第 3 章 服务器搭建

3.1 服务器功能简介

服务器在网络中为其他客户机(如 PC、智能手机、ATM 等终端,甚至是火车系统等大型设备)提供计算或者应用服务。服务器具有高速的 CPU 运算能力、长时间的可靠运行、强大的 I/O 外部数据吞吐能力以及更好的扩展性。根据服务器所提供的服务,一般来说服务器都具备承担响应服务请求、承担服务、保障服务的能力。服务器架构(如图 3.1 所示)主要分为网络通信层、应用逻辑层和数据资产层。网络通信层接受并处理客户端发送的信息,将收到的信息处理之后传递到应用逻辑层,由应用逻辑层决定进行相应的操作,从数据资产层中提取相应的资产信息,触发相关的消息机制。

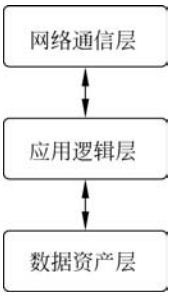

图 3.1 服务器架构图

应用逻辑层主要包括接收客户端发来的消息,调用对应模型,触发相关的消息机制。其中,消息种类包括场景类、信息类、状态类、模型类、消息类。针对不同的资产数据类型,建立对应的模型类,用于处理不同的消息请求,包括系统、场景、角色、三维模型、动画、纹理、材质、图片、文本等。数据库将对上述各类资产,建立对应的数据表,存

储相关信息。

数据资产层主要包括数字资产和数据库管理。数字资产包括用适当格式保存的场景、角色、三维模型、动画、纹理、材质、图片、文本等。数据库管理常见的为MySQL、SQLite等。

服务器技术(详情参见附录：服务器技术及Python集合)有很多，常见的有Flask、Socket、PHP、Apache、Django等。Django是一个由Python编写的开放源代码的Web重量级应用框架，拥有强大的数据库功能，自带强大的后台功能，内部封装了很多功能组件，使开发变得简单快捷。使用Django，只要很少的代码，Python程序开发人员就可以轻松地完成一个正式网站所需要的大部分内容，并进一步开发出全功能的Web服务。Django本身基于MVC模型，即Model(模型)＋View(视图)＋Controller(控制器)设计模式。MVC模式使后续对程序的修改和扩展简化，并且使程序某一部分的重复利用成为可能。Python＋Django是快速开发、设计、部署网站、Web服务器的最佳组合。

本书的案例应用采用Python 3.7＋Django 3.0服务器框架。

3.2　安装Python和Django

在Python官网下载Windows X86类型下的Pyhton 3.7安装包，如图3.2(a)所示。下载完成后，运行安装包，勾选将Python加入系统环境选项，如图3.2(b)所示，然后进行安装。安装完成后，按Win＋R组合键并输入“cmd”，打开命令提示符终端。在命令提示符窗口中输入如下命令查看是否安装成功。

```
python
```

若结果如图3.2(c)所示，则安装成功。Python的IDE有很多，包括PyCharm、Jupyter notebook、Spyder等，读者可用自己习惯的IDE去开发，本书使用的是PyCharm。

安装Django比较简单，直接用命令行即可，在终端输入以下命令：

```
pip install Django
```

如图3.3(a)所示，等待安装完成即可。安装完成后，依次输入以下命令：

(a) (b)

```
C:\Users\cp>python
Python 3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
```

(c)

图 3.2　下载 Python 安装包、安装及验证

```
python
import django
print(django.VERSION)
```

若结果为输出 Django 的版本信息，如图 3.3(b)所示，则表示 Django 安装成功。

```
C:\Users\cp>pip install Django
Collecting Django
  Downloading Django-3.0.9-py3-none-any.whl (7.5 MB)
     |████████████████████████████████| 7.5 MB 1.4 MB/s
Collecting sqlparse>=0.2.2
  Downloading sqlparse-0.3.1-py2.py3-none-any.whl (40 kB)
     |████████████████████████████████| 40 kB 2.5 MB/s
Requirement already satisfied: pytz in d:\anaconda3\lib\site-packages (from Django) (2018.7)
Collecting asgiref~=3.2
  Downloading asgiref-3.2.10-py3-none-any.whl (19 kB)
Installing collected packages: sqlparse, asgiref, Django
Successfully installed Django-3.0.9 asgiref-3.2.10 sqlparse-0.3.1
```

(a)

```
C:\Users\cp>python
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> print(django.VERSION)
(3, 0, 9, 'final', 0)
>>>
```

(b)

图 3.3　Django 的安装及验证

3.3 “排行榜”Web 服务器搭建

游戏排行榜最能体现玩家对游戏的熟练度，并激发玩家的游戏兴趣。一个很好的例子是微信小游戏“打飞机”，由于其具有分数联网排行，一经推出便吸引了众多玩家。很多玩家为了当周第一相互竞争，每天花费大量时间在游戏上，这增加了游戏的日活跃度。本书中的 Demo 也增加了一个联网排名，用于玩家的竞争比较。

在安装完 Django 之后，便可开始搭建第一个 Django 应用。新建一个名为 AR_App 的文件夹，用于存放服务器与 AR 应用的源代码。打开终端，切换到该目录下，输入以下命令：

```
django-admin startproject Server
```

该命令为在当前目录创建一个名为 Server 的 Django 项目。然后切换到 Server 目录下：

```
cd Server
```

之后输入命令行：

```
django-admin startapp ar_server
```

该命令为创建一个名为 ar_server 的应用。在创建完成之后，尝试运行这个 Django 项目。以上执行步骤及结果见图 3.4。

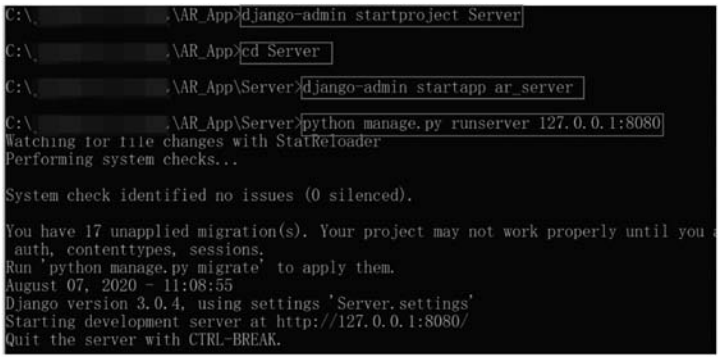

图 3.4　新建一个 Django 项目及运行

输入命令行运行项目：

```
python  manage.py runserver 127.0.0.1:8080
```

运行之后，打开浏览器，访问 http://127.0.0.1:8080/，若出现如图 3.5 所示页面，则表示 Django 项目运行成功。

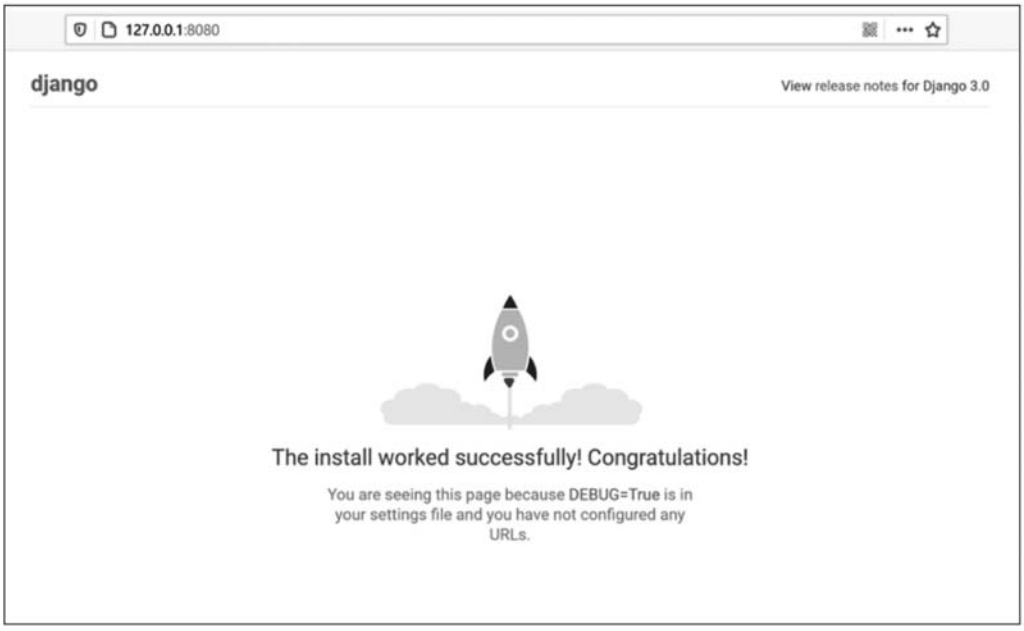

图 3.5　访问 Django 项目

创建完成之后，可以查看目录 AR_App 下已经出现一个 Django 项目，使用 PyCharm 打开 Server 目录。可以看到如图 3.6 所示的目录结构，其中各目录说明如下。

ar_server：应用的容器。

ar_server/migrations：数据库迁移文件夹。

ar_server/admin.py：后台 admin 配置文件。

ar_server/models.py：数据库模型文件。

ar_server/views.py：应用的视图函数处理文件。

Server：项目的容器。

Server/asgi.py：ASGI 兼容的 Web 服务器的入口，以便运行项目。

Server/settings.py：项目配置文件。

Server/urls.py：该 Django 项目的 URL 声明，用于管理访问地址。

Server/wsgi.py：WSGI 兼容的 Web 服务器的入口，以便运行项目。

manage.py：命令行工具，可以让开发者以各种方式与 Django 项目进行交互。

开发主要围绕 models.py、urls.py 和 views.py。

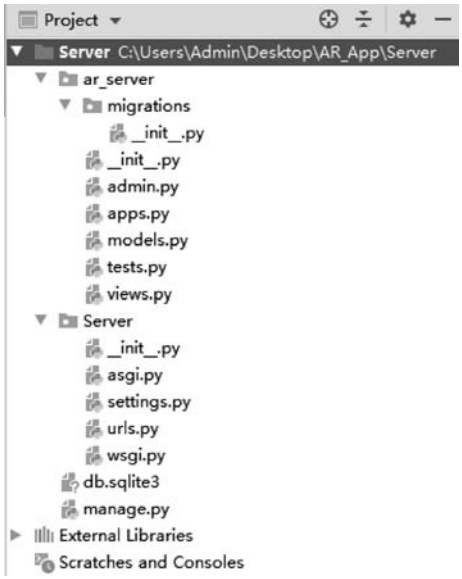

图 3.6　Django 项目目录结构

我们需要为开发的 AR 游戏应用提供一个可以存储并访问的“排行榜”服务器，为此，需要一个可存储玩家分数的数据库。Django 支持许多数据库，包括 MySQL、Oracle、PostgreSQL 等，开发者可自行选择需要的数据库，与数据库连接的操作也非常简单，只需在 Server/settings.py 文件中配置相应的数据库即可。本书使用 Django 默认支持的数据库系统 SQLite3，无须额外配置。Django 中内嵌了 ORM 框架，不需要直接面向数据库编程，而是定义模型类，通过模型类和对象完成数据表的增删改查操作。ORM 即 Object Relation Mapping，简单来说就是映射对象和数据库的关系。在 ORM 框架中，会自动将类和数据表进行映射，通过类和类对象就能操作它所对应的表格中的数据，此外，ORM 会根据开发者设计的类自动生成数据库中的表格，省去了建表的时间。

由于服务器只需要存储玩家的分数，故只需一个排名表，存储玩家姓名、分数和时

间即可,需要在 ar_server/models.py 中定义该 rank 类。

```
from django.db import models
class rank(models.Model):                          #创建一个 rank 类,用于存储玩家姓名、分数、
                                                   #时间
    name = models.CharField(max_length=11) #姓名属性,数据类型为 Char,最大长度
                                             #为 11
    score = models.IntegerField()          #分数属性,数据类型为 Integer
    time = models.DateTimeField(auto_now_add=True)    #时间属性,数据类型为
                                                      #DateTime,设置为当前时间
```

在定义 rank 类之后,需要在 ar_server/admin.py 中注册 rank 表:

```
from django.contrib import admin
from ar_server import models
admin.site.register(models.rank)        #注册 rank 表
```

Django 项目需要使用应用时,需要在 Server/settings.py 的 INSTALLED_APPS 中注册:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'ar_server',          #注册应用
]
```

在 Django 框架下,每次更改了数据库,均需要进行数据库的迁移。数据库的迁移非常简单,只需两条命令即可。在 PyCharm 的终端依次输入:

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

结果如图 3.7 所示,即完成数据库的迁移。

完成数据库的建立及迁移后,需要为服务器开发一个接口,用来接收并存储上传的玩家分数信息。在获得玩家分数信息并存储后,需要返回排行榜中分数前 10 的信息,分数相同,则时间早的优先。在 ar_server/views.py 中:

```
C:\Users\Admin\Desktop\AR_App\Server>python manage.py makemigrations
Migrations for 'ar_server':
  ar_server\migrations\0001_initial.py
    - Create model rank

C:\Users\Admin\Desktop\AR_App\Server>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, ar_server, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying ar_server.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying sessions.0001_initial... OK
```

图 3.7　数据库迁移

```
from django.shortcuts import render
from django.core import serializers
from django.http import JsonResponse
from ar_server import models
def upload_score(request):
    if request.method == 'GET':                          #传输方式为 GET
        player_name = request.GET.get('name')            #获取字段为"name"的值
        player_score = request.GET.get('score')          #获取字段为"score"的值
        models.rank.objects.create(name = player_name, score = player_score)
                                                         #添加一条数据进 rank 表
        num = models.rank.objects.count()                #查询 rank 表中数据的条数
        num = num if num < 10 else 10                    #取前 10 条,不足 10 条就取全部
        models_list = models.rank.objects.all().order_by('-score','time')[:num]
                                    #查询 rank 表中所有数据,并按 score 降序、time 升序
        rank = serializers.serialize('json', models_list)      #list 转换为 json 1
        ranks = json.loads(rank)                         #list 转换为 json 2
    return JsonResponse(ranks,safe = False)              #返回 JSON 数据
```

开发完用于接收、存储并返回玩家分数的接口后，需要为该接口注册一个可访问的地址。可在 Server/urls.py 中注册地址：

```
from django.contrib import admin
from django.urls import path
from ar_server import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path(r'upload_score',views.upload_score,name = 'upload_score'),      #注册地址
]
```

其中，第一个 upload_score 为地址声明，在响应请求时，用于 URL 的地址识别匹配。第二个 upload_score 为接口声明，在匹配到地址后，调用相应的接口。第三个 upload_score 为地址 name 声明，为 URL 取名能够使开发者在 Django 的任意地方唯一地引用它，尤其是在模板中，这个特性允许开发者只改一个文件就能全局地修改某个 URL 模式。只有第二个参数需要连接相应的接口，其余开发者可自行定义，只不过在访问、调用时和此处注册的一致即可。

而开发的基于 HUAWEI AR Engine 的 AR 第三方应用需安装在移动端使用，故服务器需要开放局域网访问，让所有设备与运行本服务器的计算机连接同一个局域网即可访问。我们需要在 Server/settings.py 中设置允许访问所有端口：

```
#ALLOWED_HOSTS = []
ALLOWED_HOSTS = ['*']      #允许所有端口访问
```

至此，服务器部分的开发全部完成，只需让其运行起来即可，在 PyCharm 终端输入：

```
python manage.py runserver 0.0.0.0:8080
```

如图 3.8 所示，在输入命令后，Django 项目成功运行。

运行 Django Web 服务器后，可通过浏览器简单测试一下。访问地址为：运行 Django 计算机的 IP 地址：8080。需要测试 upload_score 这个接口是否能正常工作，故以 GET 的 HTTP 传输协议上传 name 和 score 参数。访问地址为：172.16.13.62：8080/upload_score?name=xiaoming&score=100。

此次测试上传了用户名为 xiaoming，分数为 100 的数据，结果如图 3.9 所示，服务

器返回了一条 JSON 数据(目前数据库仅这一条数据,故只返回了 xiaoming 的数据)。读者需将访问地址中的 172.16.13.62 替换为自己运行 Django 的计算机的 IP 地址。

```
C:\Users\Admin\Desktop\AR_App\Server>python manage.py runserver 0.0.0.0:8080
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 09, 2020 - 22:31:53
Django version 3.0.6, using settings 'Server.settings'
Starting development server at http://0.0.0.0:8080/
Quit the server with CTRL-BREAK.
```

图 3.8　在局域网内启动 Django

图 3.9　上传测试

小　　结

本章简述了服务器概念及架构,并为读者介绍 Django、PHP 等主流的服务器技术。本书的案例应用 Python 3.7＋Django 3.0 服务器框架,基于此为读者介绍了 Python 和 Django 的安装及验证。在最后一步介绍了搭建“排行榜”Web 服务器的过程及运行验证过程。读者可以快速熟悉服务器的搭建,为搭建自己的 Web 服务器提供了参考。

习　　题

1. 理解服务器的概念、架构及用途，并列举经典服务器技术。

2. 使用 Python＋Django＋HTML 搭建一个可访问的 Web 服务器。

3. 使用其他服务器技术（PHP、Apache）搭建“排行榜”Web 服务器。