DeanWu 发布于 Python之路
2017年02月23日 · 5.3k 次阅读

Django signal 使用总结

本文最早发表于个人博客 Pylixm'wiki-django signal使用总结

最近在已经开发好的项目上加功能，想到了django的signal，整理记录如下备查。

什么是django的signal

官方文档描述如下：

> Django includes a “signal dispatcher” which helps allow decoupled applications get notified when actions occur elsewhere in the framework.In a nutshell, signals allow certain senders to notify a set of receivers that some action has taken place. They’re especially useful when many pieces of code may be interested in the same events.

Django内部包含了一位“信号调度员”：当某事件在框架内发生时，它可以通知到我们的应用程序。 简而言之，当event（事件）发生时，signals（信号）允许若干 senders（寄件人）通知一组 receivers（接收者）。这在我们多个独立的应用代码对同一事件的发生都感兴趣时，特别有用。

个人理解，django的signal可理解为django内部的钩子，当一个事件发生时，其他程序可对其作出相关反应，可通过signal来回调定义好的处理函数（receivers），从而更大程度的解耦我们的系统。

最佳使用场景

通知类

通知是signal最常用的场景之一。例如，在论坛中，在帖子得到回复时，通知楼主。从技术上来讲，我们可以将通知逻辑放在回复保存时，但是这并不是一个好的处理方式，这样会时程序耦合度增大，不利于系统的后期扩展维护。如果我们在回复保存时，只发一个简单的信号，外部的通知逻辑拿到信号后，再发送通知，这样回复的逻辑和通知的逻辑做到了分开，后期维护扩展都比较容易。

初始化类

信号的另一个列子便是事件完成后，做一系列的初始化工作。

其他一些使用场景总结

以下情况不要使用signal:

- signal与一个model紧密相关, 并能移到该model的save()时
- signal能使用model manager代替时
- signal与一个view紧密相关, 并能移到该view中时

以下情况可以使用signal:

- signal的receiver需要同时修改对多个model时
- 将多个app的相同signal引到同一receiver中处理时
- 在某一model保存之后将cache清除时
- 无法使用其他方法, 但需要一个被调函数来处理某些问题时

如何使用

django 的 signal 使用可分为2个模块：

- signal ： signal定义及触发事件
- receiver : signal 接受函数

内建signal的使用

django 内部有些定义好的signal供我们使用：

模型相关：

- pre_save 对象save前触发
- post_save 对象save后触发
- pre_delete 对象delete前触发
- post_delete 对象delete后触发
- m2m_changed ManyToManyField 字段更新后触发

请求相关：

- request_started 一个request请求前触发
- request_finished request请求后触发

针对django自带的signal，我们只需要编写receiver 即可，使用如下。

第一步，编写receiver并绑定到signal

myapp/signals/handlers.py

```
from django.core.signals import request_finished

## decorators 方式绑定
@receiver(request_finished, dispatch_uid="request_finished")
def my_signal_handler(sender, **kwargs):
    print("Request finished!================================")

# 普通绑定方式
def my_signal_handler(sender, **kwargs):
    print("Request finished!================================")

request_finished.connect(my_signal_handler)

#########################################################
# 针对model 的signal
from django.dispatch import receiver
from django.db.models.signals import post_save

from polls.models import MyModel


@receiver(post_save, sender=MyModel, dispatch_uid="mymodel_post_save")
def my_model_handler(sender, **kwargs):
 print('Saved: {}'.format(kwargs['instance'].__dict__))
```

- dispatch_uid 确保此receiver 只调用一次

第二步，加载signal

myapp/__init__py

```
default_app_config = 'myapp.apps.MySendingAppConfig'
```

myapp/apps.py

```
from django.apps import AppConfig


class MyAppConfig(AppConfig):
    name = 'myapp'

    def ready(self):
        # signals are imported, so that they are defined and can be used
        import myapp.signals.handlers
```

到此，当系统受到request 请求完成后，便会执行receiver。

其他内建的signal，参考官方文档：
https://docs.djangoproject.com/en/1.9/topics/signals/

自定义signal的使用

自定义signal，需要我们编写 signal和 receiver 。

第一步, 编写signal

myapp.signals.signals.py

```
import django.dispatch

my_signal = django.dispatch.Signal(providing_args=["my_signal_arg1", "my_signal_arg_2"])
```

第二步，加载signal

myapp/__init__py

```
default_app_config = 'myapp.apps.MySendingAppConfig'
```

myapp/apps.py

```
from django.apps import AppConfig


class MyAppConfig(AppConfig):
    name = 'myapp'
```

```
    def ready(self):
        # signals are imported, so that they are defined and can be used
        import myapp.signals.handlers
```

第三步，事件触发时，发送signal

myapp/views.py

```
from .signals.signals import my_signal

my_signal.send(sender="some function or class",
               my_signal_arg1="something", my_signal_arg_2="something else"])
```

自定义的signal，django已经为我们编写了此处的事件监听。

第四步，收到signal，执行receiver

myapp/signals/handlers.py

```
from django.dispatch import receiver
from myapp.signals.signals import my_signal


@receiver(my_signal, dispatch_uid="my_signal_receiver")
def my_signal_handler(sender, **kwargs):
    print('my_signal received')
```

此时，我们自定义的signal 便开发完成了。

总结

- django signal 的处理是同步的，勿用于处理大批量任务。
- django signal 对程序的解耦、代码的复用及维护性有很大的帮助。

以上为个人观点，如有疑问欢迎交流。

参考

http://sabinemaennel.ch/django/signals-in-django/
https://docs.djangoproject.com/en/1.10/topics/signals/