django 开发简介一

王敏虎

电子科协网宣部

2017 年 6 月 27 日

EESAST

1. HTTP 与网页开发
2. Django 概述
3. Django 安装与第一个项目
4. View：第一个页面

EESAST

1 HTTP 与网页开发

2 Django 概述

3 Django 安装与第一个项目

4 View：第一个页面

网页/网站是什么？

- 网页是一堆文件的组合，其中 HTML 文件和静态文件 (图片、视频) 控制内容，CSS 文件控制样式，JavaScript 文件控制动态。
- 网页储存在特殊的计算机中，这些计算机称为服务器 (server)，访问网站的过程就是把网页文件从服务器中取回的过程。
- 计算机 (client) 与服务器的通信遵循所谓的 HTTP 协议，HTTP 协议可以理解为一种供 client 和 server 交流的一种特殊的格式。
- HTTP 协议的内容十分丰富，如果希望详细了解可以阅读《HTTP 权威指南》一书，书中不但详细介绍了 HTTP 协议的各方面知识，并且就如何在 HTTP 协议层面对网站性能进行优化提出了建议。

EESAST

我们使用到的各种工具在做什么？

- Django：后续处理 HTTP Request——库管
- MySQL：数据库，储存数据——仓库
- Apache：处理 HTTP Request——看门的
- Linux：操作系统——仓库所在的工业园区

1 HTTP 与网页开发

2 Django 概述

3 Django 安装与第一个项目

4 View：第一个页面

Django 概述

Django 是什么?

Django

Django 是基于 python 语言的网页开发框架, 采用了 MVC 的软件设计模式, 最初被用于劳伦斯出版集团旗下的新闻网站开发。Django 的目标是使开发复杂的数据库驱动网站变得简单，因此 Django 注重组件的可重用性和模块的可插拔性。DRY(Don't Repeat Yourself) 原则在 Django 中也有很好的体现。

Django 现状

Django 目前由 Django 基金会 (Django Software Foundation) 管理，基于 BSD 许可证开源。
官方网站是`www.djangoproject.com`。

Django 概述

Django 是什么?

Django 核心组件

Django 使用 MVC(模型-视图-控制器模式，因此其包含三个基本组件：

- 面向对象的数据库接口
- 基于正则表达式的 URL 分发器
- 视图及模板系统

Django 组件

除了核心组件，Django 还提供了有用的核心框架和内置应用，例如：

- 一个轻量的 web 服务器
- 表单序列化及验证系统
- 缓存框架
- 可扩展认证系统
- 动态站点管理页面

Django 概述

Django 的性能瓶颈？

- 通常，网站服务不是计算密集型的应用，性能瓶颈主要存在于硬盘 I/O 中，Python 语言性能偏弱的缺点不会很明显的暴露出来。
- 由于 python 的特性，性能敏感的部分可以很容易使用 C 重写并和 python 部分耦合在一起，可以同时保证开发速度和代码性能。
- Instagram 在全球拥有超过七亿用户，它使用的开发框架为 Django。

EESAST

Django 概述

为什么要选择 Django?

Django 的优点

- 基于 python 语言，降低学习成本
- 通过复用已经设计成熟的组件和框架，提高开发速度
- 不同组件之间耦合极低，方便协同开发
- 有优秀的官方文档支持
- 大版本内前后兼容

Django 的不足

- 迭代速度快，缺乏有体系的入门书籍
- 缺乏中文参考资料
- 框架大，细节多，需要花费时间熟悉 API

1 HTTP 与网页开发

2 Django 概述

3 Django 安装与第一个项目

4 View：第一个页面

Django IDE

Django 使用简单的文本编辑器和命令行工具即可开发，也有非常方便的集合了多种功能的 IDE 辅助工具，例如：

- JetBrains PyCharm
- Microsoft Visual Studio 2015/2017

在本次讲座中，我们不使用也不介绍具体 IDE，但是学习 IDE 的使用方法对于实际开发是有帮助的。

Python/Pip 安装

pip

pip 是 python 的包管理工具。一般通过 pip 来安装、管理 python 包，如今天要用到的 Django。

windows

python 的下载地址为：`https://www.python.org/downloads/`
下载已经编译好的 python 安装程序，遵循与安装普通软件相同的步骤即可完成安装，记住 python 的安装目录。可以选择令其按默认设置安装，如果需要自定义安装注意勾选“install pip”和“add python to PATH”。

mac/linux

macOS/linux 中通常已经安装有 python2.x 版本，可以到官网下载 python3.x 版本的安装包。pip 可通过操作系统的包管理程序下载或通过源码编译等方式安装。

Django 安装

pip 安装 django

找到 python 的安装目录。进入目录中的 Scripts 文件夹，如果 pip 安装成功，那么文件夹中应该可以找到 pip.exe 文件。在该目录下打开 bash(windows 下为 cmd/powershell)，输入 pip install django。
pip 会自动下载并安装 django。

注意

在 macOS/linux 下，可能会因为账户权限不足导致安装失败的情况出现，解决方法因操作系统而异，通常使用 sudo 提权即可。

Django 安装

Django 确认

在命令行中输入 python，进入 python 交互界面。
输入 import django，如果不报错，说明 django 安装成功。
通过 django.VERSION 可以查看 django 的版本，最新版本为 1.11 版，django 在大版本号，即 1.x 内保证前后兼容。

Django 创建项目

可以通过如下代码在当前文件夹创建一个 Django 项目：

```
python /path/to/python/Scripts/django-admin.py startproject <
    projectname>
```

Django 会为项目自动生产基本的框架文件。打开项目文件夹，可以看到如下文件：

- <projectname>
  - __init.py__
  - settings.py
  - urls.py
  - wsgi.py
- manage.py

django 初始文件

mamage.py 文件

manage.py 是开发中使用的脚本文件，可以理解为进行 django 开发的“遥控器”，可以通过该文件打开调试用的 web 服务器、与数据库交互、处理静态文件等。

<projectname>/settings.py

settings.py 是整个 django 项目的设置文件，其中保存着例如时区、语言等设置。

<projectname>/urls.py

urls.py 是 django 项目的 URL 分发文件，即将请求的网址 (url) 分配给对应的处理函数。

setting.py 中的几个重要设置

LANGUAGE_CODE

django 为其内置框架提供了国际化翻译支持，因此首先需要将使用语言改为 zh-hans，即简体中文。

TIME_ZONE

为处理网站国际化中遇到的时区问题，django 默认储存格林尼治标准时间，并承担了将用户输入时间转换为标准时间和将标准时间转换为当地时间渲染页面的繁琐任务。因此需要将时区调整为东八区 Asia/Shanghai 。

DEBUG

DEBUG 是一个重要的开关。如果 DEBUG==True, 那么 django 将在出错时显示详细的错误页面以供开发者进行 debug，但是这会将网站的漏洞和源代码暴露出来，因此在开发环境以外，应该始终保持 DEBUG == False 状态。

EESAST

调试用 web 服务器

虽然还没有开始编写 django 代码，但是可以先启动 django 自带的轻量 web 服务器，确认 django 正常工作。在 shell 中输入：python manage.py runserver。在这里 django 会警告我们还有一些操作没有进行，我们稍后再解释这一警告的含义。
在浏览器中输入`http://127.0.0.1:8000`，如果一切正常的话，可以看到 django 的欢迎页面。

说明

实际上这个服务器的功能是非常完备的，如果有独立 ip，原则上可以通过该轻量级服务器架设站点到互联网上，但是理论上该服务器同时只支持一人访问，在设计时也缺乏对安全性的周密设计，因此只适用于开发时使用。善加利用该服务器可以为开发提供便利，例如，当同时适配 PC 端和手机端时，可以直接使用手机浏览网站渲染效果。

EESAST

1 HTTP 与网页开发

2 Django 概述

3 Django 安装与第一个项目

4 View：第一个页面

app

app

django 以 app 为功能的基本单位，这是为了组件复用考虑，例如，如果需要在网站中添加一套投票系统，那么可以将成熟的开源代码直接加入项目，并在 settings.py 的 INSTALLED_APPS 中加入对应 app 名称即可。这为基于 django 的二次开发提供了方便。

创建 app

创建新 app 的方式如下：

```
python manage.py startapp <appname>
```

注意，在创建新的 app 后，需要将新 app 的名称填入 settings.py 的 INSTALLED_APPS 中，才能使 app 生效。

urls.py

url 分发

django 使用一种非常优雅的方式——正则表达式处理 url，一个简单的 url 包括三部分：协议、域名和路径，例
如`https://www.eesast.com/post/lecture/1/`, 其中 http 表示其使用 http 协议，www.eesast.com 是其域名，而/post/lecture/1 是其路径，django 使用正则表达式解析这一路径，并分配给对应的 view 函数处理。这也正是 MVC 架构中的控制部分。

正则表达式

正则表达式是一个广泛使用的概念和工具，直观的说，一个正则表达式是一个包含某些特殊字符的字符串，它可以用来匹配遵循某一模式的具体字符串。正则表达式在许多编程语言中都是语言或标准库中的一部分，例如 python、javascript、C++(11)。
为编写 django 代码，并不需要成为正则表达式的专家，在简单的情况，也是大多数情况下，只需要掌握两个符号，^ 和\$, 前者表示以此为开头，后者表示以此为结束，例如，"^abc" 表示以 abc 开头的所有字符串。

逐级分发

django 支持逐级传递 URL，例如，如果用户请求的路径为/a/b/c，而设定的匹配规则为凡是以 a 开头的 URL 都交给某一 app 处理，那么 django 会将/b/c 传递给该 app 对应的 urls.py 进行下一级的 url 分发。

编写规则

url 分发规则被放置在 urls.py 中的 urlconf 列表中：

```
from django.conf.urls import url,include
from django.contrib import admin

urlpatterns = [
url(r'^try/', include('MyApp.urls'), name="MyApp"),
url(r'^admin/', admin.site.urls)
]
```

这段代码表明，所以以"try" 为路径开头的请求都会交给 MyApp 这个 App 处理，那么在 MyApp 中，我们还需要有一个 urls.py 负责将这一请求具体分配给某个函数处理[1]。

```
#MyApp/urls.py
urlpatterns = [
url(r'^hello$', views.Hello, name = "hello")
]
```

我们将所有"http://domine/try/hello" 的请求交给了 Hello 函数处理。

[1]实际上，Django 支持面向对象的写法，使用这一写法，处理请求的将是类而不是函数。

view 中的函数

在 Django 中，函数与界面紧密的连接在一起，可以认为一个请求 (url) 对应一个函数，同时对应一个界面。Django 工作的一般流程即是函数接受用户的请求，并返回对应的界面。一个简单的 Hello 函数如下:

```
#MyApp/views.py
from django.http import HttpResponse

def Hello(request):
    return HttpResponse("Hello,world!")
```

在这里，简单的 Hello 函数接受包含了用户请求的复杂数据结构 request，返回了一个 HttpResponse 类型的数据。Django 会将它按照 HTTP 协议的要求包装成正确的格式并返回，其结果就是 client 端收到来自网站的欢迎信息

本质上，一个网页只是一串符合某些规则的字符串。Django 为动态生成网页提供了一套很好的模板工具。在 APP 中新建一个"Template" 文件夹，在文件夹中再新建一个与 APP 同名的文件夹，模板即放在该文件夹中[2]。
一个模板是一个特殊的 HTML 文件，在这个文件中可以使用所有正常的 HTML 语法并和 CSS、JavaScript 耦合。但是支持 Django 特殊的标记，例如：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset = "utf-8">
        <title>
            Hello!
        </title>
    </head>
    <body>
        {{content}}
    </body>
</html>
```

[2]引入这样的文件结构与静态文件管理有关，在部署中我们会再涉及。

其中的"content" 是模板待补充的部分，我们需要在 view 函数中指出它的内容

```
from django.shortcuts import render
def Hello2(request):
    IP = request.META['REMOTE_ADDR']
    return render(request,'MyApp/Hello.html',{'content':"Hello,{0}
        ".format(IP)})
```

render 函数的调用格式为

render(request,template,dict of content)

在这里，我们从 request 结构中获取到了访问者的 IP 地址，并在返回结果中渲染出来。

模板中另一个基本的用法是 if-else 结构。与刚才的内容结构稍有不同，if-else 结构的使用方法如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset = "utf-8">
        <title>
            Hello!
        </title>
    </head>
    <body>
        {% if content %}
           {{content}}
        {% else %}
            Nothing
        {{% endif %}}
    </body>
</html>
```

如果希望渲染一个可迭代的结构，Django 提供了直接的解决方案。

```
{% if list %}
    <ul>
    {% for item in list %}
        <li>{{ item }}</li>
    {% endfor %}
    </ul>
    {% else %}
    Nothing
{% endif %}
```