TSOOP DAY02

VueCLI项目的上线流程

前后端分离项目需要上线的项目有两个：

1. nodejs+mysql+express写的后端项目，提供后台接口。

   访问：`https://web.codeboy.com/bmdapi/xxx` 就可以访问后台接口。

2. VueCLI、Vue2、ElementUI、Axios等写的前端项目，提供静态页面。

   访问：`https://web.codeboy.cn/home/index` 就可以看到首页。该页面将自动通过axios向`https://web.codeboy.com/bmdapi/xxx`发请求访问接口，从而完成相关功能。

上线一个完整项目所需要在服务端安装的软件：

```
mysql   pm2   nodejs
nginx
```

大致步骤

1. **将后端项目上线并测试通过。**
   1. 安装mysql。
   2. 初始化sql脚本。
   3. 将后端程序上传到服务器，配置好数据库访问参数，通过pm2启动服务即可。
2. **将前端项目编译打包上线并完成联调。**
   1. 将VueCLI项目进行打包编译。输出为dist静态资源包。
   2. 下载安装nginx软件，启动nginx服务，代理dist目录作为网站根目录。
   3. 域名管理。

将前端项目编译打包上线并完成联调

1. 将VueCLI项目进行打包编译。输出为dist静态资源包。

   进入vueCLI项目根目录，执行命令：

   ```
   npm  run  build
   ```

   命令执行完毕后，将会在当前目录下生成dist文件夹。

Nginx

nginx是一款优秀的web反向代理服务器，一旦启动nginx软件，将会默认占用80端口，提供web服务（供客户端发送http请求）。

实现步骤：

双击nginx.exe启动nginx服务。

修改conf/nginx.conf配置文件：

```
http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;
    server {
        listen       80;
        server_name  www.bmd.com;
        location / {
            root   dist;
            index  index.html index.htm;
        }
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
    }
}
```

重启nginx：

`nginx.exe -s reload`

DNS域名解析

现在已经上线到`127.0.0.1`。 可以通过以下地址访问该项目：

```
http://127.0.0.1
```

通常情况下由于IP地址不容易记忆，所以建议使用域名代替，例如：

```
http://www.bmd.com
http://www.hantang.com
```

一个域名对应了一个IP地址，这个域名IP的映射关系就由DNS（域名解析）系统来管理。

一台计算机操作系统中也包含了本机的域名解析系统，由一个hosts文件来管理域名与ip的映射关系，在windows系统中，hosts文件的位置：

```
C:\Windows\System32\drivers\etc\hosts
```

该文件有保护权限，修改该hosts文件的内容，即可定义本机的域名解析。

```
# 在hosts文件的末尾追加
127.0.0.1    www.bmd.com
127.0.0.1    www.hantang.com
```

Typescript

Typescript是Javascript的一个超集。Typescript在原有js的基础之上又添加了编译期的类型检查的功能。意味着如果在ts的环境下开发时，会对变量的数据类型进行较为严格的验证，防止程序员写出可能出问题的代码，规范编程习惯。比较适合大项目开发时使用。

Typescript代码的编写及运行方式

typescript代码写在后缀名为`.ts`的文件中，这种文件可以被typescript编译器编译解析，最终将生成一套功能相同的`js`代码，输出到`.js`文件中，typescript语言的类型验证都是在编译期间来处理的。

如果在编译过程中，ts语法有类型验证的错误，则中断编译，报错。

全局安装typescript编译器

```
npm  install  -g  typescript
npm  install  -g  ts-node
npm  install  @types/node
```

安装成功后，就可以使用tsc命令，对ts文件进行编译。

```
tsc  demo.ts
```

安装vscode的插件，方便的运行ts代码。

```
code runner
```

Typescript的数据类型

https://www.w3cschool.cn/typescript/typescript-basic-types.html

```
// basictype.ts  测试ts的基本数据类型

// boolean类型
let married: boolean = true
let loading: boolean = false

// number类型 描述数字
let n1: number = 8          // 十进制
let n2: number = 0b1010     // 二进制
let n3: number = 0o12       // 8进制
let n4: number = 0x0f       // 16进制
console.log({n1, n2, n3, n4})
```

```
// string类型  描述字符串
let name1: string = '亮亮'
console.log(`名字是：${name1}`)

// Array类型  描述数组
let names: string[] = ['亮亮', '小新', '泡泡']
console.log(names)
let ages: number[] = [18, 19, 15]
console.log(ages)
let hobby: Array<string> = ['摊煎饼', '玩单杠', '玩钢管']
console.log(hobby)

// 元组类型    （元素类型可以不一致的数组）
let person: [number, number, string] = [180, 130, '男']
console.log(person)
person = [166, 100, '女']
console.log(person)
// 访问元素 可以得到明确数据类型的结果
console.log(person[2])

// 枚举类型
// 当希望为某一个变量赋值，而赋的值的选择是某一个固定的范围
// 那么就可以声明一个枚举类型的数据来表示可以选择的值的范围
// 还可以给这些值设置一些友好的名字

// 声明一个枚举类型变量，保存新款小米手机可选择的所有的颜色
enum PhoneColor { Red='#f00', Blue='#00f', Green='#0f0' }
// 从枚举中选择一个颜色，赋值给变量 p1Color
let p1Color = PhoneColor.Red
console.log(p1Color)

// 声明一个枚举类型，保存可能会在sessionStorage中使用的KEY值
enum KEYS {User='user', Token='token', Cityname='city'}
// 当希望存储token时：
// sessionStorage.setItem(KEYS.Token, 'token字符串')
// sessionStorage.setItem(KEYS.User, user对象)

// 声明一个枚举类型，保存新增电影时的电影类型 热映、待映、经典
enum Category {Hot=1, Wait=2, Classic=3}

// 声明一个枚举类型，保存可以选择的性别
enum Gender {Male, Female}
console.log('Gender.Male:' + Gender.Male)
console.log('Gender.Female:' + Gender.Female)
// if(person.gender == Gender.Female) {
//   xxxxxx
// }

enum paths {
  ADD_ACTOR='/actor/add',
  LIST_ACTOR='/actors',
  DELETE_ACTOR='/actor/del',
```

```
  LIST_ACTORS_NAME='/actors/name',
}

// 对于枚举类型，TS还贴心的给出通过值反查枚举名称的语法：
// 查询 电影类别是2的电影，属于哪一个类别的电影？
console.log(Category[2])
console.log(Category[1])
console.log(Category[3])
// // 查询 sessionStorage中以token作为key保存的数据，到底是什么数据？
// let key = 'token'
// console.log(KEYS.key)

// any类型    有时候有些变量的类型说不清楚，不一定是什么类型
// 给一个变量设置any类型就意味着告诉ts编译器，不要对这个any变量
// 进行类型语法的检查，当做普通的js变量即可
let str:string = 'abc,cde,def'
console.log(str.split(','))
// console.log(str.toFixed(2))  编译错误  str没有toFixed方法

let str2:any = 'abc,cde,def'
// console.log(str2.toFixed(2))  不会提示编译错误  但是无法运行

// 类型断言（可以理解为 在某些语言中的类型转换）
// 通过类型断言可以清楚的告诉ts编译器 指定某个数据的数据类型
let resp = '{"code":200, "msg":"ok"}'
let respObj = JSON.parse(resp)
// let msg = respObj.msg
// 如何才能让respObj.可以有提示呢？
// 就需要明确的指明 respObj 的数据类型，这里就需要使用类型断言
let r = respObj as {code:number, msg:string}
console.log(r.code)  // 有提示
console.log(r.msg)   // 有提示
```

Typescript中的函数

和JavaScript一样，TypeScript函数可以创建有名字的函数和匿名函数。 你可以随意选择适合应用程序的方式，不论是定义一系列API函数还是只使用一次的函数。