初識 Django

Django 最初被設計用於具有快速開發需求的新聞類網站，目的是要實現簡單快捷的網站開發。以下內容簡要介紹了如何使用 Django 實現一個資料庫驅動的 Web 應用。

為了讓您充分理解 Django 的工作原理，這份文件為您詳細描述了相關的技術細節，不過這並不是一份入門教學或者是參考文件（我們當然也為您準備了這些）。如果您想要馬上開始一個專案，可以從 實例教學 開始入手，或者直接開始閱讀詳細的 參考文件 。

設計模型

Django 無需資料庫就可以使用，它提供了 對象關聯映射器 透過此技術，你可以使用 Python 程式來描述資料庫結構。

你可以使用強大的 數據-模型語句 來描述你的數據模型，這解決了數年以來在資料庫模式中的難題。以下是一個簡明的例子：

mysite/news/models.py

```python
from django.db import models

class Reporter(models.Model):
    full_name = models.CharField(max_length=70)

    def __str__(self):
        return self.full_name

class Article(models.Model):
    pub_date = models.DateField()
    headline = models.CharField(max_length=200)
    content = models.TextField()
    reporter = models.ForeignKey(Reporter, on_delete=models.CASCADE)

    def __str__(self):
        return self.headline
```

應用數據模型

接下來，執行Django命令欄實用程式以自動建立資料庫表：

```
$ python manage.py makemigrations
$ python manage.py migrate
```

該 **makemigrations** 命令查找所有可用的models，為任意一個在資料庫中不存在對應資料表的model建立 migrations 腳本文件。**migrate** 命令則執行這些 migrations 自動建立資料庫表。還提供可選的 更豐富的控制模式。

享用便捷的 API

接下來，你就可以使用一套便捷而豐富的 Python API 開啟你的數據。API是動態建立的，不需要程式產生：

```
# Import the models we created from our "news" app
>>> from news.models import Article, Reporter

# No reporters are in the system yet.
```

```
>>> Reporter.objects.all()
<QuerySet []>

# Create a new Reporter.
>>> r = Reporter(full_name='John Smith')

# Save the object into the database. You have to call save() explicitly.
>>> r.save()

# Now it has an ID.
>>> r.id
1

# Now the new reporter is in the database.
>>> Reporter.objects.all()
<QuerySet [<Reporter: John Smith>]>

# Fields are represented as attributes on the Python object.
>>> r.full_name
'John Smith'

# Django provides a rich database lookup API.
>>> Reporter.objects.get(id=1)
<Reporter: John Smith>
>>> Reporter.objects.get(full_name__startswith='John')
<Reporter: John Smith>
>>> Reporter.objects.get(full_name__contains='mith')
<Reporter: John Smith>
>>> Reporter.objects.get(id=2)
Traceback (most recent call last):
    ...
DoesNotExist: Reporter matching query does not exist.

# Create an article.
>>> from datetime import date
>>> a = Article(pub_date=date.today(), headline='Django is cool',
...     content='Yeah.', reporter=r)
>>> a.save()

# Now the article is in the database.
>>> Article.objects.all()
<QuerySet [<Article: Django is cool>]>

# Article objects get API access to related Reporter objects.
>>> r = a.reporter
>>> r.full_name
'John Smith'

# And vice versa: Reporter objects get API access to Article objects.
>>> r.article_set.all()
<QuerySet [<Article: Django is cool>]>

# The API follows relationships as far as you need, performing efficient
# JOINs for you behind the scenes.
# This finds all articles by a reporter whose name starts with "John".
>>> Article.objects.filter(reporter__full_name__startswith='John')
<QuerySet [<Article: Django is cool>]>

# Change an object by altering its attributes and calling save().
>>> r.full_name = 'Billy Goat'
>>> r.save()
```

```
# Delete an object with delete().
>>> r.delete()
```

一個動態管理接口：並非徒有其表

當你的模型完成定義，Django 就會自動產生一個專業的生產級 管理接口 ——一個允許認證用戶增加、更改和刪除對象的 Web 網站。你只需在 admin 網站上注冊你的模型即可：

mysite/news/models.py

```
from django.db import models

class Article(models.Model):
    pub_date = models.DateField()
    headline = models.CharField(max_length=200)
    content = models.TextField()
    reporter = models.ForeignKey(Reporter, on_delete=models.CASCADE)
```

mysite/news/admin.py

```
from django.contrib import admin

from . import models

admin.site.register(models.Article)
```

這樣設計所遵循的理念是，網站編輯人員可以是你的員工、你的客戶、或者就是你自己——而你大概不會樂意去廢半天勁建立一個只有內容管理功能的管理管理界面。

建立 Django 應用的典型流程是，先建立數據模型，然後搭建管理網站，之後你的員工（或者客戶）就可以向網站裡填充數據了。後面我們會談到如何展示這些數據。

規劃 URLs

簡潔優雅的 URL 規劃對於一個高質量 Web 應用來說至關重要。Django 推崇優美的 URL 設計，所以不要把諸如 **.php** 和 **.asp** 之類的冗餘的後綴放到 URL 裡。

為了設計你自己的 URLconf，你需要建立一個叫做 URLconf 的 Python 模組。這是網站的目錄，它套件含了一張 URL 和 Python 回調函數之間的映射表。URLconf 也有利於將 Python 程式與 URL 進行解耦（譯注：使各個模組分離，獨立）。

下面這個 URLconf 適用於前面 **Reporter**/**Article** 的例子：

mysite/news/urls.py

```
from django.urls import path

from . import views

urlpatterns = [
    path('articles/<int:year>/', views.year_archive),
    path('articles/<int:year>/<int:month>/', views.month_archive),
    path('articles/<int:year>/<int:month>/<int:pk>/', views.article_detail),
]
```

上述程式將 URL 路徑映射到了 Python 回調函數（“視圖”）。路徑字符串使用參數標籤從URL中“捕獲”相應值。當用戶請求頁面時，Django 依次遍歷路徑，直至初次匹配到了請求的 URL。(如果無匹配項，Django 會調用 404 視圖。) 這個過程非常快，因為路徑在載入時就編譯成了正則表達式。

一旦有 URL 路徑匹配成功，Django 會調用相應的視圖函數。每個視圖函數會接受一個請求對象——套件含請求元資訊——以及在匹配式中獲取的參數值。

例如，當用戶請求了這樣的 URL "/articles/2005/05/39323/"，Django 會調用 **`news.views.article_detail(request, year=2005, month=5, pk=39323)`**。

編寫視圖

視圖函數的執行結果只可能有兩種：返回一個套件含請求頁面元素的 **`HttpResponse`** 對象，或者是拋出 **`Http404`** 這類異常。至於執行過程中的其它的動作則由你決定。

通常來說，一個視圖的工作就是：從參數獲取數據，裝載一個模板，然後將根據獲取的數據對模板進行渲染。下面是一個 **`year_archive`** 的視圖樣例：

mysite/news/views.py

```
from django.shortcuts import render

from .models import Article

def year_archive(request, year):
    a_list = Article.objects.filter(pub_date__year=year)
    context = {'year': year, 'article_list': a_list}
    return render(request, 'news/year_archive.html', context)
```

這個例子使用了 Django 模板系統，它有著很多強大的功能，而且使用起來足夠簡單，即使不是程式員也可輕松使用。

設計模板

上面的程式載入了 **`news/year_archive.html`** 模板。

Django 允許設置搜索模板路徑，這樣可以最小化模板之間的冗餘。在 Django 設置中，你可以透過 **`DIRS`** 參數指定一個路徑欄表用於檢索模板。如果第一個路徑中不套件含任何模板，就繼續檢查第二個，以此類推。

讓我們假設 **`news/year_archive.html`** 模板已經找到。它看起來可能是下面這個樣子：

mysite/news/templates/news/year_archive.html

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

{% block title %}Articles for {{ year }}{% endblock %}

{% block content %}
<h1>Articles for {{ year }}</h1>

{% for article in article_list %}
    <p>{{ article.headline }}</p>
    <p>By {{ article.reporter.full_name }}</p>
    <p>Published {{ article.pub_date|date:"F j, Y" }}</p>
{% endfor %}
{% endblock %}
```

我們看到變數都被雙大括號括起來了。 **`{{ article.headline }}`** 的意思是“輸出 article 的 headline 屬性值”。這個“點”還有更多的用途，比如查找字典鍵值、查找索引和函數調用。

我們注意到 **{{ article.pub_date|date:"F j, Y" }}** 使用了 Unix 風格的“管道符”（“|”字符）。這是一個模板過濾器，用於過濾變數值。在這裡過濾器將一個 Python datetime 對象轉化為指定的格式（就像 PHP 中的日期函數那樣）。

你可以將多個過濾器連在一起使用。你還可以使用你 自定義的模板過濾器 。你甚至可以自己編寫 自定義的模板標籤 ，相關的 Python 程式會在使用標籤時在管理執行。

Django 使用了 "模板繼承" 的概念。這就是 **{% extends "base.html" %}** 的作用。它的含義是"先載入名為 base 的模板，並且用下面的標記塊對模板中定義的標記塊進行填充"。簡而言之，模板繼承可以使模板間的冗餘內容最小化：每個模板只需套件含與其它文件有區別的內容。

下面是 base.html 可能的樣子，它使用了 靜態文件 ：

mysite/templates/base.html

```
{% load static %}
<html>
<head>
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    <img src="{% static "images/sitelogo.png" %}" alt="Logo">
    {% block content %}{% endblock %}
</body>
</html>
```

簡而言之，它定義了這個網站的外觀（利用網站的 logo），並且給子模板們挖好了可以填的”坑“。這就意味著你可以透過修改基礎模板以達到重新設計網頁的目的。

它還可以讓你利用不同的基礎模板並重用子模板建立一個網站的多個版本。透過建立不同的基礎模板，Django 的建立者已經利用這一技術來創造了明顯不同的手機版本的網頁。

注意，你並不是非得使用 Django 的模板系統，你可以使用其它你喜歡的模板系統。盡管 Django 的模板系統與其模型層能夠集成得很好，但這不意味著你必須使用它。同樣，你可以不使用 Django 的資料庫 API。你可以用其他的資料庫抽象層，像是直接讀取 XML 文件，亦或直接讀取磁碟文件，你可以使用任何方式。Django 的任何組成——模型、視圖和模板——都是獨立的。

這僅是基本入門知識

以上只是 Django 的功能性概述。Django 還有更多實用的特性：

- 緩存框架 可以與 memcached 或其它後端集成。
- 聚合器框架 可以透過編寫一個小型 Python 類來建立 RRS 和 Atom摘要。
- 功能豐富的自動產生的管理——這份概要只是簡單介紹了下。

接下來您可以 下載 Django ， 閱讀 實例教學 ， 然後加入我們的 社區 ！ 感謝您的關注!
