简述Django的信号机制

Django 框架包含了一个信号机制，它允许若干个发送者（sender）通知一组接收者（receiver）某些操作已经发生了， 接收者收到指令信号后再去执行特定的操作。这在多处业务逻辑与同一事件有关联的情况下是很有用的。其实这种信号机制就是观察者模式，又叫发布-订阅（Publish/Subscribe）模式 。当发生一些动作的时候，发出信号，然后监听了这个信号的函数就会执行。在本节将介绍 Django 的信号机制。

1. Django中的信号机制

1) 信号机制基本概念

Django 内置了许多信号，允许用户的代码获得特定操作的通知。例如在 Model 保存前触发的信号 `pre_save`、在 Model 保存后触发的信号 `post_save` 等。同时，为了满足实际的业务场景，Django 也允许自定义信号，这也非常容易实现。那么什么是信号呢？有编程基础的同学也许会了解一些，但如果你是初学者可能会不怎么了解，那么在这里就给大家做一下介绍。

在 Linux 编程中也存在信号的概念，这里的信号主要用于进程之间的通信，用来通知进程发生的异步事件。而 Django 框架中的信号与其不同。Django 中的信号用于在框架执行操作时解耦。当某些动作发生的时候，系统会根据信号定义的函数执行相应的操作，Django 中的信号主要包含以下三个要素：

- 发送者（sender）：信号的发出方。
- 信号（signal）：发送的信号本身。
- 接收者（receiver）：信号的接收者。

其中信号接收者其实就是一个简单的回调函数，将这个函数注册到信号上，当特定的事件发生时，发送者发送信号，回调函数被执行。

2) 信号机制适用场景

通知是信号中最常用的场景，比如当一个用户登录成功后，给该用户发送通知消息，或者在论坛、博客当你更新话题或者动态。就可以使用信号做信息的推送；当你的发布的动态有其他的用户给你评论的时候，也可以使用信号来通知你。

在发布新的话题或话题有了新的评论时，就会发送一个信号，预先定义好的信号接收者执行对应的操作，然后达到对外发布推送消息的目的。这样不仅消除了不同业务逻辑的耦合，而且在这种场景下往往只是所推送消息不同，所以在一定程度上能够减少代码的冗余。

信号的另一种使用场景是某些事件发生之后，做一些清理或初始化的工作。比如：从数据库中读取数据如果延迟较高的情况下，我们可以将其放在内存或缓存数据库中。但是这样的话，以后对该数据表进行更新就不仅要同步到数据库中，还需要同步到缓存中。在这里，我们仍然可以使用信号来完成对缓存的更新。每当数据表更新完成后，就发送信号通知回调函数完成更新缓存的操作。

3) 信号不适用的场景

看了上面的举例，你会感觉 Django 的信号机制有很强大的功能，确实如此，不过在一些场景下也不适合使用信号，比如说耗时任务，由于信号是同步执行的，因此耗时的任务会影响服务体验，所以此时需要考虑使用异步任务而不是信号机制了。

2. Django中的内置信号

Django 提供一组内置的信号，允许用户的编写的代码获得 Django 特定操作的通知。Django 的 Signal 类（信号类）定义于如下路径文件中：

django/dispatch/dispatcher.py
它的构造函数如下所示：

```
class Signal:
    def __init__(self, providing_args=None, use_caching=False):
        """
        Create a new signal.
        """
```

它接受两个参数，它的分别如下所示：

- providing_args：可选的列表类型，其中每一个元素都是字符串，标识信号提供给接收者的参数。
- use_caching：默认值是 False，如果设置为 True，则缓存会被设置为弱引用。

1) 与Model相关的内置信号

上面参数大家有一点可能不明白，那就是弱引用，它是 Python 的语言特性，这在下一节中会进行分析讲解。下面我们介绍 Django 中的一些常用内置信号。首先，介绍与 Model 相关的信号，这些信号由各个 Model 的方法发送，如 save、__init__等，且通常都是成对出现的，如下所示：

- django.db.models.signals.pre_init 与 django.db.models.signals.post_init：实例化模型之前与之后发送的信号，即在__init__方法执行的前后。
- django.db.models.signals.pre_save 与 django.db.models.signals.post_save：模型实例保存（执行save方法）前后发送的信号。
- django.db.models.signals.pre_delete 与 django.db.models.signals.post_delete：模型实例或 QuerySet 的 delete() 方法执行前后发送的信号。
- django.db.models.signals.m2m_changed：模型实例中的ManyToManyField（多对多）字段被修改（add，remove，clear）的前后发送的信号。

2) 与HTTP相关的内置信号

Django 对于 HTTP 请求的处理定义了三个信号：

- django.core.signals.request_started 与 django.core.signals.request_finished：建立和关闭 HTTP 请求时发送的信号。
- django.core.signals.got_request_exception：在处理HTTP请求的过程中出现异常，将会发送此信号。

3) 与数据库迁移相关的内置信号

在做数据库迁移（migrate）时，Django也会发送信号，这类信号由 Django 的管理工具发送：

django.db.models.signals.pre_migrate 与 django.db.models.signals.post_migrate 表示在执行 migrate 命令的前后触发。

对于 Django 内置的信号，只需要定义回调函数并将它注册到信号上，这里的回调函数作为信号得接收者（receiver）。当程序执行到相应的操作时，自动触发信号，执行回调函数。为了更好地理解信号，在下一节我们将讲述 Django 内置信号的执行过程以及它的相关特性，从而深入理解 Django 的信号机制。