ _('最后访问于') }}: <p>{{ moment(user.last_seen).format('LLL') }}
17            </p>
18            {% endif %}
19            <p>
20                {{ _('粉丝') }} {{ user.followers.count() }}, {{ _('关注') }} {{
21                user.followed.count() }}
22            </p>
23            {% if user == current_user %}
24                <p><a href="{{ url_for('main.user_info_edit') }}">{{ _('编辑您的资
25                料') }}</a></p>
26            {% elif not current_user.is_following(user) %}
27                <p><a href="{{ url_for('main.follow', username=user.username) }}">
28                    {{ _('关注') }}</a></p>
29            {% else %}
30                <p><a href="{{ url_for('main.unfollow', username=user.username)
31                    }}">{{ _('取消关注') }}</a></p>
32            {% endif %}
33        </small>
34    </td>
35</tr>
36</table>
```

1.3. 新增弹出窗口关联的DOM元素

修改 `app/templates/_post.html` 模板文件，增加弹出窗口关联的DOM元素。新增 `` 元素，将 `<a>` 元素封装在 `` 元素中，然后将悬停事件和弹出窗口与 `` 相关联，关联方式是通过其样式类 `user_popup` 来关联实现。新创建的弹窗元素会被自动的添加到 `` 元素内的 `<a>` 元素后面。

```
1 .....
2         {% set user_link %}
3             <span class="user_popup">
4                 <a href="{{ url_for('main.user_info',
5 username=post.author.username) }}">
6                     {{ post.author.username }}
7                 </a>
8             </span>
9         {% endset %}
10 .....
```

1.4. 悬停事件实现

通过修改 `app/templates/base.html` 模板文件，新增基于ajax的悬停事件。其中通过CSS类选择器可以选定页面中新增的样式类为 `user_popup` 的 `` 元素。其中使用到的 `const` 是ES6语法中新增的变量定义方式，定义的一般为只读常量，其作用于属于块级作用域。

使用jQuery，可以通过调用 `element.hover(handlerIn, handlerOut)` 将悬停事件附加到任何HTML元素。如果在元素集合上调用这个函数，jQuery方便地将事件附加到所有元素上。这两个参数是两个函数，分别在用户将鼠标指针移入和移出目标元素时调用对应的函数。

```
1 {% block scripts %}
2     {{ super() }}
3     {{ moment.include_moment() }}
4     {{ moment.lang(g.locale) }}
5     <script>
6         $(function(){
7             $('.user_popup').hover(
8                 function(event) {
9                     const elem = $(event.currentTarget);
10                    .....
11                },
12                function(event) {
13                    const elem = $(event.currentTarget);
14                    .....
15                }
16            )
17        })
18    </script>
19 {% endblock %}
```

通过增加定时器的方式，可以实现鼠标停留1秒后激活弹出行为的悬停延迟效果。在鼠标移入函数中，增加1秒的定时器，鼠标悬停1秒后，进行弹出窗口创建，在鼠标移出函数中则取消定时器。

```

1      $(function(){
2          let timer = null;
3          $('.user_popup').hover(
4              function(event) {
5                  const elem = $(event.currentTarget);
6                  timer = setTimeout(function() {
7                      timer=null;
8                      // -----弹出窗口实现代码----- //
9                  }, 1000);
10             },
11             function(event) {
12                 const elem = $(event.currentTarget);
13                 if (timer) {
14                     clearTimeout(timer);
15                     timer=null;
16                 }
17             }
18         )
19     })

```

定时器内增加1秒后的弹窗创建处理。通过ajax方式请求后台的 /info/<username>/popup 服务，done 中为请求成功后的回调函数，用于创建 manual 模式的弹窗对象，该模式的弹窗对象可手工创建、销毁。

在鼠标移出函数中增加处理，若鼠标悬停在1秒内，则直接停止定时器；若鼠标悬停超过1秒，但是ajax通讯处理中，则停止ajax通讯；若ajax通讯正常处理完成了，则将弹窗销毁。

```

1      $(function(){
2          let timer = null;
3          let xhr = null;
4          $('.user_popup').hover(
5              function(event) {
6                  // 获取span元素
7                  const elem = $(event.currentTarget);
8                  // 创建定时器
9                  timer = setTimeout(function() {
10                      timer=null;
11                      // 通讯后台，获取弹窗展示页面
12                      xhr = $.ajax(
13                          `/info/${elem.first().text().trim()}/popup`).done(
14                              function(data){
15                                  xhr = null;
16                                  // 创建弹窗对象并展示
17                                  elem.popover({
18                                      trigger: 'manual',
19                                      html: true,
20                                      animation: false,
21                                      container: elem,
22                                      content: data
23                                  }).popover('show');
24                                  // 渲染通过Ajax添加新的Flask-Moment元素
25                                  flask_moment_render_all();
26                              }
27              );

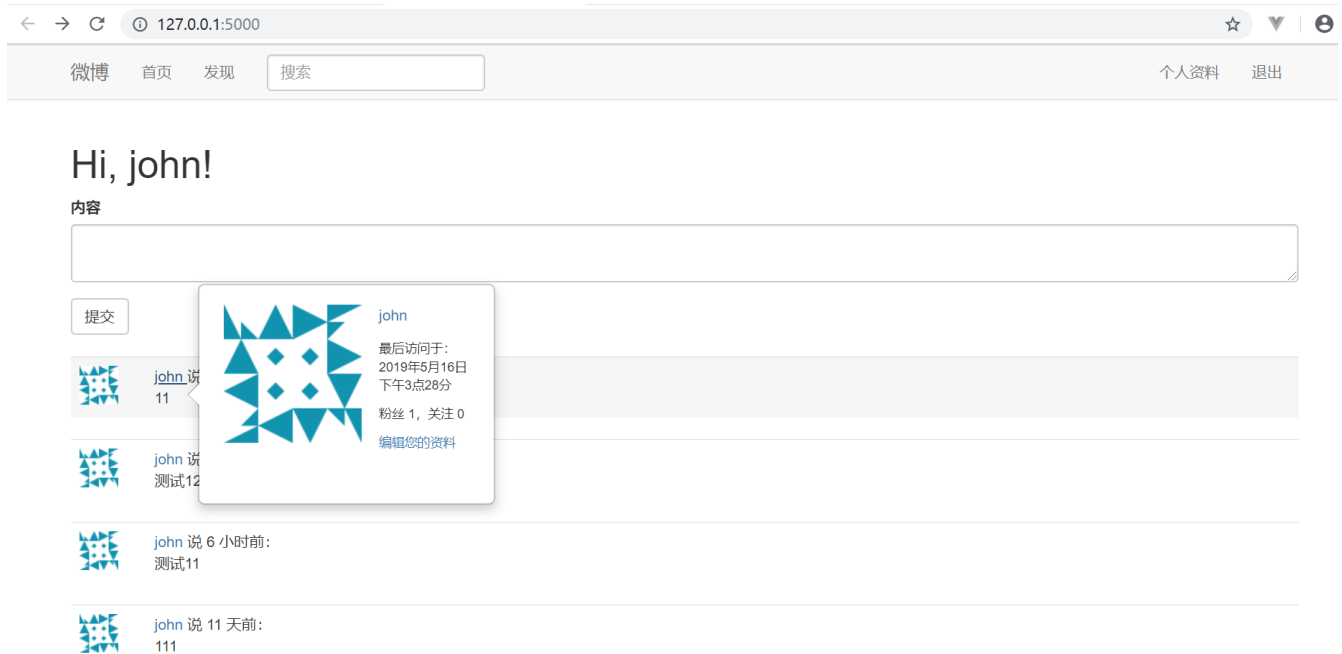
```

```

28         }, 1000);
29     },
30     function(event) {
31         const elem = $(event.currentTarget);
32         if (timer) {
33             clearTimeout(timer);
34             timer=null;
35         } else if (xhr) {
36             xhr.abort();
37             xhr = null;
38         } else {
39             elem.popover('destroy');
40         }
41     }
42 )
43 })

```

1.5. 启动服务测试效果



2. 发送私信功能

2.1. 新增私信数据库模型

修改 app/models.py 脚本，新增私信数据模型。

```

1 class Message(db.Model):
2     """私信"""
3     __tablename__ = 'messages'
4     id = db.Column(db.Integer, primary_key=True)
5     sender_id = db.Column(db.Integer, db.ForeignKey('users.id'))
6     recipient_id = db.Column(db.Integer, db.ForeignKey('users.id'))
7     body = db.Column(db.String(140))
8     timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow())
9
10    def __repr__(self):
11        return '<Message {}>'.format(self.body)

```

修改 `User` 模型，增加对私信的反向查询支持，并增加用户最后阅读信息时间以及根据最后阅读时间统计的未读信息条数。

```

1 class User(UserMixin, db.Model):
2     # .....
3     messages_sent = db.relationship('Message', foreign_keys='Message.sender_id',
4                                     backref='author', lazy='dynamic')
5     messages_received = db.relationship('Message',
6                                         foreign_keys='Message.recipient_id',
7                                         backref='recipient', lazy='dynamic')
8     last_message_read_time = db.Column(db.DateTime)
9
10    def new_messages(self):
11        """查询用户未读信息条数"""
12        last_read_time = self.last_message_read_time or datetime(1900, 1, 1)
13        return Message.query.filter_by(recipient=self).filter(Message.timestamp >
14                                last_read_time).count()
15    # .....

```

进行数据库迁移

```

1 (venv) D:\Projects\learn\flask-mega-tutorial>flask db migrate -m "私信"
2 [2019-05-16 15:49:33,108] INFO in logger: 博客已启动
3 INFO [alembic.runtime.migration] Context impl SQLiteImpl.
4 INFO [alembic.runtime.migration] Will assume non-transactional DDL.
5 INFO [alembic.autogenerate.compare] Detected added table 'messages'
6 INFO [alembic.autogenerate.compare] Detected added index 'ix_messages_timestamp' on
7 ['timestamp']
8 INFO [alembic.autogenerate.compare] Detected added column
9 'users.last_message_read_time'
10 Generating D:\Projects\learn\flask-mega-tutorial\migrations\versions\73b28d6c195e_私
11 信.py ... done
12
13 (venv) D:\Projects\learn\flask-mega-tutorial>flask db upgrade
14 [2019-05-16 15:49:42,810] INFO in logger: 博客已启动
15 INFO [alembic.runtime.migration] Context impl SQLiteImpl.
16 INFO [alembic.runtime.migration] Will assume non-transactional DDL.
17 INFO [alembic.runtime.migration] Running upgrade 251e4854ef4c -> 73b28d6c195e, 私信

```

2.2. 编写发送私信表单类

修改 `app/main/forms.py` 脚本，新增发送私信表单类。

```
1 class MessageForm(FlaskForm):
2     """私信表单"""
3     message = TextAreaField(_l('内容'), validators=[DataRequired(), Length(min=0,
4     max=140)])
5     submit = SubmitField(_l('提交'))
```

2.3. 新增发送私信模板

新增 `app/templates/send_message.html` 文件，用于编写发送私信页面。

```
1 {% extends 'base.html' %}
2 {% import 'bootstrap/wtf.html' as wtf %}
3
4 {% block app_content %}
5     <h1>{{ _l('向%(recipient)s发送信息', recipient=recipient) }}</h1>
6     <div class="row">
7         <div class="col-md-4">
8             {{ wtf.quick_form(form) }}
9         </div>
10    </div>
11 {% endblock %}
```

2.4. 编写发送私信视图类

修改 `app/main/views.py` 脚本，新增发送私信视图类。

```
1 class SendMessageView(View):
2     """发送私信视图"""
3     methods = ['GET', 'POST']
4     decorators = [login_required]
5
6     def dispatch_request(self, recipient):
7         user = User.query.filter_by(username=recipient).first_or_404()
8         form = MessageForm()
9         if form.validate_on_submit():
10             msg = Message(author=current_user, recipient=user, body=form.message.data)
11             db.session.add(msg)
12             db.session.commit()
13             flash(_l('您的消息已被发送'))
14             return redirect(url_for('main.user_info', username=recipient))
15         return render_template('send_message.html', title=_l('发送消息'), form=form,
16                                recipient=recipient)
```

修改 `app/main/__init__.py` 脚本，注册视图。

```

1 def register_views(bp):
2     # 在函数中引入可以避免循环依赖问题
3     from app.main.views import IndexView, ExploreView, UserInfoView, UserInfoEditView,
FollowView, \
4         UnfollowView, SearchView, UserPopupView, SendMessageView
5     # .....
6     # 发送私信视图
7     bp.add_url_rule('/send_message/<recipient>',
view_func=SendMessageView.as_view('send_message'))

```

2.5. 个人主页增加发送私信链接

修改 `app/templates/user_info.html` 文件，增加发送私信链接。

```

1         {% if user != current_user %}
2             <p><a href="{ { url_for('main.send_message',
recipient=user.username) } }">{ { _('发送私信') } }</a></p>
3         {% endif %}
4     </td>
5 </tr>

```

2.6. 启动服务测试效果

通过susan用户登录，查看john的个人资料信息。

[微博](#)
[首页](#)
[发现](#)

[个人资料](#)
[退出](#)



用户: john

最后访问于:
2019年5月16日下午4点20分

粉丝 1, 关注 0

[取消关注](#)
[发送私信](#)



john 说 6 小时前:

11



john 说 6 小时前:

测试12



john 说 6 小时前:

测试11

点击发送私信链接，跳转至发送私信页面。

向john发送信息

内容

提交

发送私信后，跳转回john个人资料页面。

您的消息已被发送



用户: john

最后访问于:

2019年5月16日下午4点20分

粉丝 1, 关注 0

[取消关注](#)[发送私信](#)john 说 6 小时前:
11john 说 6 小时前:
测试12john 说 6 小时前:
测试12

flask-mega-tutorial [D:\Projects\learn\flask-mega-tutorial] - main.messages [flask] - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

Database flask schemas main messages

Project main

- __init__.py
- forms.py
- views.py

static

- templates
- auth
- email
- errors
- _post.html
- base.html
- index.html
- search.html
- send_message.html
- user_info.html
- user_info_edit.html
- user_popup.html

translations

- __init__.py
- cli.py
- config.py
- email.py
- logger.py
- models.py
- search.py

models.py forms.py send_message.html views.py __init__.py user_info.html main.messages [flask]

<Filter criteria>

| id | sender_id | recipient_id | body | timestamp |
|----|-----------|--------------|------|--------------------------|
| 1 | 1 | 2 | 私信测试 | 2019-05-16 08:20:02.7... |

1 row

Tab-se...d (TSV) DDL View Query

1 row retrieved starting from 1 in 79 ms (execution: 11 ms, fetching: 68 ms)

Git: master Stat: 11 01

3. 查看私信功能

3.1. 编写查看私信视图类

修改 `app/main/views.py` 脚本，增加查看私信视图类。

```
1 class MessageView(View):
2     """私信视图"""
3     methods = ['GET']
4     decorators = [login_required]
5
6     def dispatch_request(self):
7         # 更新私信最后查看时间
8         current_user.last_message_read_time = datetime.utcnow()
9         db.session.commit()
10        page = request.args.get('page', 1, type=int)
11        messages = current_user.messages_received.order_by(
12            Message.timestamp.desc()).paginate(
13            page, current_app.config['POSTS_PER_PAGE'], False)
14        next_url = url_for('main.messages', page=messages.next_num) if
messages.has_next else None
15        prev_url = url_for('main.messages', page=messages.prev_num) if
messages.has_prev else None
16        return render_template('messages.html', messages=messages.items,
next_url=next_url, prev_url=prev_url, page=page)
```

修改 `app/main/__init__.py` 脚本，注册视图。

```
1 def register_views(bp):
2     # 在函数中引入可以避免循环依赖问题
3     from app.main.views import IndexView, ExploreView, UserInfoView, UserInfoEditView,
FollowView, \
4         UnfollowView, SearchView, UserPopupView, SendMessageView, MessageView
5     .....
6     # 查看私信视图
7     bp.add_url_rule('/messages', view_func=MessageView.as_view('messages'))
```

3.2. 新增查看私信模板

新增 `app/templates/messages.html` 文件，编写展示私信页面。

```
1 {% extends 'base.html' %}
2
3 {% block app_content %}
4     <h1>{{ _('私信') }}</h1>
5     {% for post in messages %}
6         {% include '_post.html' %}
7     {% endfor %}
8     <nav aria-label="...">
9         <ul class="pager">
```

```

10         <li class="previous{% if not prev_url %} disabled {% endif %}">
11             <a href="{% prev_url or '#' %}">
12                 <span aria-hidden="true">&larr;</span> {{ _('上一页') }}
13             </a>
14         </li>
15         <li>{{ _('第%(page)s页', page=page) }}</li>
16         <li class="next{% if not next_url %} disabled {% endif %}">
17             <a href="{% next_url or '#' %}">
18                 {{ _('下一页') }} <span aria-hidden="true">&rarr;</span>
19             </a>
20         </li>
21     </ul>
22 </nav>
23 {% endblock %}

```

3.3. 导航栏增加查看私信链接

修改 `app/templates/base.html` 文件，增加查看私信链接。

```

1     .....
2         <ul class="nav navbar-nav navbar-right">
3             {% if current_user.is_anonymous %}
4                 <li><a href="{% url_for('auth.login') %}">{{ _('登录') }}</a>
5             </li>
6             {% else %}
7                 <li><a href="{% url_for('main.messages') %}">{{ _('私信') }}</a>
8             </li>
9         </ul>
10     .....

```

3.4. 更新翻译文件

作为所有更改的一部分，还有一些新的文本被添加到几个位置，并且需要将这些文本合并到语言翻译中。第一步是更新所有的语言目录：

```
1 | (venv) D:\Projects\learn\flask-mega-tutorial>flask translate update
```

然后，`app/translations` 中的每种语言都需要使用新翻译更新其 `messages.po` 文件，并进行编译。

```
1 | (venv) D:\Projects\learn\flask-mega-tutorial>flask translate compile
```

3.5. 启动服务测试效果

通过用户 `john` 登录，首页增加私信链接。

Hi, john!

内容

提交



john 说 7 小时前:
11



john 说 7 小时前:
测试12



john 说 7 小时前:
测试11

点击私信链接，查看私信信息。

私信



susan 说 42 分钟前:
私信测试

← 上一页

第1页

下一页 →

4. 消息通知徽章功能

4.1. 静态消息通知徽章

现在私有消息功能已经实现，但是还没有通过任何渠道告诉用户有私有消息等待阅读。导航栏上的未读消息标志的最简单实现可以使用Bootstrap badge小部件渲染到基础模板中。

修改 `app/templates/base.html` 文件，在导航栏中增加静态消息通知徽章。通过 `current_user.new_messages()` 函数获取用户当前未读信息条数。

```
1 .....
2
3         <ul class="nav navbar-nav navbar-right">
4             {% if current_user.is_anonymous %}
5                 <li><a href="{{ url_for('auth.login') }}">{{ _('登录') }}</a>
6             </li>
7             {% else %}
8                 <li>
9                     <a href="{{ url_for('main.messages') }}">
10                        {{ _('私信') }}
11                        {% set new_messages = current_user.new_messages() %}
12                        {% if new_messages %}
```

```

11         <span class="badge">{{ new_messages }}</span>
12         {% endif %}
13     </a>
14 </li>
15 .....

```

4.2. 动态消息通知徽章模板修改

对于静态消息通知徽章仍存在问题，即使未读消息数量不能自动刷新，需要点击链接并刷新页面后才能展示最新的未读消息。

目前当加载页面消息计数为非零时，徽章才在页面中渲染。为了实现动态消息徽章，首先要修改 `app/templates/base.html` 文件，始终保持在导航栏中包含徽章，并在消息计数为零时将其标记为隐藏。为表示徽章的元素添加了一个 `id` 属性，以便使用 `$('#message_count')` jQuery 选择器来简化这个元素的选取。

```

1 .....
2         <ul class="nav navbar-nav navbar-right">
3             {% if current_user.is_anonymous %}
4                 <li><a href="{{ url_for('auth.login') }}">{{ _('登录') }}</a>
5             </li>
6             {% else %}
7                 <li>
8                     <a href="{{ url_for('main.messages') }}">
9                         {{ _('私信') }}
10                        {% set new_messages = current_user.new_messages() %}
11                        <span id="message_count" class="badge"
12                            style="visibility: {% if new_messages %}visible
13                                {% else %}hidden {% endif %}";>
14                            {{ new_messages }}
15                        </span>
16                    </a>
17                </li>
18            .....

```

还是修改 `app/templates/base.html` 文件，编写一个JavaScript函数，将该徽章更新为最新的数字：

```

1 .....
2     <script>
3         .....
4         function set_message_count(n) {
5             $('#message_count').text(n);
6             $('#message_count').css('visibility', n ? 'visible': 'hidden');
7         }
8     </script>
9 .....

```

4.3. 新增通知模型

现在还需要做的，就是客户端可以定期接收有关用户拥有的未读消息数量的更新。当更新发生时，客户端将调用 `set_message_count()` 函数来使用户知道更新，目前存在两种处理方式：

- **客户端定期请求服务器**：客户端通过发送异步请求定期向服务器请求更新。根据请求的响应信息来更新页面的不同元素，例如未读消息计数标记，此方式的缺点在于不能实时获取最新信息；
- **服务端自动推送客户端**：需要客户端和服务器之间的特殊连接类型，以允许服务器自由地将数据推送到客户端，此方式的缺点在于消耗资源较大。

目前先通过第一种方式来处理，首先，要修改 `app/models.py` 脚本，添加一个新模型来跟踪所有用户的通知，以及用户模型中的关系。其中通知模型中 `payload_json` 字段用于存通知信息，每种类型的通知都会有所不同，所以将它写为JSON字符串，因为这样可以编写列表，字典或单个值（如数字或字符串）。为了方便，添加 `get_data()` 方法，以便调用者不必操心JSON的反序列化。

```

1 class User(UserMixin, db.Model):
2     # .....
3     notifications = db.relationship('Notification', backref='user', lazy='dynamic')
4     # .....
5
6 class Notification(db.Model):
7     """通知"""
8     id = db.Column(db.Integer, primary_key=True)
9     name = db.Column(db.String(128), index=True)
10    user_id = db.Column(db.Integer, db.ForeignKey('users.id'))
11    timestamp = db.Column(db.Float, index=True, default=time)
12    payload_json = db.Column(db.Text)
13
14    def get_data(self):
15        return json.loads(str(self.payload_json))

```

数据库迁移

```

1 (venv) D:\Projects\learn\flask-mega-tutorial>flask db migrate -m "通知"
2 [2019-05-16 17:57:47,995] INFO in logger: 博客已启动
3 INFO [alembic.runtime.migration] Context impl SQLiteImpl.
4 INFO [alembic.runtime.migration] Will assume non-transactional DDL.
5 INFO [alembic.autogenerate.compare] Detected added table 'notification'
6 INFO [alembic.autogenerate.compare] Detected added index 'ix_notification_name' on
7 ['name']
8 INFO [alembic.autogenerate.compare] Detected added index 'ix_notification_timestamp'
9 on ['timestamp']
10 Generating D:\Projects\learn\flask-mega-tutorial\migrations\versions\d8b0c0657b94_通知
11 知.py ... done
12
13 (venv) D:\Projects\learn\flask-mega-tutorial>flask db upgrade
14 [2019-05-16 17:57:54,403] INFO in logger: 博客已启动
15 INFO [alembic.runtime.migration] Context impl SQLiteImpl.
16 INFO [alembic.runtime.migration] Will assume non-transactional DDL.
17 INFO [alembic.runtime.migration] Running upgrade 73b28d6c195e -> d8b0c0657b94, 通知

```

修改 `microblog.py` 脚本，将新增的 `Message` 和 `Notification` 模型添加到shell上下文，这样就可以直接在用 `flask shell` 命令启动的解释器中使用这两个模型了。

```

1 from app import create_app, db
2 from app.models import User, Post, Message, Notification
3
4 app = create_app()
5
6 @app.shell_context_processor
7 def make_shell_context():
8     return {'db': db, 'User': User, 'Post': Post, 'Message': Message, 'Notification':
    Notification}

```

4.4. 更新用户通知

在用户模型中添加一个 `add_notification()` 辅助方法，以便更轻松地处理这些对象。此方法不仅为用户添加通知给数据库，还确保如果具有相同名称的通知已存在，则会首先删除该通知。将要使用的通知将被称为 `unread_message_count`。如果数据库已经有一个带有这个名称的通知，例如值为3，则当用户收到新消息并且消息计数变为4时，我就会替换旧的通知。

```

1 class User(UserMixin, db.Model):
2     # .....
3     def add_notification(self, name, data):
4         """新增通知"""
5         self.notifications.filter_by(name=name).delete()
6         n = Notification(name=name, payload_json=json.dumps(data), user=self)
7         db.session.add(n)
8         db.session.commit()
9         return n

```

修改 `app/main/views.py` 脚本，修改发送私信视图类，当用户接收到新的私信时，更新用户通知信息。

```

1 class SendMessageView(View):
2     # .....
3     def dispatch_request(self, recipient):
4         # .....
5         if form.validate_on_submit():
6             # .....
7             user.add_notification('unread_message_count', user.new_messages())
8             db.session.commit()

```

修改 `app/main/views.py` 脚本，修改查看私信视图函数，当用户跳转到查看私信页面时，未读计数需要清零。

```

1 class MessageView(View):
2     # .....
3     def dispatch_request(self):
4         # 更新私信最后查看时间
5         current_user.last_message_read_time = datetime.utcnow()
6         current_user.add_notification('unread_message_count', 0)
7         db.session.commit()

```

4.5. 编写通知视图类

修改 `app/main/views.py` 脚本，增加通知视图类，用于提供客户端对于当前登录用户的通知检索功能。

```
1 class NotificationView(View):
2     """通知"""
3     methods = ['GET']
4     decorators = [login_required]
5
6     def dispatch_request(self):
7         # 实现通过查询条件，来限制只请求给定时间戳之后产生的通知
8         since = request.args.get('since', 0.0, type=float)
9         notifications = current_user.notifications.filter(
10             Notification.timestamp > since).order_by(Notification.timestamp.asc())
11         return jsonify([{'name': n.name, 'data': n.get_data(), 'timestamp':
n.timestamp} for n in notifications])
```

修改 `app/main/__init__.py` 脚本，增加视图注册。

```
1 def register_views(bp):
2     # 在函数中引入可以避免循环依赖问题
3     from app.main.views import IndexView, ExploreView, UserInfoView, UserInfoEditView,
FollowView, \
4         UnfollowView, SearchView, UserPopupView, SendMessageView, MessageView,
NotificationView
5     # .....
6     # 查询通知视图
7     bp.add_url_rule('/notifications',
view_func=NotificationView.as_view('notifications'))
```

4.6. 客户端轮询通知

修改 `app/templates/base.html` 脚本，增加调用后台服务查询通知信息并实时更新页面的功能。

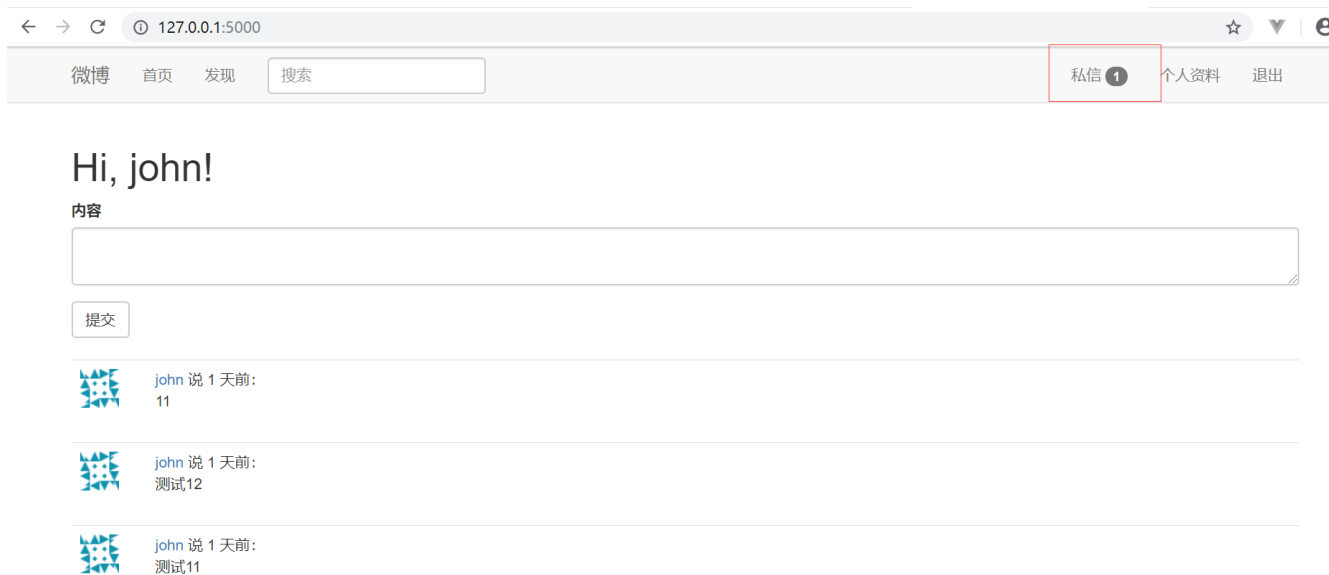
```
1 <script>
2     .....
3     {% if current_user.is_authenticated %}
4     $(function () {
5         let since = 0;
6         // 定期调用回调函数，每10秒轮询一次
7         setInterval(function () {
8             $.ajax('{{ url_for('main.notifications') }}?since='+since).done(
9                 function (notifications) {
10                     for (let i = 0; i < notifications.length; i++) {
11                         if (notifications[i].name === 'unread_message_count') {
12                             // 根据名为unread_message_count的通知，更新消息计数徽标
13                             set_message_count(notifications[i].data);
14                         }
15                         // 根据最后获取时间，限制只处理新的通知信息
16                         since = notifications[i].timestamp;
17                     }
18                 }
19             );
```



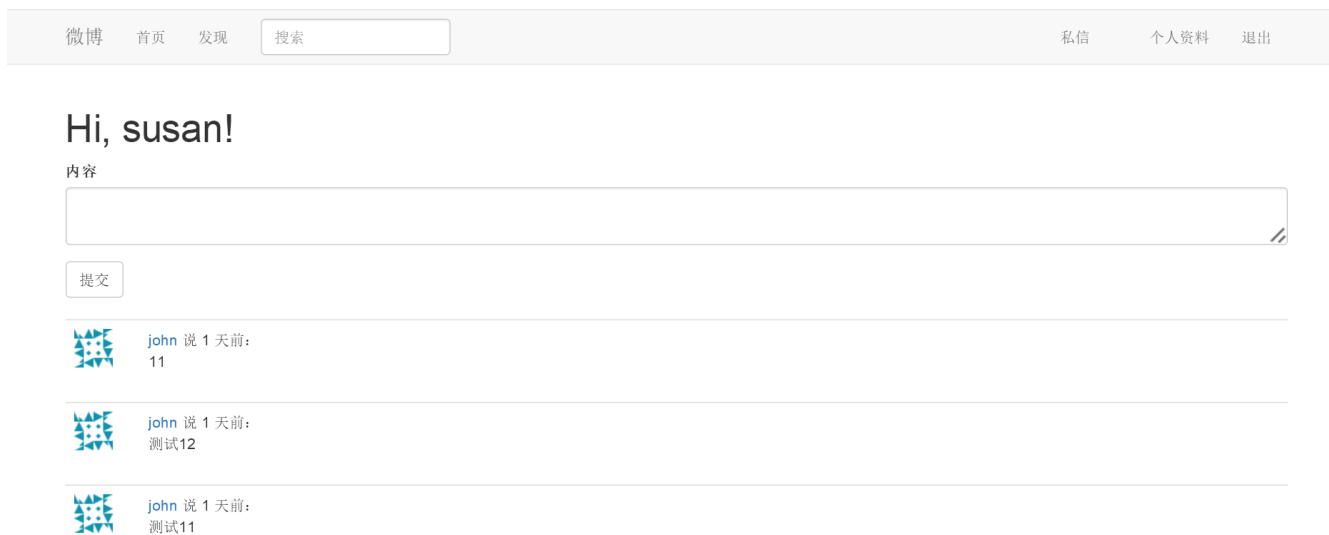
```
20         }, 10000);
21     });
22     {% endif %}
23 </script>
```

4.7. 启动服务测试效果

客户john登录，存在一条未读私信。



用另一个浏览器，通过客户susan登录



通过客户susan向john发送私信。

向john发送信息

内容

测试1111




提交

john首页自动更新未读私信条数为2。

Hi, john!





内容

提交

- john 说 1 天前:
11
- john 说 1 天前:
测试12
- john 说 1 天前:
测试11

点击私信进入查看具体信息，同时私信徽标清零。

私信

- susan 说 4 分钟前:
测试1111
- susan 说 4 分钟前:
测试测试
- susan 说 19 分钟前:
测试
- susan 说 10 分钟前: