全栈“食”代：Django + Nuxt 实现美食分享网站（下）

一只图雀 2020-04-05 阅读 12 分钟

在上篇中，我们分别用 Django 和 Nuxt 实现了后端和前端的雏形。在这一部分，我们将实现前后端之间的通信，使得前端可以从后端获取数据，并且将进一步丰富网站的功能。

> 本文所涉及的源代码都放在了 Github 上，如果您觉得我们写得还不错，希望您能给❤️**这篇文章点赞+Github仓库加星**❤️哦~ *本文代码改编自 Scotch*。

从服务器获取数据

在这一部分，我们将真正实现一个全栈应用——让前端能够向后端发起请求，从而获取想要的数据。

配置 Django 的静态文件服务

首先我们要配置一下 Django 服务器，使前端能够访问其静态文件。调整 api/api/urls.py 文件如下：

```
# ...
from django.conf import settings
```

```
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('core.urls')),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

> **注意**
>
> 这样配置静态文件路由的方式仅应当在**开发环境**中使用。在生产环境中（settings.py 中的 `DEBUG` 设为 `False` 时），静态文件路由将自动失效（因为 Django 并不适合作为静态文件服务器，应该选用类似 Nginx 之类的服务器，在后续教程中我们将更深入地讨论）。

实现前端的数据请求功能

在客户端，我们先要对 Nuxt 进行全局配置。Nuxt 包括 axios 包，这是一个非常出色的基于 Promise 的 HTTP 请求库。在 nuxt.config.js 中的 `axios` 一项中添加 Django 服务器的 URL：

```
export default {
  // ...

  /*
  ** Axios module configuration
  ** See https://axios.nuxtjs.org/options
  */
  axios: {
    baseURL: 'http://localhost:8000/api',
  },

  // ...
}
```

将食谱列表页面中暂时填充的假数据删去，通过 `asyncData` 方法获取数据。由于我们之前配置好了 axios，所以 `asyncData` 函数可以获取到 `$axios` 对象用于发起 HTTP 请求。我们实现页面加载的数据获取以及 `deleteRecipe` 事件，代码如下：

```
<template>
  <main class="container mt-5">
    <div class="row">
      <div class="col-12 text-right mb-4">
        <div class="d-flex justify-content-between">
          <h3>吃货天堂</h3>
          <nuxt-link to="/recipes/add" class="btn btn-info">添加
```

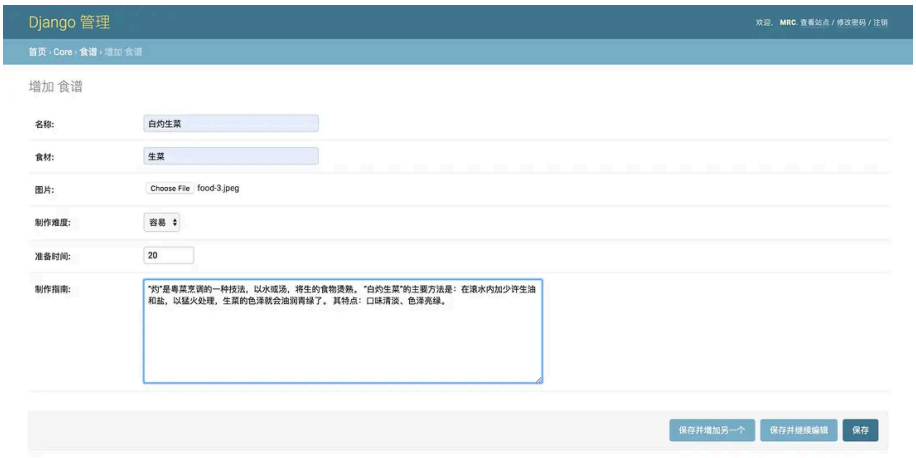
```
        </div>
      </div>
      <template v-for="recipe in recipes">
        <div :key="recipe.id" class="col-lg-3 col-md-4 col-sm-6
          <recipe-card :onDelete="deleteRecipe" :recipe="recipe
        </div>
      </template>
    </div>
  </main>
</template>

<script>
import RecipeCard from "~/components/RecipeCard.vue";

export default {
  head() {
    return {
      title: "食谱列表"
```

实现食谱详情页面

我们进一步实现食谱详情页面。在 pages/recipes 目录中创建 _id 目录，在其中添加 index.vue 文件，代码如下：

```
<template>
  <main class="container my-5">
    <div class="row">
      <div class="col-12 text-center my-3">
        <h2 class="mb-3 display-4 text-uppercase">{{ recipe.nam
      </div>
      <div class="col-md-6 mb-4">
        <img
          class="img-fluid"
          style="width: 400px; border-radius: 10px; box-shadow:
          :src="recipe.picture"
          alt
        >
      </div>
      <div class="col-md-6">
        <div class="recipe-details">
          <h4>食材</h4>
          <p>{{ recipe.ingredients }}</p>
          <h4>准备时间 ⏱</h4>
          <p>{{ recipe.prep_time }} mins</p>
          <h4>制作难度</h4>
          <p>{{ recipe.difficulty }}</p>
          <h4>制作指南</h4>
          <textarea class="form-control" rows="10" v-html="reci
        </div>
```

为了测试前端页面能否真正从后端获取数据，我们先要在后端数据库中添加一些数据，而这对 Django 来说就非常方便了。进入 api 目录，运行 `python manage.py runserver` 打开服务器，然后进入后台管理页面（http://localhost:8000/admin），添加一些数据：



再运行前端页面，可以看到我们刚刚在 Django 后台管理中添加的项目：

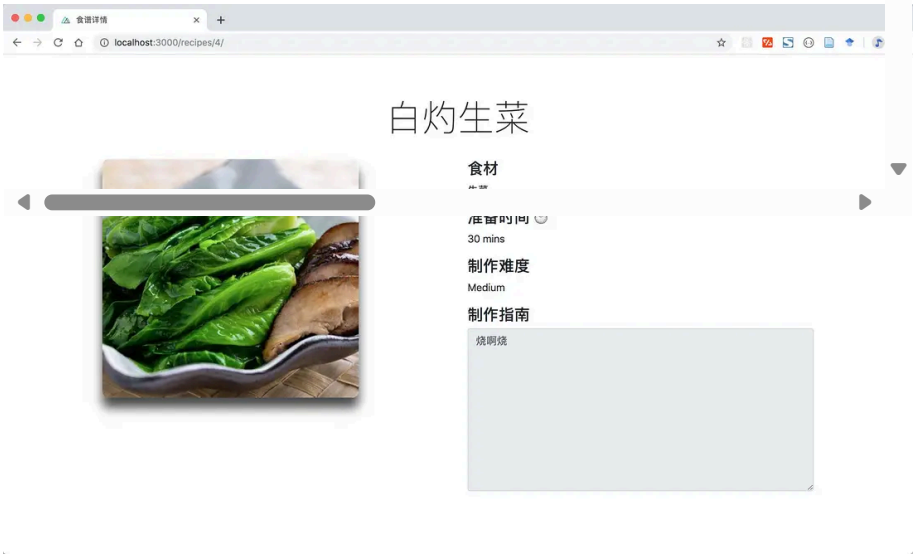
实现食谱的编辑和创建页面

有了前面的铺垫，实现食谱的添加和删除也基本上是按部就班了。我们在 pages/recipes/_id 中实现 `edit.vue`（食谱编辑页面），代码如下：

```
<template>
  <main class="container my-5">
    <div class="row">
      <div class="col-12 text-center my-3">
        <h2 class="mb-3 display-4 text-uppercase">{{ recipe.nam
      </div>
      <div class="col-md-6 mb-4">
        <img v-if="!preview" class="img-fluid" style="width: 40
```

```
      <img v-else class="img-fluid" style="width: 400px; bord
    </div>
    <div class="col-md-4">
      <form @submit.prevent="submitRecipe">
        <div class="form-group">
          <label for>Recipe Name</label>
          <input type="text" class="form-control" v-model="re
        </div>
        <div class="form-group">
          <label for>Ingredients</label>
          <input type="text" v-model="recipe.ingredients" cla
        </div>
        <div class="form-group">
          <label for>Food picture</label>
          <input type="file" @change="onFileChange">
        </div>
        <div class="row">
```

实现之后的页面如下：

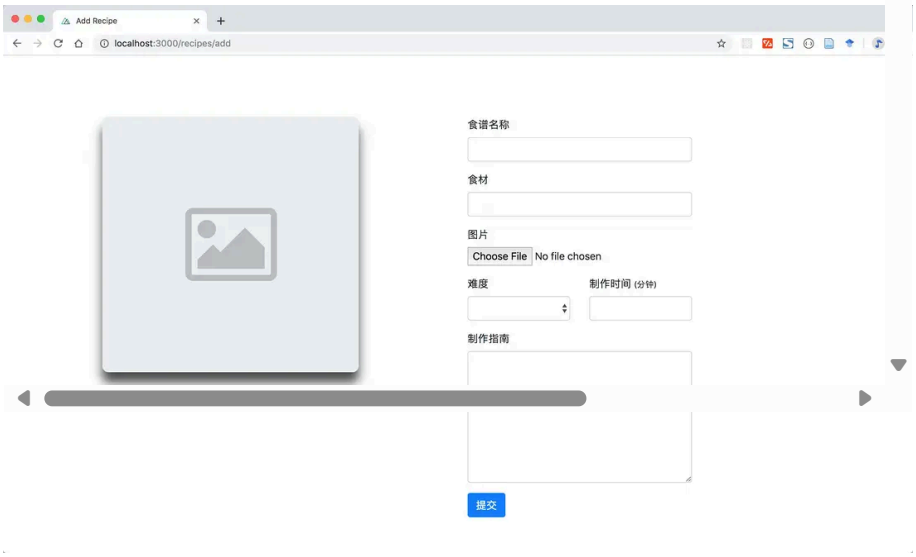


继续在 pages/recipes/_id 中实现 `add.vue` （创建食谱页面）如下：

```
<template>
  <main class="container my-5">
    <div class="row">
      <div class="col-12 text-center my-3">
        <h2 class="mb-3 display-4 text-uppercase">{{ recipe.nam
      </div>
      <div class="col-md-6 mb-4">
        <img
          v-if="preview"
          class="img-fluid"
          style="width: 400px; border-radius: 10px; box-shadow:
          :src="preview"
          alt
```

```
      >
      <img
        v-else
        class="img-fluid"
        style="width: 400px; border-radius: 10px; box-shadow:
        src="@/static/images/placeholder.png"
      >
    </div>
    <div class="col-md-4">
      <form @submit.prevent="submitRecipe">
        <div class="form-group">
          <label for>食谱名称</label>
```

实现的页面如下：



一点强迫症：全局页面跳转效果

在这一节中，我们将演示如何在 Nuxt 中添加全局样式文件，来实现前端页面之间的跳转效果。

首先在 assets 目录中创建 css 目录，并在其中添加 transition.css 文件，代码如下：

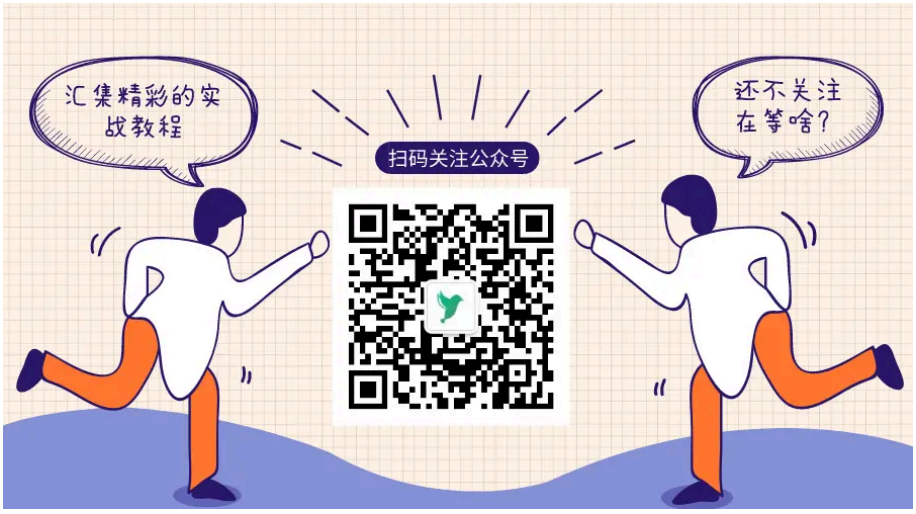
```
.page-enter-active,
.page-leave-active {
  transition: opacity .3s ease;
}

.page-enter,
.page-leave-to {
  opacity: 0;
}
```

在 Nuxt 配置文件中将刚才写的 transition.css 中添加到全局 CSS 中：

```
export default {
  // ...

  /*
  ** Global CSS
  */
  css: [
    '~/assets/css/transition.css',
  ],

  // ...
}
```

欧耶，一个具有完整增删改查功能、实现了前后端分离的美食分享网站就完成了！

> 想要学习更多精彩的实战技术教程？来图雀社区逛逛吧。
> 本文所涉及的源代码都放在了 Github 上，如果您觉得我们写得还不错，希望您能给❤️**这篇文章点赞+Github仓库加星**❤️哦~ *本文代码改编自 Scotch*。



vue.js nuxt.js python django

阅读 1.4k • 发布于 2020-04-05

一只图雀

863 声望 • 1.2k 粉丝

我们图雀社区是一个供大家分享用 Tuture 写作工具撰写教程的一个平台。在这里，读者们可以尽情享受高质量的实战教...

关注作者

« 上一篇
全栈“食”代：用 Django + Nuxt 实...

下一篇 »
类型即正义：TypeScript 从入门到...

引用和评论

推荐阅读

Taro 小程序开发大型实战（九）：使用 Authing 打造具有微信登录的企业级用户系统
一只图雀 • 赞 1 • 阅读 4.4k

vxe-table vue CRUD Table
abc26296 • 赞 33 • 阅读 21.9k • 评论 11

Vxe UI vxe-table 基本使用
abc26296 • 赞 32 • 阅读 39.8k • 评论 31

Vxe UI vxe-grid 的基本使用
abc26296 • 赞 17 • 阅读 11.8k • 评论 5

使用vue-grid-layout完成桌面拖拽布局功能
YOLO_Y • 赞 12 • 阅读 47.1k • 评论 35

【记录】Vue 中使用海康视频插件（监控）
九霄 • 赞 8 • 阅读 9.7k • 评论 18

文件导出
热饭班长 • 赞 8 • 阅读 3k

0 条评论

得票 最新

撰写评论 ...

提交评论

评论支持部分 Markdown 语法：`**粗体** _斜体_ [链接](http://example.com) `代码` - 列表 > 引用`。你还可以使用 @ 来通知其他用户。