前后端分离开发入门

在传统的Web应用开发中，大多数的程序员会将浏览器作为前后端的分界线。将浏览器中为用户进行页面展示的部分称之为前端，而将运行在服务器，为前端提供业务逻辑和数据准备的所有代码统称为后端。所谓前后端分离的开发，就是前后端工程师约定好数据交互接口，并行的进行开发和测试，后端只提供数据，不负责将数据渲染到页面上，前端通过HTTP请求获取数据并负责将数据渲染到页面上，这个工作是交给浏览器中的JavaScript代码来完成。

使用前后端分离开发有诸多的好处，下面我们简要的说下这些好处：

1. **提升开发效率。**前后端分离以后，可以实现前后端代码的解耦，只要前后端沟通约定好应用所需接口以及接口参数，便可以开始并行开发，无需等待对方的开发工作结束。在这种情况下，前后端工程师都可以只专注于自己的开发工作，有助于打造出更好的团队。除此之外，在前后端分离的开发模式下，即使需求发生变更，只要接口与数据格式不变，后端开发人员就不需要修改代码，只要前端进行变动即可。
2. **增强代码的可维护性。**前后端分离后，应用的代码不再是前后端混合，只有在运行期才会有调用依赖关系，这样的话维护代码的工作将变得轻松愉快很多，再不会牵一发而动全身。当你的代码变得简明且整洁时，代码的可读性和可维护性都会有质的提升。
3. **支持多终端和服务化架构。**前后端分离后，同一套数据接口可以为不同的终端提供服务，更有助于打造多终端应用；此外，由于后端提供的接口之间可以通过HTTP(S)进行调用，有助于打造服务化架构（包括微服务）。

接下来我们就用前后端分离的方式来改写之前的投票应用。

返回JSON格式的数据

刚才说过，在前后端分离的开发模式下，后端需要为前端提供数据接口，这些接口通常返回JSON格式的数据。在Django项目中，我们可以先将对象处理成字典，然后就可以利用Django封装的`JsonResponse`向浏览器返回JSON格式的数据，具体的做法如下所示。

```
def show_subjects(request):
    queryset = Subject.objects.all()
    subjects = []
    for subject in queryset:
        subjects.append({
            'no': subject.no,
            'name': subject.name,
            'intro': subject.intro,
            'isHot': subject.is_hot
        })
    return JsonResponse(subjects, safe=False)
```

上面的代码中，我们通过循环遍历查询学科得到的`QuerySet`对象，将每个学科的数据处理成一个字典，在将字典保存在名为`subjects`的列表容器中，最后利用`JsonResponse`完成对列表的序列化，向浏览器返回JSON格式的数据。由于`JsonResponse`序列化的是一个列表而不是字典，所以需要指定`safe`参数的值为`False`才能完成对`subjects`的序列化，否则会产生`TypeError`异常。

可能大家已经发现了，自己写代码将一个对象转成字典是比较麻烦的，如果对象的属性很多而且某些属性又关联到一个比较复杂的对象时，情况会变得更加糟糕。为此我们可以使用一个名为bpmappers的三方库来简化将对象转成字典的操作，这个三方库本身也提供了对Django框架的支持。

安装三方库bpmappers。

```
pip install bpmappers
```

编写映射器（实现对象到字典转换）。

```
from bpmappers.djangomodel import ModelMapper

from poll2.models import Subject


class SubjectMapper(ModelMapper):

    class Meta:
        model = Subject
```

修改视图函数。

```
def show_subjects(request):
    queryset = Subject.objects.all()
    subjects = []
    for subject in queryset:
        subjects.append(SubjectMapper(subject).as_dict())
    return JsonResponse(subjects, safe=False)
```

配置URL映射，然后访问该接口，可以得到如下所示的JSON格式数据。

```
[
    {
        "no": 101,
        "name": "Python全栈+人工智能",
        "intro": "Python是一种计算机程序设计语言。是一种面向对象的动态类型语言，最初被设计用于编写自动化脚本(shell)，随着版本的不断更新和语言新功能的添加，越来越多被用于独立的、大型项目的开发。",
        "create_date": "2017-08-01",
        "is_hot": true
    },
    // 此处省略下面的内容
]
```

如果不希望在JSON数据中显示学科的成立时间，我们可以在映射器中排除`create_date`属性；如果希望将是否为热门学科对应的键取名为`isHot`（默认的名字是`is_hot`），也可以通过修改映射器来做到。具体的做法如下所示：

```
from bpmappers import RawField
from bpmappers.djangomodel import ModelMapper

from poll2.models import Subject


class SubjectMapper(ModelMapper):
    isHot = RawField('is_hot')

    class Meta:
        model = Subject
        exclude = ('create_date', 'is_hot')
```

再次查看学科接口返回的JSON数据。

```
[
    {
        "no": 101,
        "name": "Python全栈+人工智能",
        "intro": "Python是一种计算机程序设计语言。是一种面向对象的动态类型语言，最初被设计用于编写自动化脚本(shell)，随着版本的不断更新和语言新功能的添加，越来越多被用于独立的、大型项目的开发。",
        "isHot": true
    },
    // 此处省略下面的内容
]
```

关于bpmappers详细的使用指南，请参考它的官方文档，这个官方文档是用日语书写的，可以使用浏览器的翻译功能将它翻译成你熟悉的语言即可。

使用Vue.js渲染页面

关于Vue.js的知识，我们在第21天到第30天的内容中已经介绍过了，这里我们不再进行赘述。如果希望全面的了解和学习Vue.js，建议阅读它的官方教程或者在YouTube上搜索Vue.js的新手教程（Crash Course）进行学习。

重新改写subjects.html页面，使用Vue.js来渲染页面。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>学科</title>
</head>
<body>
    <h1>所有学科</h1>
    <hr>
    <div id="app">
        <div v-for="subject in subjects">
            <h3>
                <a :href="getTeachersHref(subject.no)">{{ subject.name }}</a>
                <img v-if="subject.isHot" src="/static/images/hot.png"
width="32">
            </h3>
            <p>{{ subject.intro }}</p>
        </div>
    </div>
    <script src="https://cdn.bootcss.com/vue/2.6.10/vue.min.js"></script>
    <script>
        const app = new Vue({
            el: '#app',
            data: {
                subjects: []
            },
            created() {
                fetch('/subjects/')
                    .then(resp => resp.json())
                    .then(json => this.subjects = json)
            },
```

```
            methods: {
                getTeachersHref(sno) {
                    return `/static/teachers.html/?sno=${sno}`
                }
            }
        })
    </script>
</body>
</html>
```

前后端分离的开发需要将前端页面作为静态资源进行部署，项目实际上线的时候，我们会对整个Web应用进行动静分离，静态资源通过Nginx或Apache服务器进行部署，生成动态内容的Python程序部署在uWSGI或者Gunicorn服务器上，对动态内容的请求由Nginx或Apache路由到uWSGI或Gunicorn服务器上。

在开发阶段，我们通常会使用Django自带的测试服务器，如果要尝试前后端分离，可以先将静态页面放在之前创建的放静态资源的目录下，具体的做法可以参考项目完整代码。