第 3 章

# 企业门户网站框架设计

从本章开始将以一个完整的企业门户网站为例，详细阐述基于 Python Web 的网络应用开发过程。具体地，将使用流行的 Django 框架实现相关功能。另外，考虑到项目的完整性，对 Web 前端页面设计也会同步进行阐述。本书开发的企业门户网站案例已按照第 11 章教程部署在云服务器上，读者可以访问 http://python3web.com 查看整个网站的实现效果。本书从零开始，将会逐步带领读者完成网站页面设计和后端开发，直至最终将网站部署上线。所有代码和素材均可以从本书配套资源中获取。需要注意的是，本书所开发的企业门户网站仅作为示例教程使用，用于教学需要，所有网站内容包括企业名称、新闻动态、产品等均为虚构内容，读者阅读完本书后可以根据自己的实际需求进行修改、二次开发以方便定制化扩展。

视频讲解

## 3.1 需求概述

随着全球信息化速度的不断加快，信息化技术被广泛应用到了各个方面，包括企业管理上面。信息化技术的应用不仅能够提高企业的管理效率，而且还能为员工的工作提供一定的便利，因此受到广大企业的欢迎。企业通过各种信息管理系统的建设，还能有效降低信息传递的错误率，提高信息查找速度，进而提高员工的工作效率，最终提高企业的运营管理效率。企业门户网站的建设则是加强企业与外界联系的一个重要方式，同时也是展现企业形象，提高企业社会影响力的一种有效方式。因此，企业门户网站建设在企业发展的过程中具有十分重要的作用。

本书拟开发一个科技型企业门户网站，该门户网站包括如下四方面需求。

**1. 展示企业形象**

企业门户网站通常包含企业简介、企业荣誉、主要产品、企业新闻等内容，而网站访问者通过这些内容就可以对该企业有一个初步的了解，使得企业在访问者心目中树立一个简明的形象。而企业在进行门户网站建设时应当抓住重点，突出企业的主要特色，并且还要积极

展示企业在行业内的一些业绩、专业水平、技术实力、荣誉等内容，以此来增强访问者对企业的了解，并且使其对企业形成一个良好的印象。在进行形象展示时，除了文字说明外，还应当配以相应的图片，让访问者对企业有一个更加直观的了解。

**2. 人才招聘**

企业门户网站是企业进行人才招聘的一个重要渠道。访问者通过企业门户网站可以对企业有一个充分的了解，而且门户网站展示良好的企业文化可以吸引有意向的应聘人员。企业门户网站发布的人才招聘信息具有权威性，而且招聘信息更加详细，有助于企业招募到合适的人才。

**3. 服务与支持**

为企业客户提供服务与支持是企业日常运营的一项重要内容。对于科技型企业来说，服务与支持涉及的信息内容很多，包括产品手册、驱动程序、产品开发包等，在门户网站上需要提供这些不同种类文件的管理和下载功能。另外，随着近年来人工智能的兴起，众多科技型企业开始提供基于 Web API 的人工智能服务，而这些人工智能算法需要在门户网站上给出调用示例和在线演示接口，从而吸引用户体验和使用，以进一步提高科技企业核心产品的关注度。

**4. 后台管理**

网站后台管理系统主要是用于对网站前台的信息进行管理，如文字、图片和其他日常文件的发布、更新、删除等操作。简单来说就是对网站数据进行维护，使得前台内容能够得到及时更新和调整。企业门户网站需要建设高效稳定的后台管理系统以方便管理人员查看、编辑、修改网站内容。在实际的网站部署时，企业门户网站往往部署在云服务器上，网站管理人员只需要通过互联网即可登录后台管理系统实现网站维护。交互性友好的后台管理系统可以使得网站管理人员即使没有开发经验依然可以快速地进行信息查找和修改，实现网站的稳定运维。

企业门户网站建设对企业发展起着非常重要的作用。企业门户网站作为企业展示自身形象的平台，不仅能够提高企业的社会影响力，还能帮助企业招贤纳士，提高产品的认知度，进而推动企业的快速发展。因此，本书选择科技型企业门户网站为突破口，运用 Python Web 技术实现网站的全流程开发。

根据上述要求，本书模拟一个科技型企业进行门户网站设计，企业名称为恒达科技，具体包括："首页""公司简介""新闻动态""服务支持""产品中心""科研基地""人才招聘"共 7 个模块，每个模块可能包含两至三个子模块用于进一步细分功能。各模块及其子模块结构如图 3-1 所示。

下面对每个模块需求和功能进行简要介绍。

（1）科研基地：该模块主要用于展示企业科研基地相关内容，包括科研基地的文字描述和图片等，该模块功能较为单一，仅包含一个页面，并以静态页面形式提供访问和浏览。

（2）公司简介：包含两个子模块，分别是"企业概况"和"荣誉资质"。"企业概况"用于对企业进行简短的展示，包括说明文字和图片，用以展示企业的历史、核心产品、经营理念等。"荣誉资质"子模块主要以展报形式罗列企业历年来获得的荣誉，该模块随着企业发展，需要动态地增加相关内容。因此，为了方便管理员后期编辑网站内容，需要将该页面内容与数据库进行绑定，后期可以动态地添加荣誉信息。

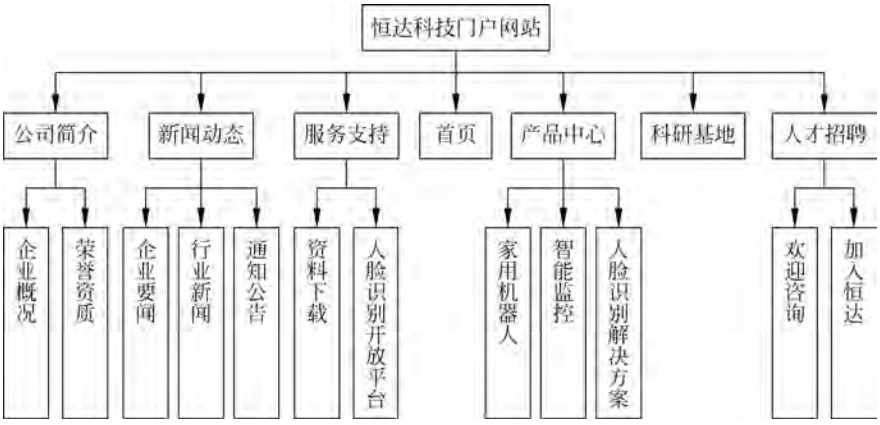


图 3-1　门户网站模块结构

（3）产品中心：包含三个子模块，分别是“家用机器人”“智能监控”和“人脸识别解决方案”，对应企业的三大主流产品。用户浏览产品时通过各个子模块链接切换可以按类别对产品进行浏览。因此，“产品中心”模块实现时需要能够从后台数据库按类型获取数据并进行页面渲染。

（4）新闻动态：“新闻动态”模块包含三个子模块，分别是“企业要闻”“行业新闻”和“通知公告”。与“产品中心”模块类似，用户可以通过各个子模块链接按新闻类型切换到指定子模块进行浏览。新闻内容以图文形式进行展现，并且管理员可以自定义图片和文字的格式。

（5）人才招聘：该模块包含两个子模块，分别是“欢迎咨询”和“加入恒达”。“欢迎咨询”子模块主要用于展示企业联系人信息以及企业地理位置，需要在实现时嵌入百度地图以方便浏览者查找位置。“加入恒达”子模块用来为企业招聘提供一个互动渠道，招聘员在网站上发布招聘职位并显示在该子模块页面上，应聘者浏览该页面并通过该页面上的表单提交个人信息。

（6）服务支持：该模块包含两个子模块，分别是“资料下载”和“人脸识别开放平台”。“资料下载”子模块用于为用户提供资料下载链接，用户通过这些链接可以下载该企业的相关产品数据，包括驱动、说明文档、SDK 开发包等。“人脸识别开放平台”子模块作为一个展示企业科技产品的模块，可以提供在线的人工智能算法支持，以 API 方式让用户体验产品。

（7）首页：该模块作为一个企业门户网站的入口模块具有非常重要的作用，需要能够全方位地展示企业各版块功能，同时需要兼具美观、清晰的特性。另外，作为一个集成模块，首页需要调用其他模块的相关数据信息，并能够对数据进行过滤和排序。

读者可以先访问 http://python3web.com 自行体验整体网站的功能，然后再进入后面的开发教程，这样可以加深对各模块内容的理解，为后面的开发做好准备。3.2 节将进入具体的开发环节。

## 3.2　搭建项目框架

本节搭建整个项目的框架，重点在于设计合理的项目文件结构。一个项目通过合理的结构设计和安排可以极大程度地提高项目整体的开发效率，减少冗余并提高项目组件的复

用性。在创建项目前先确保计算机已安装 Python 3.7 以及 Django 2.2.4。如果读者对当前开发环境不确定，可以在 Windows 系统下打开 cmd 命令行工具，输入命令 python 后，按 Enter 键来查看当前系统 Python 版本号，正常呈现效果如下所示。

```
PS C:\Users\Administrator > python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

此时，Python 已经安装成功并且进入了 Python 交互式环境。接下来输入下述命令退出 Python 的交互式环境。

```
exit()
```

然后输入下述命令检查当前环境已经安装的 Python 第三方依赖包。

```
pip list
```

检查依赖包中是否含有 Django 2.2.4，如果没有可以通过输入下述命令来进行在线安装。

```
pip install django == 2.2.4
```

确保 Python 以及 Django 均已安装完成后，打开 VS Code，在终端中通过 cd 命令切换到希望创建项目的目录位置，然后输入下述命令创建一个名为 hengDaProject 的 Django 项目。

```
django - admin startproject hengDaProject
```

依次选择 VS Code 菜单栏中的 File→Open Folder 命令，打开新创建的 hengDaProject 文件夹，导入项目内容。最后在终端中输入下述命令启动项目来检查项目是否创建成功。

```
python manage.py runserver
```

此时默认情况下开发服务器会以本机网址 http://127.0.0.1 进行部署，这是 Django 自带的轻量级开发服务器，旨在让开发人员快速查看开发效果，默认访问端口为 8000。采用开发服务器运行项目可以实现热更新，即后台对代码的修改会自动地被开发服务器检测到并且在不关闭服务器的情况下自动更新页面内容。通过浏览器访问 http://127.0.0.1:8000 查看项目运行情况。按 Ctrl+C 组合键可以停止项目运行。这里注意，尽管开发服务器能够不断地动态监测项目文件是否被修改，但是这种监测并不是每一次都能够完全捕捉到，尤其是针对 Web 前端的一些 HTML、CSS、JavaScript 等文件的修改，此时稳妥的方法是按 Ctrl+C 组合键将项目停止然后再通过命令 python manage.py runserver 重启。

## 3.2.1 文件结构设计

2.13 节中以一个在线计算器为例初步接触了一个完整的项目雏形，但是该实例仅有一个访问页面，内容和结构设计相对简单，因此只需要建立一个应用即可完成需求功能。但是在实际情况中，一个完整的网站项目拥有较多的访问页面和多种不同功能的应用模块。如果将所有的页面访问和逻辑实现全部放在一个应用下进行开发，那么会造成项目的冗余以及结构的混乱，不利于团队协作也不利于组件的复用，使得项目后期维护和扩展变得异常困难。Django 提供了一种多应用机制，即一个 Django 项目可以包含多个应用，每个应用可以实现一定的功能，或者每个应用对应部分访问内容。这样做的好处很明显，可以进一步精炼、浓缩项目功能，抽象出项目共享的模板文件和静态资源，而多个应用通过共享、继承这些文件可以提高项目开发效率，并且减少项目的冗余。

根据 3.1 节阐述的门户网站需求，可以对整个网站的功能有一个清晰的了解。针对“恒达科技”这个门户网站的需求可以发现，每个页面下基本都有 2～3 个二级子页面，比如“公司简介”下有“企业概况”和“荣誉资质”两个子页面，“产品中心”下有“家用机器人”“智能监控”和“人脸识别解决方案”3 个子页面。在这种情况下，为了清晰地表述项目结构，一种便捷有效的结构设计方法即将每一个一级页面看作一个功能应用，具体地可以分为“首页”“公司简介”“新闻动态”“产品中心”“服务支持”“科研基地”“人才招聘”共 7 个应用，这样每个应用下都只需要开发 2～3 个相似的子页面即可。采用这种方式的好处在于今后如果开发其他类似的项目可以重复利用这些功能模块，只需要在项目中简单地配置即可。接下来，按照上述结构思路为每个页面建立应用。

以“首页”模块为例，在终端中输入下述命令创建一个名为 homeApp 的应用。

```
python manage.py startapp homeApp
```

接下来按照同样方法创建其他模块对应的应用，依次输入以下命令。

```
python manage.py startapp aboutApp
python manage.py startapp newsApp
python manage.py startapp productsApp
python manage.py startapp serviceApp
python manage.py startapp scienceApp
python manage.py startapp contactApp
```

通过上述命令依次创建“公司简介”(aboutApp)、“新闻动态”(newsApp)、“产品中心”(productsApp)、“服务支持”(serviceApp)、“科研基地”(scienceApp)、“人才招聘”(contactApp)这几个功能应用。所有应用创建完成后，项目结构如图 3-2 所示。


图 3-2 一个 Django 项目集成多个应用

接下来在项目工作目录下创建 templates 文件夹，该文件夹用于存放各个应用共享的模板文件，此处一般指用于共享和继承的 HTML 模板文件。通常门户网站各个功能页面具有统一的风格，即网页头部、导航栏、尾部等内容一般是相同的，因此可以将这些相同的内容编辑成模板文件，其他功能页面在开发的过程中可以继承该模板文件，然后通过少量代码的修改即可实现页面的

复用，这种方式可以极大地提高开发效率。Django 的模板概念将会在 3.2.3 节中进行介绍。

另外，为了能够使得多个应用共享静态资源文件，在项目根目录下创建 static 文件夹，在该文件夹中分别建立 css、fonts、img、js 这四个子文件夹，分别用来存储项目共享的样式文件、字体库文件、图片文件和 JavaScript 代码文件。由于本书采用的是 Bootstrap 前端框架，因此按照 2.13 节中介绍的方法，将 Bootstrap 对应的配置文件分别导入到各个对应的项目文件夹中，配置完成后各资源文件夹目录结构如图 3-3 所示。

注意在 css 文件夹中额外添加了自定义的 style.css 文件，该文件并不是 Bootstrap 提供的文件，而是本书为了定制化 HTML 样式建立的 CSS 文件。尽管可以采用 Bootstrap 框架完成页面的布局，但是依然有很多组件需要细微地调整，此时就需要通过在 style.css 文件中编写代码覆盖 Bootstrap 的基础样式来实现。

最后，给出 hengDaProject 项目完整的文件结构图，如图 3-4 所示。


图 3-3　导入 Bootstrap 配置文件至 Django 项目


图 3-4　项目完整文件结构示意图

## 3.2.2　多级路由配置和访问

本节学习 Django 的多级路由配置和访问，即如何在大型项目中合理地设计每个访问页面对应的路由。在前面章节中已经接触过 Django 的路由配置，基本配置方法是在 urls.py 文件中通过 path()函数将路由和视图处理函数进行绑定，使得访问时能够正常解析路由并使用绑定的视图函数进行处理，基本形式如下。

```
path('home/', home, name = 'home'),
```

path()函数接收的第一个参数是相对访问路由，如果项目部署在本地 8000 端口时，此时对应的访问网址为 http://127.0.0.1:8000/home。第二个参数即指定绑定的视图函数。第三个参数用于在模板中进行逆向解析，Django 模板的使用在 3.2.3 节中将详细介绍。以上配置 path 路由的过程相对简单，适用于小规模 Django 项目。当项目规模较大、具有较多

的路由时，此时如果将所有路由依然放置在同一个 urls.py 文件中，会使得路由管理显得混乱、逻辑不清晰。一种比较好的解决方案就是将与各个应用相关的路由放置在各个应用文件夹下，每个应用单独管理一个 urls.py 文件。例如，“公司简介”模块下有一个“企业概况”页面，如果按照单层路由进行设计，其对应的路由形式为：

```
path('survey/', survey, name = 'survey')
```

如果希望将上述路由设置于 aboutApp 应用下面，那么就需要定义二级路由。一级路由指定路由所在的应用 App，在项目根路由文件 urls.py 中进行声明，形式如下。

```
path('aboutApp/', include('aboutApp.urls'))
```

path()函数第一个参数用来匹配应用对应的前缀，即如果想要访问 aboutApp 应用下的路由，在根网址后都必须加上 aboutApp 前缀。path()第二个参数用来接收由 include 指定的二级路由对应的路由文件。

二级路由可以在应用中创建一个 urls.py 文件，然后将二级路由与视图函数进行绑定即可。下面以“首页”(homeApp)和“公司简介”(aboutApp)两个应用为例，具体阐述如何配置多级路由并且实现访问。尽管前面已经创建了 7 个应用，但是所有应用并没有加载到项目中。因此，首先将 homeApp 和 aboutApp 两个应用加载到项目中来。打开 hengDaProject 文件夹下的 settings.py 文件，找到 INSTALLED_APPS 字段，在该字段末尾添加应用：

```
INSTALLED_APPS = [
    …其他应用…
    'homeApp',      #添加"首页"应用
    'aboutApp',     #添加"公司简介"应用
]
```

接下来需要实现每个应用对应的页面访问。此处为了快速测试功能，暂时不对前端进行设计，只让每个页面显示不同的字符串即可，即访问首页时页面出现“首页”字符，访问企业概况时页面出现“企业概况”字符，同样，访问荣誉资质页面时出现“荣誉资质”字符。

Django 提供了 HttpResponse()函数用来直接通过代码生成页面并返回给前端浏览器渲染。具体地，编辑 homeApp 文件夹中的 views.py 文件，代码如下。

```
from django.shortcuts import HttpResponse

def home(request):
    html = '<html><body>首页</body></html>'
    return HttpResponse(html)
```

上述代码首先从 django 包的 shortcuts 模块中导入 HttpResponse()函数，然后定义了首页路由的视图处理函数 home()。home()函数以 request 为接收参数，通过调用 HttpResponse()函数返回一个 HTML 字符串给前端进行显示。这里将 HTML 代码硬编码在 Python 代码中，以字符串的形式进行返回。

接下来对“首页”路由进行配置。尽管本书将首页也作为一个独立的应用进行设置，但是首页的路由比较特殊，一般情况下，网站的根访问路径对应网站首页，即类似访问 http://127.0.0.1:8000/即可浏览首页，无须定位到某个具体的应用路径下。因此，只需要在项目的 urls.py 文件中(hengDaProject/urls.py)找到 urlpatterns 字段然后添加首页路由即可，代码如下。

```
from homeApp.views import home

urlpatterns = [
    …其他路由…
    path('', home, name = 'home'),
]
```

配置完首页后开始配置“企业概况”和“荣誉资质”页面对应的二级路由。这两个子页面均属于“公司简介”应用模块，因此将这两个页面的路由放置在 aboutApp 下单独进行管理。

首先在项目根路由文件 urls.py 的 urlpatterns 字段中添加 aboutApp 应用对应的一级路由。

```
urlpatterns = [
    …其他路由…
    path('aboutApp/', include('aboutApp.urls')) #添加"公司简介"路由
]
```

上述配置通过 include()函数将 aboutApp 应用中的 urls.py 文件包含进来。为了能够使用 include()函数，在头部需要添加：

```
from django.conf.urls import include
```

访问时，需要在所有的二级子目录前添加 aboutApp 前缀。接下来在应用 aboutApp 中添加 urls.py 文件，然后编辑代码如下。

```
from django.urls import path
from . import views

app_name = 'aboutApp'                                    #设置应用名

urlpatterns = [
    path('survey/', views.survey, name = 'survey'),      #企业概况
    path('honor/', views.honor, name = 'honor'),         #荣誉资质
]
```

上述路由设置与普通的一级路由设置相同，将“企业概况”和“荣誉资质”对应的路由分别绑定到 survey()和 honor()函数。值得注意的是，需要在应用的 urls.py 文件中显式地设置应用名 app_name = 'aboutApp'，从而方便后续使用 Django 模板实现路由的逆向解析。

最后，为了能够访问网页内容，同样将页面文字以硬编码方式嵌入到 Python 字符串中

并通过 HttpResponse()函数返回给浏览器。编辑 aboutApp 应用下的 views.py 文件,添加代码如下。

```
from django.shortcuts import render
from django.shortcuts import HttpResponse

def survey(request):
    html = '<html><body>企业概况</body></html>'
    return HttpResponse(html)

def honor(request):
    html = '<html><body>荣誉资质</body></html>'
    return HttpResponse(html)
```

为了方便后期部署和访问,需要开放访问权限。编辑 hengDaProject 文件夹下的 settings.py 文件,修改 ALLOWED_HOSTS 字段如下。

```
ALLOWED_HOSTS = ['*',]
```

添加上述代码后,运行项目,浏览器访问 http://127.0.0.1:8000 即可查看首页内容(出现"首页"字符串),如果想要访问"企业概况"和"荣誉资质"页面,可以通过浏览器分别访问 http://127.0.0.1:8000/aboutApp/survey 和 http://127.0.0.1:8000/aboutApp/honor 查看效果。上述网址中的 aboutApp 即为在应用 urls.py 文件中设置的 app_name。survey 和 honor 对应 urls.py 文件中由 path 绑定的路由。

到这里,已经完成了"首页"和"公司简介"两个模块各页面的路由配置。按照上述步骤为其他应用也添加类似的页面访问内容,完善整个项目框架。项目根 urls 文件最终内容如下。

```
from django.contrib import admin
from django.urls import path
from django.conf.urls import include
from homeApp.views import home

urlpatterns = [
    path('admin/', admin.site.urls),                    #管理员
    path('', home, name='home'),                        #首页
    path('aboutApp/', include('aboutApp.urls')),        #公司简介
    path('contactApp/', include('contactApp.urls')),    #人才招聘
    path('newsApp/', include('newsApp.urls')),          #新闻动态
    path('productsApp/', include('productsApp.urls')),  #产品中心
    path('scienceApp/', include('scienceApp.urls')),    #科研基地
    path('serviceApp/', include('serviceApp.urls')),    #服务支持
]
```

contactApp 应用(人才招聘)的 urls.py 文件关键配置内容如下。

```
app_name = 'contactApp'

urlpatterns = [
    path('contact/', views.contact, name = 'contact'),   #欢迎咨询
    path('recruit/', views.recruit, name = 'recruit'),   #加入恒达
]
```

newsApp 应用(新闻动态)的 urls.py 文件关键配置内容如下：

```
app_name = 'newsApp'

urlpatterns = [
    path('company/', views.company, name = 'company'),       #企业要闻
    path('industry/', views.industry, name = 'industry'),    #行业新闻
    path('notice/', views.notice, name = 'notice'),          #通知公告
]
```

productsApp 应用(产品中心)的 urls.py 文件关键字段内容如下。

```
app_name = 'productsApp'

urlpatterns = [
    path('robot/', views.robot, name = 'robot'),                #家用机器人
    path('monitoring/', views.monitoring, name = 'monitoring'),#智能监控
    path('face/', views.face, name = 'face'),                   #人脸识别解决方案
]
```

serviceApp 应用(服务支持)的 urls.py 文件关键字段内容如下。

```
app_name = 'serviceApp'

urlpatterns = [
    path('download/', views.download, name = 'download'),   #资料下载
    path('platform/', views.platform, name = 'platform'),   #人脸识别开放平台
]
```

scienceApp 应用(科研基地)的 urls.py 文件关键字段内容如下。

```
app_name = 'scienceApp'

urlpatterns = [
    path('science/', views.science, name = 'science'),   #科研基地
]
```

配置完每个应用的 urls.py 文件后需要在每个应用的 views.py 文件中同步编辑视图处理函数。本章设计框架时需要编写完整的视图处理函数代码，为了方便测试功能，可以先使用 HttpResponse 返回固定的字符串显示即可。

接下来将所有新建的应用添加到项目配置文件 settings.py 中的 INSTALLED_APPS

字段，详细代码如下。

```
INSTALLED_APPS = [
    …其他应用…
    'homeApp',     #添加"首页"应用
    'aboutApp',    #添加"公司简介"应用
    'contactApp',  #添加"人才招聘"应用
    'newsApp',     #添加"新闻动态"应用
    'productsApp', #添加"产品中心"应用
    'serviceApp',  #添加"服务支持"应用
    'scienceApp',  #添加"科研基地"应用
]
```

最后，启动项目，按照下述网址逐个通过浏览器进行访问并检查是否有错误。

(1) 首页：http://127.0.0.1:8000/。

(2) 企业概况：http://127.0.0.1:8000/aboutApp/survey/。

(3) 荣誉资质：http://127.0.0.1:8000/aboutApp/honor/。

(4) 欢迎咨询：http://127.0.0.1:8000/contactApp/contact/。

(5) 加入恒达：http://127.0.0.1:8000/contactApp/recruit/。

(6) 企业要闻：http://127.0.0.1:8000/newsApp/company/。

(7) 行业新闻：http://127.0.0.1:8000/newsApp/industry/。

(8) 通知公告：http://127.0.0.1:8000/newsApp/notice/。

(9) 家用机器人：http://127.0.0.1:8000/productsApp/robot/。

(10) 智能监控：http://127.0.0.1:8000/productsApp/monitoring/。

(11) 人脸识别解决方案：http://127.0.0.1:8000/productsApp/face/。

(12) 资料下载：http://127.0.0.1:8000/serviceApp/download/。

(13) 人脸识别开放平台：http://127.0.0.1:8000/serviceApp/platform/。

(14) 科研基地：http://127.0.0.1:8000/scienceApp/science/。

通过本节 Django 路由的学习，相信读者已经掌握如何在实际网站项目中设置和管理二级路由。二级路由的使用可以使得项目结构进一步细化，提高了代码结构的清晰度，同时有利于后期组件的重复使用。

## 3.2.3 Django 模板概述

在 3.2.2 节中，为了快速生成 HTML，将 HTML 对应的内容直接硬编码到 Python 代码中并返回，这种方法比较直观，可以快速设计项目结构，但是直接将 HTML 内容编码到 Python 代码中这本身并不是一种良好的实现方式，前端设计和后端开发存在紧耦合，当页面内容更改时需要手动地对 Python 代码进行修改，影响开发效率。实际情况中，后台 Python 代码的编写和前端 HTML 编码设计是两项不同的工作，一般是由两个团队分别完成，中间通过设计好的接口协议进行交互。为了有效解决这个问题，Django 提供了模板机制，仅需少量的改动即可让后端服务器调用并渲染前端 HTML 页面。

Django 提供的模板通常用来处理 HTML，其本质是一种文本。简单来说，Django 在普

通的 HTML 文件中嵌入一些特殊意义的字符，这时候该 HTML 文件就称为模板，而这些特殊字符可以归纳为两种：变量和模板标签。

变量由两个大括号括起来，一般形式为：

```
{{ name }}
```

变量提供了一种页面内容动态生成的方法，使得后端服务器在渲染 HTML 页面时可以动态地将变量值插入到 HTML 中。

模板标签由大括号和百分号包围，一般形式为：

```
{ % if today_is_weekend % }
    <p>Welcome to the weekend!</p>
{ % else % }
    <p>Get back to work.</p>
{ % endif % }
```

以上模板语句类似于 Python 的 if 条件判断，可以在 HTML 中执行逻辑流程控制。尽管有些情况下 Django 的模板标签和 Python 语法提供的功能相同，但是本质上不一样。Django 提供的模板标签相对较少，并没有 Python 提供的丰富。从项目前后端分离的角度考虑，Django 并不希望使用过多的模板标签来破坏 HTML 本身的完整性，只是提供了必要的控制语句在关键处进行改写实现控制即可。

尽管 Django 提供了方便的模板机制使得服务器可以渲染 HTML，但是并不严格要求必须使用 Django 模板。本质上来看，采用 Django 的模板依然破坏了 HTML 原来的内容。每次前端完成设计后都需要按照 Django 模板语法进行修改，否则无法准确地渲染 HTML 页面。在第 12 章中将会阐述基于 RESTful API 的概念，采用这种方式可以完全脱离 Django 模板的束缚，真正做到前后端分离。但是不可否认，在项目规模不是很庞大并且没有充足的前后端开发人员的情况下，使用 Django 模板会更加便捷。

## 3.2.4 基于 Django 模板的静态资源配置

本节基于 Django 的模板机制将对各个应用模块的静态资源进一步进行配置，最终实现通过首页上的 Bootstrap 导航条能够跳转到各个子页面。

首先，对“首页”(homeApp)模块进行页面编辑，通过使用 Bootstrap 的导航条组件在首页中添加跳转到各个子页面的链接。由于需要使用 Bootstrap 框架进行页面设计，因此先确保在项目根目录下的 static 文件夹中已经将 Bootstrap 需要的 JS、CSS、FONTS 等配置文件正确导入，具体导入方法可以参考 2.12 节和 2.13 节。

本节不再采用 Python 硬编码方式生成 HTML，而是直接使用 Django 提供的 render() 函数进行 HTML 模板渲染。首先在 homeApp 文件夹下新建 templates 文件夹(注意该文件夹名字必须为 templates)用来存放 HTML 模板文件。然后在该 templates 文件夹下新建 home.html 文件，该文件即为首页页面文件。编辑 home.html，在头部引用 Bootstrap 对应的配置文件，代码如下。

```
{ % load staticfiles % }
<!DOCTYPE html>
<html lang = "zh-cn">

<head>
    <meta charset = "utf-8">
    <meta http-equiv = "X-UA-Compatible" content = "IE=edge">
    <meta name = "viewport" content = "width=device-width, initial-scale=1">
    <title>恒达科技(教学示例网站)</title>
    <link href = "{ % static 'css/bootstrap.css' % }" rel = "stylesheet">
    <link href = "{ % static 'css/style.css' % }" rel = "stylesheet">
    <script src = "{ % static 'js/jquery.min.js' % }"></script>
    <script src = "{ % static 'js/bootstrap.min.js' % }"></script>
</head>

<body>
</body>

</html>
```

上述代码首先引入了{% load staticfiles %}语句,这里使用了 Django 模板提供的 staticfiles 标签,通过导入该标签可以在页面中通过关键字 static 定位到项目的静态资源,最终所有静态文件资源的引用都可以采用类似"{% static 'css/bootstrap.css' %}"这种形式。上述引用方式是 Django 模板最核心的部分,后面会大量使用这种静态资源引用方式,读者需要牢牢掌握该方法。

接下来在<body>标签部分,引用 Bootstrap 提供的现成的导航栏组件 nav 来包含各子页面链接,详细代码如下。

```
<!-- 导航条 -->
<nav class = "navbar navbar-default" role = "navigation">
    <div class = "container">
        <div class = "navbar-header">
            <button type = "button" class = "navbar-toggle collapsed" data-toggle =
"collapse" data-target = "#bs-example" aria-expanded = "false">
                <span>导航栏</span>
            </button>
        </div>
        <div class = "collapse navbar-collapse" id = "bs-example">
            <ul class = "nav navbar-nav" style = "width:100%;">
                <li class = "active nav-top">
                    <a href = "{ % url 'home' % }">首页</a>
                </li>
                <li class = "dropdown nav-top">
                    <a href = "#" class = "dropdown-toggle on" data-toggle = "dropdown">
                        公司简介</a>
                    <ul class = "dropdown-menu">
                        <li><a href = "{ % url 'aboutApp:survey' % }">企业概况</a></li>
```

```
                        <li><a href="{% url 'aboutApp:honor' %}">荣誉资质</a></li>
                    </ul>
                </li>
                <li class="dropdown nav-top">
                    <a href="#" class="dropdown-toggle on" data-toggle="dropdown">
                        新闻动态</a>
                    <ul class="dropdown-menu">
                        <li><a href="{% url 'newsApp:company' %}">企业要闻</a></li>
                        <li><a href="{% url 'newsApp:industry' %}">行业新闻</a></li>
                        <li><a href="{% url 'newsApp:notice' %}">通知公告</a></li>
                    </ul>
                </li>
                <li class="dropdown nav-top">
                    <a href="#" class="dropdown-toggle on" data-toggle="dropdown">
                        产品中心</a>
                    <ul class="dropdown-menu">
                        <li><a href="{% url 'productsApp:robot' %}">
                                家用机器人</a></li>
                        <li><a href="{% url 'productsApp:monitoring' %}">
                                智能监控</a></li>
                        <li><a href="{% url 'productsApp:face' %}">
                                人脸识别解决方案</a></li>
                    </ul>
                </li>
                <li class="dropdown nav-top">
                    <a href="#" class="dropdown-toggle on" data-toggle="dropdown">
                        服务支持</a>
                    <ul class="dropdown-menu">
                        <li><a href="{% url 'serviceApp:download' %}">
                                资料下载</a></li>
                        <li><a href="{% url 'serviceApp:platform' %}">
                                人脸识别开放平台</a></li>
                    </ul>
                </li>
                <li class="nav-top">
                    <a href="{% url 'scienceApp:science' %}">科研基地</a>
                </li>
                <li class="dropdown nav-top">
                    <a href="#" class="dropdown-toggle on" data-toggle="dropdown">
人才招聘</a>
                    <ul class="dropdown-menu">
                        <li><a href="{% url 'contactApp:contact' %}">欢迎咨询</a></li>
                        <li><a href="{% url 'contactApp:recruit' %}">加入恒达</a></li>
                    </ul>
                </li>
            </ul>
        </div>
    </div>
</nav>
```

上述代码中通过使用类 class="dropdown nav-top"来调用导航栏的下拉菜单。注意每个链接<a>标签部分，其中，href 属性使用了 Django 提供的模板标签{% url '逆向路径' %}来逆向寻找访问路径，寻找方式采用“应用名:函数名”的形式，应用名如果没有可以不使用，例如首页仅采用下面这种无应用名的形式：

```
<a href="{% url 'home' %}">首页</a>
```

具体的反向解析形式与之前配置的 url 路由相对应。例如，配置“企业概况”页面时，在该应用的 urls.py 文件中定义了应用名 app_name = 'aboutApp'，并且绑定了二级路由：

```
path('survey/', views.survey, name='survey'), #企业概况
```

上述路由定义了函数名为 survey，因此在模板中最终的反向解析形式为：

```
"{% url 'aboutApp:survey' %}"
```

在 HTML 中使用这种逆向网址解析的方式主要是为了方便变换根访问路径。例如，在开发服务器上可以通过访问 http://127.0.0.1:8000/aboutApp/survey/ 来浏览企业概况页面，但是当将整个项目部署到生产服务器上时就需要全局去修改所有的访问路径，这显然不方便。因此比较好的方式就是通过为每个路由设置名字，然后采用逆向解析的方式根据名字去动态地查找当前对应的访问网址。

最后，为了能够正常访问首页，需要修改 homeApp 应用下 views.py 文件中的 home 处理函数，修改代码如下。

```
def home(request):
    return render(request, 'home.html')
```

另外，由于新建的共享 static 资源文件夹是在项目根路径下，需要告诉 Django 模板当前共享的静态资源路径位置。打开项目配置文件夹 hengDaProject 下的 settings.py 文件，在文件末尾添加静态资源路径设置：

```
STATICFILES_DIRS = (
    os.path.join(BASE_DIR, "static"),
)
```

运行项目，其效果如图 3-5 所示。单击首页页面各链接可以正常跳转到指定的子页面，单击浏览器返回按钮可以回退到首页。


图 3-5　首页导航栏初始效果

# 小结

本章介绍了实战项目的基本需求和最终演示效果。根据项目需求设计了项目的整体框架，包括项目文件结构、多级路由配置和 Django 模板。通过本章内容的学习，读者可以学习到实际项目开发过程中的结构设计，能够对项目的前后端逻辑有一个清晰的认识。本章作为实战项目的先导篇，为后续章节的子模块开发奠定了基础，后续各模块的开发将在现有框架基础上逐步深入实现。