

邮件重置密码实现

邮件重置密码实现

1. Flask-Mail扩展库使用
 - 1.1. 扩展库安装
 - 1.2. 扩展库注册
 - 1.3. 编写发送邮件公共函数
2. 请求重置密码实现
 - 2.1. 登录模板增加请求重置密码链接
 - 2.2. 编写申请重置密码表单
 - 2.3. 编写申请重置密码模板
 - 2.4. 编写申请重置密码视图
 - 2.5. 编写JWT令牌生成及验证方法
 - 2.6. 编写发送密码重置邮件函数
 - 2.7. 编写申请密码修改邮件模板
 - 2.8. 增加申请重置密码视图发送邮件函数引用
 - 2.9. 注册申请重置密码视图
3. 重置密码实现
 - 3.1. 编写重置密码表单
 - 3.2. 编写重置密码模板
 - 3.3. 编写重置密码视图
 - 3.4. 注册重置密码视图
4. 异步发送邮件实现
5. 启动服务测试

1. Flask-Mail扩展库使用

1.1. 扩展库安装

对于实际的邮件发送而言，Flask有一个名为[Flask-Mail](#)的流行插件，可以使任务变得非常简单。同时，密码重置链接将包含有一个安全令牌。为了生成这些令牌，我将使用[JSON Web Tokens](#)，它也有一个流行的Python包 `pyjwt`。

```
pip install flask-mail
pip install pyjwt
```

1.2. 扩展库注册

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

如果希望真实地发送电子邮件，则需要使用真实的电子邮件服务器，只需要设置 `MAIL_SERVER`、`MAIL_PORT`、`MAIL_USE_TLS`、`MAIL_USERNAME` 和 `MAIL_PASSWORD` 等环境变量。

```

.....
from flask_mail import Mail
.....
# 实例化flask_mail
mail = Mail()

def create_app(test_config=None):
    .....
    # 初始化flask_mail
    mail.init_app(application)

```

1.3. 编写发送邮件公共函数

新建 `app/email.py` 脚本，编写发送邮件的公共函数。

```

from flask_mail import Message

from app import mail

def send_email(subject, sender, recipients, text_body, html_body):
    """
    发送电子邮件
    :param subject: 标题
    :param sender: 发送者
    :param recipients: 接收者列表
    :param text_body: 纯文本内容
    :param html_body: HTML格式内容
    :return:
    """
    msg = Message(subject, sender=sender, recipients=recipients)
    msg.body = text_body
    msg.html = html_body
    mail.send(msg)

```

2. 请求重置密码实现

2.1. 登录模板增加请求重置密码链接

修改 `app/templates/login/login.html` 文件，增减请求重置密码链接。

```

.....
<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>
</form>
{% endblock %}

```

2.2. 编写申请重置密码表单

修改 `app/forms.py` 脚本，新增申请重置密码表单。

```
class ResetPasswordRequestForm(FlaskForm):
    """重置密码请求表单"""
    email = StringField('邮箱', validators=[DataRequired(), Email()])
    submit = SubmitField('请求密码重置')
```

2.3. 编写申请重置密码模板

新建 `app/templates/login/reset_password_request.html` 文件，编写申请重置密码模板。

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

{% block content %}
    <h1>重置密码</h1>
    <form action="" method="post">
        {{ form.hidden_tag() }}
        <p>
            {{ form.email.label }}<br>
            {{ form.email(size=64) }}<br>
            {% for error in form.email.errors %}
                <span style="color: red;">{{ error }}</span>
            {% endfor %}
        </p>
        <p>{{ form.submit() }}</p>
    </form>
{% endblock %}
```

2.4. 编写申请重置密码视图

修改 `app/login.py` 脚本，新增申请重置密码视图函数。其中，发送重置密码申请邮件的函数 `send_password_reset_email()` 会在下面有具体说明。

```
class ResetPasswordRequestView(View):
    """重置密码申请视图"""
    methods = ['GET', 'POST']

    def dispatch_request(self):
        if current_user.is_authenticated:
            return redirect(url_for('index'))
        form = ResetPasswordRequestForm()
        if form.validate_on_submit():
            user = User.query.filter_by(email=form.email.data).first()
            if not user:
                flash('该电子邮箱未注册')
                return redirect(url_for('reset_password_request'))

            send_password_reset_email(user)
            flash('查看您的电子邮箱消息，以重置您的密码')
            return redirect(url_for('login'))
```

```
return render_template('login/reset_password_request.html', title='重置密码',
form=form)
```

2.5. 编写JWT令牌生成及验证方法

修改 `app/models.py` 脚本中的 `User` 模型，增加用户 JWT 令牌生成及验证方法。

用于密码重置令牌的有效载荷格式为 `{'reset_password': user_id, 'exp': token_expiration}`。`exp` 字段是 JWTs 的标准，如果它存在，则表示令牌的到期时间。如果一个令牌有一个有效的签名，但是它已经过期，那么它也将被认为是无效的。

当用户点击电子邮件链接时，令牌将被作为 URL 的一部分发送回应用，处理这个 URL 的视图函数首先就是验证它。如果签名是有效的，则可以通过存储在有效载荷中的 ID 来识别用户。一旦得知用户的身份，该用户即可进行密码修改。

```
import jwt

class User(UserMixin, db.Model):
    .....
    def get_jwt_token(self, expires_in=600):
        """获取JWT令牌"""
        return jwt.encode({'reset_password': self.id, 'exp': time() + expires_in},
                           current_app.config['SECRET_KEY'],
                           algorithm='HS256').decode('utf8')

    @staticmethod
    def verify_jwt_token(token):
        try:
            user_id = jwt.decode(token,
                                  current_app.config['SECRET_KEY'],
                                  algorithms='HS256')['reset_password']

        except Exception as e:
            print(e)
            return
        return User.query.get(user_id)

    def __repr__(self):
        """打印类对象时的展示方式"""
        return '<User %r>' % self.username
```

2.6. 编写发送密码重置邮件函数

修改 `app/email.py` 脚本，增加 `send_password_reset_email()` 函数，用于发送密码重置电子邮件。其中发送者 `sender` 参数取值自 `config.py` 配置文件中配置的参数。

```
def send_password_reset_email(user):
    """发送密码重置电子邮件"""
    token = user.get_jwt_token()
    send_email('[博客] 重置您的密码',
               sender=current_app.config['MAIL_USERNAME'],
               recipients=[user.email],
               text_body=render_template('email/reset_password.txt', user=user,
                                         token=token),
               html_body=render_template('email/reset_password.html', user=user,
                                         token=token))
```

2.7. 编写申请密码修改邮件模板

通过 `render_template()` 函数从模板生成申请密码修改邮件内容。模板接收用户和令牌作为参数，以便可以生成个性化的电子邮件消息。其中，`url_for()` 函数中的 `_external=True` 参数设置为 `True`，就会生成一个URL的完全路径。

新增纯文本格式模板 `app/templates/email/reset_password.txt`：

```
亲爱的 {{ user.username }}，

请点击下面的链接来重置您的密码：

{{ url_for('reset_password', token=token, _external=True) }}

如果您未申请密码重置请忽略此信息。

谨致问候，

博客团队
```

新增HTML格式模板 `app/templates/email/reset_password.html`：

```
<p>亲爱的 {{ user.username }}，</p>
<p>
    重置您的密码
    <a href="{{ url_for('reset_password', token=token, _external=True) }}">
        点击此处
    </a>。
</p>
<p>另外，你也可以在浏览器的地址栏中粘贴以下链接:</p>
<p>{{ url_for('reset_password', token=token, _external=True) }}</p>
<p>如果您未申请密码重置请忽略此信息。</p>
<p>谨致问候，</p>
<p>博客团队</p>
```

2.8. 增加申请重置密码视图发送邮件函数引用

修改 `app/login.py` 脚本，增加发送邮件函数 `send_password_reset_email()` 的引入。

```
class ResetPasswordRequestView(view):
    .....
    from app.email import send_password_reset_email
    send_password_reset_email(user)
    flash('查看您的电子邮箱消息，以重置您的密码')
    return redirect(url_for('login'))
    return render_template('login/reset_password_request.html', title='重置密码',
form=form)
```

2.9. 注册申请重置密码视图

修改 `app/__init__.py` 脚本，增加申请重置密码视图函数的注册。

```
def create_app(test_config=None):
    .....
    # 注册申请重置密码视图URL
    from app.login import ResetPasswordRequestView
    application.add_url_rule('/reset_password_request',

view_func=ResetPasswordRequestView.as_view('reset_password_request'))
```

3. 重置密码实现

3.1. 编写重置密码表单

修改 `app/forms.py` 脚本，编写重置密码表单。

```
class ResetPasswordForm(FlaskForm):
    """重置密码表单"""
    password = PasswordField('密码', validators=[DataRequired()])
    password2 = PasswordField('确认密码', validators=[DataRequired(), EqualTo('password')])
    submit = SubmitField('请求密码重置')
```

3.2. 编写重置密码模板

新增 `app/templates/login/reset_password.html` 文件，编写重置密码模板。

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

{% block content %}
<h1>重置您的密码</h1>
<form action="" method="post">
    {{ form.hidden_tag() }}
    <p>
        {{ form.password.label }}<br>
        {{ form.password(size=32) }}<br>
        {% for error in form.password.errors %}
            <span style="color: red;">{{ error }}</span>
        {% endfor %}
    </p>
</form>
```

```

</p>
<p>
    {{ form.password2.label }}<br>
    {{ form.password2(size=32) }}<br>
    {% for error in form.password2.errors %}
        <span style="color: red;">{{ error }}</span>
    {% endfor %}
</p>
<p>{{ form.submit() }}</p>
</form>
{% endblock %}

```

3.3. 编写重置密码视图

修改 `app/login.py` 脚本，编写重置密码视图函数。

```

class ResetPasswordView(View):
    """重置密码视图"""
    methods = ['GET', 'POST']

    def dispatch_request(self, token):
        if current_user.is_authenticated:
            return redirect(url_for('index'))
        user = User.verify_jwt_token(token)
        if not user:
            return redirect(url_for('index'))
        form = ResetPasswordForm()
        if form.validate_on_submit():
            user.set_password(form.password.data)
            db.session.commit()
            flash('您的密码已被重置')
            return redirect(url_for('login'))
        return render_template('login/reset_password.html', form=form)

```

3.4. 注册重置密码视图

修改 `app/__init__.py` 脚本，注册重置密码视图函数。

```

def create_app(test_config=None):
    .....
    # 注册重置密码视图URL
    from app.login import ResetPasswordView
    application.add_url_rule('/reset_password/<token>',
                             view_func=ResetPasswordView.as_view('reset_password'))

```

4. 异步发送邮件实现

修改 `app/email.py` 脚本，将发送邮件函数 `send_email()` 修改为异步发送方式。由于发送电子邮件可能会大大减慢应用的速度，所以在发送电子邮件启动一个后台线程异步处理。

```
from threading import Thread

from flask_mail import Message

from app import create_app, mail


def send_async_email(msg):
    app = create_app()
    app.app_context().push()
    with app.app_context():
        mail.send(msg)


def send_email(subject, sender, recipients, text_body, html_body):
    """
    发送电子邮件
    :param subject: 标题
    :param sender: 发送者
    :param recipients: 接收者列表
    :param text_body: 纯文本内容
    :param html_body: HTML格式内容
    :return:
    """
    msg = Message(subject, sender=sender, recipients=recipients)
    msg.body = text_body
    msg.html = html_body
    Thread(target=send_async_email, args=(msg,)).start()
```

5. 启动服务测试

点击登录页面[重置密码](#)链接，跳转至重置密码页面

← → ↺ ⓘ 127.0.0.1:5000/reset_password_request

博客: [首页](#) [发现](#) [登录](#)

重置密码

邮箱

请求密码重置

录入用户已注册邮箱，点击[请求密码重置](#)按钮，跳转回登录页面，异步发送邮件

博客: [首页](#) [发现](#) [登录](#)

- 查看您的电子邮箱消息，以重置您的密码

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](#)


注册邮箱接收到密码重置邮件





【微博】重置您的密码 ☆

发件人: <[redacted]@[redacted].com> 

时间: 2019年5月6日(星期一) 下午2:29

收件人: [redacted] >



纯文本 |    

亲爱的 john,

重置您的密码 [点击此处](#)。

另外，你也可以在浏览器的地址栏中粘贴以下链接:

http://127.0.0.1:5000/reset_password/eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJyZXNldF9wYXNzd29yZCI6MSwiZXhwIjoxNTU3MTI0NzY1MjE5MzE2MzQ0mLCWWNADtOhUAWck7PfN9nTXhQXU3v1AmxHTc

如果您未申请密码重置请忽略此信息。

谨致问候,

微博团队

点击[点击此处](#)链接，跳转至密码重置页面

重置您的密码

密码

确认密码

请求密码重置

录入新密码，提交表单后，跳转至登录页面

- 您的密码已被重置

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](#)

根据修改后的密码登录

博客: [首页](#) [发现](#) [个人资料](#) [退出](#)

Hi, john!

内容

提交



[john](#) says:
111

第1页