Flask-Mail

web应用程序中最基本的功能之一是向用户发送电子邮件,`Flask-mail`扩展提供了一个简单的接口, 可以为Flask应用程序设置SMTP, 并从视图和脚本发送消息

安装

通过pip安装

```
pip install Flask-Mail
```

通过easy_install安装

```
easy_install install Flask-Mail
```

或者通过git获取最新版本, 通过下面的方式安装

```
git clone https://github.com/mattupstate/flask-mail.git
cd flask-mail
python setup.py install
```

如果你是使用`virtualenv`搭建的虚拟环境, 假设你为你所有的`Flask`应用程序都在同一个`virtualenv`环境中安装了`Flask-Mail`

配置Flask-Mail

`Flask-Mail`通过标准的`Flask`配置`API`来进行配置, 下面是可用的选项:

- MAIL_SERVER
  说明:邮件服务器
  默认值:'localhost'

- MAIL_PORT
  说明:端口号
  默认值:25
- MAIL_USE_TLS
  说明:是否使用LTS
  默认值:False
- MAIL_USE_SSL
  说明:是否使用SSL
  默认值:False
- MAIL_DEBUG
  说明:邮件调试对象
  默认值:app.debug
- MAIL_USERNAME
  说明:用户名
  默认值:None
- MAIL_PASSWORD
  说明:密码
  默认值:None
- MAIL_DEFAULT_SENDER
  说明:邮件默认发件人
  默认值:None
- MAIL_MAX_EMAILS
  说明:单次最大发送数量
  默认值:None
- MAIL_SUPPRESS_SEND
  说明:用来设置是否发送邮件
  默认值:app.testing
- MAIL_ASCII_ATTACHMENTS
  说明:ASCII格式的文件名
  默认值:False

此外, 标准Flask测试配置选项在单元测试中用于Flask- mail, 如下:

```
from flask import Flask
from flask_mail import Mail

app = Flask(__name__)
mail = Mail(app)
```

在本例中, 所有电子邮件都是使用传递给`Mail`类构造函数的应用程序的配置值发送的你也可以稍后在配置时设置你的邮件实例, 使用`init_app`方法

```
mail = Mail()

app = Flask(__name__)
mail.init_app(app)
```

在这种情况下, 电子邮件将使用从`Flask`的`current_app`的上下文中的配置来发送邮件. 如果您有多个应用程序在相同的进程中运行, 但是配置选项不同, 那么这是非常有用的.

发送消息

要发送消息, 首先创建一个消息实例:

```
from flask_mail import Message

@app.route("/")
def index():
    msg = Message("Hello", sender="from@example.com", recipients=["to@example.com"])
```

您可以立即设置收件人的电子邮件, 或单独设置:

```
msg.recipients = ["you@example.com"]
msg.add_recipient("somebodyelse@example.com")
```

如果你已经设置了`MAIL_DEFAULT_SENDER`, 你不需要再设置`message sender`, 因为它将默认使用这个配置值

```
msg = Message("Hello", recipients=["to@example.com"])
```

如果发件人是一个二元元组, 这将分为名称和地址

```
msg = Message("Hello", sender=("Me", "me@example.com"))

assert msg.sender == "Me <me@example.com>"
```

消息可以包含正文和/或HTML

```
msg.body = "testing"
msg.html = "<b>testing</b>"
```

最后, 要发送消息, 您使用与您的Flask应用程序配置的邮件实例

mail.send(msg)

Bulk emails

通常在web应用程序中, 每个请求会发送一到两封电子邮件. 在某些情况下, 您可能希望能够在单个批处理中发送几十或数百封电子邮件(可能是在一个外部进程中),例如命令行脚本或`cronjob`
在这种情况下, 你做的事情略有不同:

```
with mail.connect() as conn:
    for user in users:
        message = '...'
        subject = "hello, %s" % user.name
        msg = Message(recipients=[user.email], body=message, subject=subject)
        conn.send(msg)
```

与您的电子邮件主机的连接将保持活动状态, 并在发送完所有邮件后自动关闭

一些邮件服务器对单个连接中发送的邮件数量设置了限制, 您可以通过指定`MAIL_MAX_EMAILS`设置来设置重新连接之前要发送的邮件的最大数量

附件

添加附件非常简单:

```
with app.open_resource("image.png") as fp:
    msg.attach("image.png", "image/png", fp.read())
```

如果`MAIL_ASCII_ATTACHMENTS`设置为`True`, 则文件名将转换为等价的ASCII格式, 当使用修改邮件内容并在文件名为UTF-8编码时打乱内容配置规范的邮件中继时, 这可能很有用. 转换成ASCII是去掉非ASCII字符的基本方法, 对于任何可以被NFKD分解成一个或多个ASCII字符的unicode字符来说,它应该是合适的.如果你需要罗马字/音译(例如:ß→ss), 那么您的应用程序应该这样做,通过一个适当的ASCII字符串.

单元测试和禁止电子邮件

当您在单元测试内部或开发环境中发送消息时，能够禁止电子邮件发送是很有用的, 如果TESTING 设置为`True`, 电子邮件将被抑制,对消息调用`send()`不会导致实际发送任何消息
或者, 在测试环境之外, 可以将`MAIL_SUPPRESS_SEND`设置为`False`. 这也会有同样的效果. 但是, 跟踪在编写单元测试时可能发送的电子邮件仍然很有用为了跟踪发送的邮件, 使用`record_messages`方法.

```
with mail.record_messages() as outbox:
    mail.send_message(subject='testing', body='test', recipients=emails)
    assert len(outbox) == 1
    assert outbox[0].subject == "testing"
```

outbox是已发送的消息实例的列表

必须安装blinker包才能使此方法工作
请注意, 将outbox附加到g对象的旧的做事方式现在已经被废弃了

邮件头注入

为了防止头注入尝试发送主题中带有换行的消息, 发送方或接收方地址将导致BadHeaderError错误

信号支持

`Flask-Mail`现在通过`email_dispatched` 提供信号支持. 这是在发送电子邮件时发送的(即使电子邮件实际上没有发送, 例如在测试环境中)

连接到`email_dispatched` 的函数将消息实例作为第一个参数, 而Flask应用程序实例作为可选参数

```
def log_message(message, app):
    app.logger.debug(message.subject)

email_dispatched.connect(log_message)
```

API

`class flask_mail.Mail(app=None)`

功能:管理电子邮件消息
app: Flask程序实例

`send(message)`

功能:发送单个消息实例. 如果TESTING 为True, 消息将不会实际发送
参数:

- message:一个消息实例

`connect()`

功能:打开到邮件主机的连接

- send_message(*args, **kwargs)
  功能:send(msg)的快捷方式
  参数:
- 接受与消息构造函数相同的参数

`class flask_mail.Attachment(filename=None, content_type=None, data=None, disposition=None, headers=None)`

功能:封装文件附件信息
参数:

- filename:邮件附件的名字
- content_type:文件mimetype
- data:原始文件数据
- disposition:附加(如果有的话)

`class flask_mail.Connection(mail)`

功能:处理到主机的连接

`send(message, envelope_from=None)`

功能:验证和发送消息
参数:

- message：Message实例
- envelope_from：电子邮件地址将用于邮件从命令

`send_message(*args, **kwargs)

功能:send(msg)的快捷方式
参数:

- 接受与消息构造函数相同的参数

`class flask_mail.Message(subject='', recipients=None, body=None, html=None, sender=None, cc=None, bcc=None, attachments=None, reply_to=None, date=None, charset=None, extra_headers=None, mail_options=None, rcpt_options=None)`

功能:封装电子邮件消息
参数:

- `subject`:邮件主题标题
- `recipients`:电邮地址列表
- `body`:plain text message
- `html`:HTML message
- `sender`:电子邮件发件人地址，或MAIL_DEFAULT_SENDER默认
- `cc`:CC list
- `bcc`:BCC list
- `attachments`:附件实例列表
- `reply_to`:答复地址

- `date`:发送日期
- `charset`:消息字符集
- `extra_headers`:消息附加头的字典
- `mail_options`:要在MAIL FROM命令中使用的ESMTP选项列表
- `rcpt_options`:要在RCPT命令中使用的ESMTP选项列表

`attach(filename=None, content_type=None, data=None, disposition=None, headers=None)`

功能:向消息添加附件
参数:

- `filename`:附件的文件名
- `content_type`:文件mimetype
- `data`:原始文件数据
- `disposition`:附加(如果有的话)

add_recipient(recipient)

功能:将另一个收件人添加到邮件中
参数

- `recipient`: 收件人的电邮地址