微信小程序开发

网站示例

搭建个人站点

- 这部分内容通过搭建个人站点来展示前后端通信的过程。
- 搭建个人站点需要设计一套目录结构。
- 在前端页面部分，使用了foundation作为UI框架，可以把以下代码放在head标签中。

```
<link href="https://cdn.bootcss.com/foundation/6.5.3/css/foundation.min.css" rel="stylesheet">
```

设计目录结构

- images/　　图片资源所在目录。
- pages/　　html页面所在目录。
- errorpages/　　错误页面目录，比如404，500等页面。
- app.js　　应用程序文件。
- static/　　静态资源目录，存放css和js文件。
- apifile/　　API读取的文件目录，可以动态展示在页面上。

设计方案

- 在app.js中会设置站点的路由，包括了两部分：页面和接口。
- 页面数据返回后，会请求接口，返回动态数据。
- 页面中还可能引用css和js文件，所以要有一个加载页面的处理函数和获取css以及js静态文件的函数。

获取页面数据

- 获取的html页面文件在pages目录，而发生错误时，需要从errorpages读取对应的文件：404.html、500.html等。
- 通过一个函数实现html文件的读取，而在请求函数中直接调用页面获取的函数即可。

获取页面数据

```
//缓存页面数据
var _pageCache = {};
async function loadPage(pagename, pagedir = './pages') {
    if (_pageCache[pagename]) {
        return _pageCache[pagename];
    }

    var pagedata = await funcs.readFile(pagedir + '/' + pagename + '.html');
    _pageCache[pagename] = pagedata;

    return _pageCache[pagename];
}
```

页面路由

- 页面路由采用非常简单的形式：/:name。
- name是参数形式，实际是pages目录中文件的名字，比如/home会显示pages/home.html文件中的内容。
- 根目录/也会显示pages/home.html目录中的内容，这样用户直接输入域名访问也会显示主页。

页面路由实现

```
router.get('/', async c => {
    try {
        c.res.body = await loadPage('home');
    } catch (err) {
        c.res.body = await loadPage('404', 'errorpages');
        c.res.status(404);
    }
});

router.get('/:name', async c => {
    try {
        c.res.body = await loadPage(c.param.name);
    } catch (err) {
        c.res.body = await loadPage('404', 'errorpages');
        c.res.status(404);
    }
});
```

获取css和js静态文件

- 页面中需要引入css和js静态文件，这些文件放在static目录中。
- 页面中引入的链接是需要提供的，而我们的需求是，在static目录中只要存在这样的文件就可以引入，比如：

  <link href="/static/css/main.css" rel="stylesheet">

  则会向后台发送请求，而后台通过/static/css/main.css就可以知道文件在./static/css/main.css

获取静态文件

```
var _staticCache = {};
async function loadStatic(filename) {
    if (_staticCache[filename]) {
        return _staticCache[filename];
    }

    var data = await funcs.readFile('./static/'+filename);
    _staticCache[filename] = data;

    return _staticCache[filename];
}
```

静态文件路由实现

```
router.get('/static/*', async c => {
    try {
        let filename = c.param.starPath;
        let content_type = '';
        if (filename.indexOf('.css') > 0) {
            content_type = 'text/css';
        } else if (filename.indexOf('.js') > 0) {
            content_type = 'text/javascript';
        }
        c.res.setHeader('content-type', content_type);
        let data = await loadStatic(filename);
        c.res.setHeader('content-length', data.length);
        c.res.body = data;
    } catch (err) {
        console.log(err);
        c.res.status(404);
    }
});
```

路由*表示所有路径，这种形式，/static/后可以是任何路径参数，参数值被解析到c.param.starPath。

- 通过判断文件名称中是否存在.css或.js决定内容类型。
- 这种方式快速简单，只要不是类似于'a.css.js'这种形式命名文件就可以正常运行。

图片资源

- 图片资源通过另一个路由获取：/image/:name。
- 这个是带参数的路由，其中name因为是:开头，所以是被作为参数解析的。
- 访问这个路径会去images目录中读取相应的图片并返回。

获取图片的路由实现

```
router.get('/image/:name', async c => {
    let imgfile = `./images/${decodeURIComponent(c.param.name)}`;
    try {
      let content_type = '';
      let extname = c.helper.extName(imgfile);
      switch (extname.toLowerCase()) {
        case '.png':
          content_type = 'image/png'; break;
        case '.jpg':
        case '.jpeg':
          content_type = 'image/jpeg'; break;
        case '.gif':
          content_type = 'image/gif'; break;
        default:;
      }
      c.res.setHeader('content-type', content_type);
      let data = await funcs.readFile(imgfile, 'binary');
      c.res.setHeader('content-length', data.length);
      c.res.encoding = 'binary';
      c.res.body = data;
    } catch (err) {
      c.res.status(404);
    }
});
```