信号(signals)

Flask信号(signals, or event hooking)允许特定的发送端通知订阅者发生了什么（既然知道发生了什么，那我们可以知道接下来该做什么了）。

Flask提供了一些信号（核心信号）且其它的扩展提供更多的信号。信号是用于通知订阅者，而不应该鼓励订阅者修改数据。相关信号请查阅文档。

信号依赖于Blinker库。

钩子(hooks)

Flask钩子（通常出现在蓝图或应用程序现存的方法中，比如一些内置装饰器，例如before_request）不需要Blinker库并且允许你改变请求对象(request)或者响应对象(response)。这些改变了应用程序（或者蓝图）的行为。比如before_request()装饰器。

信号看起来和钩子做同样的事情。然而在工作方式上它们存在不同。譬如核心的before_request()处理程序以特定的顺序执行，并且可以在返回响应之前放弃请求。相比之下，所有的信号处理器是无序执行的，并且不修改任何数据。

一般来说，钩子用于改变行为（比如，身份验证或错误处理），而信号用于记录事件（比如记录日志）。

创建信号

因为信号依赖于Blinker库，请确保已经安装。

如果你要在自己的应用中使用信号，你可以直接使用Blinker库。最常见的使用情况是命名一个自定义的Namespace的信号。这也是大多数时候推荐的做法:

复制代码 代码如下:

```
from blinker import Namespace
my_signals = Namespace()
```

现在你可以像这样创建新的信号:

复制代码 代码如下:

```
model_saved = my_signals.signal('model-saved')
```

这里使用唯一的信号名并且简化调试。可以用name属性来访问信号名。

对扩展开发者:

如果你正在编写一个Flask扩展，你想优雅地减少缺少Blinker安装的影响，你可以这样做使用flask.signals.Namespace类。

订阅信号

你可以使用信号的connect()方法来订阅信号。第一个参数是信号发出时要调用的函数，第二个参数是可选的，用于确定信号的发送者。一个信号可以拥有多个订阅者。你可以用disconnect()方法来退订信号。

对于所有的核心Flask信号，发送者都是发出信号的应用。当你订阅一个信号，请确保也提供一个发送者，除非你确实想监听全部应用的信号。这在你开发一个扩展的时候尤其正确。

比如这里有一个用于在单元测试中找出哪个模板被渲染和传入模板的变量的助手上下文管理器:

复制代码 代码如下:

```
from flask import template_rendered
from contextlib import contextmanager

@contextmanager
def captured_templates(app):
```

```
    recorded = []
    def record(sender, template, context, **extra):
        recorded.append((template, context))
    template_rendered.connect(record, app)
    try:
        yield recorded
    finally:
        template_rendered.disconnect(record, app)
```

现在可以轻松地与测试客户端配对:

复制代码 代码如下:

```
with captured_templates(app) as templates:
    rv = app.test_client().get('/')
    assert rv.status_code == 200
    assert len(templates) == 1
    template, context = templates[0]
    assert template.name == 'index.html'
    assert len(context['items']) == 10
```

确保订阅使用了一个额外的 **extra 参数，这样当 Flask 对信号引入新参数时你的调用不会失败。

代码中，从 with 块的应用 app 中流出的渲染的所有模板现在会被记录到templates 变量。无论何时模板被渲染，模板对象的上下文中添加上它。

此外，也有一个方便的助手方法(connected_to()) ，它允许你临时地用它自己的上下文管理器把函数订阅到信号。因为上下文管理器的返回值不能按那种方法给定，则需要把列表作为参数传入:

复制代码 代码如下:

```
from flask import template_rendered

def captured_templates(app, recorded, **extra):
    def record(sender, template, context):
        recorded.append((template, context))
    return template_rendered.connected_to(record, app)
```

上面的例子看起来像这样:

复制代码 代码如下:

```
templates = []
with captured_templates(app, templates, **extra):
    ...
    template, context = templates[0]
```

发送信号

如果你想要发送信号，你可以通过调用 send() 方法来达到目的。它接受一个发件者作为第一个参数和一些可选的被转发到信号用户的关键字参数:

复制代码 代码如下:

```
class Model(object):
    ...

    def save(self):
        model_saved.send(self)
```

永远尝试选择一个合适的发送者。如果你有一个发出信号的类，把 self 作为发送者。如果你从一个随机的函数发出信号，把current_app._get_current_object()作为发送者。

基于信号订阅的装饰器

在Blinker 1.1中通过使用新的connect_via()装饰器你也能够轻易地订阅信号:

复制代码 代码如下:

```
from flask import template_rendered

@template_rendered.connect_via(app)
def when_template_rendered(sender, template, context, **extra):
    print 'Template %s is rendered with %s' % (template.name, context)
```

举例

模板渲染

template_rendered信号是Flask核心信号。

当一个模版成功地渲染，这个信号会被发送。这个信号与模板实例 template 和上下文的词典（名为 context ）一起调用。

信号发送：

复制代码 代码如下:

```
def _render(template, context, app):
    """Renders the template and fires the signal"""
    rv = template.render(context)
    template_rendered.send(app, template=template, context=context)
    return rv
```

订阅示例:
复制代码 代码如下:

```
def log_template_renders(sender, template, context, **extra):
    sender.logger.debug('Rendering template "%s" with context %s',
                        template.name or 'string template',
                        context)
```

```
from flask import template_rendered
template_rendered.connect(log_template_renders, app)
```

帐号追踪

user_logged_in是Flask-User中定义的信号，当用户成功登入之后，这个信号会被发送。

发送信号：

复制代码 代码如下:

```
# Send user_logged_in signal
user_logged_in.send(current_app._get_current_object(), user=user)
```

下面这个例子追踪登录次数和登录IP：
复制代码 代码如下:

```
# This code has not been tested

from flask import request
from flask_user.signals import user_logged_in

@user_logged_in.connect_via(app)
def _track_logins(sender, user, **extra):
    user.login_count += 1
    user.last_login_ip = request.remote_addr
    db.session.add(user)
    db.session.commit()
```

小结

信号可以让你在一瞬间安全地订阅它们。例如，这些临时的订阅对测试很有帮助。使用信号时，不要让信号订阅者（接收者）发生异常，因为异常会造成程序中断。