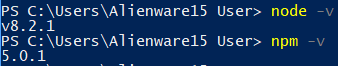
**安装篇**

**Nodejs**

<https://nodejs.org/en/>

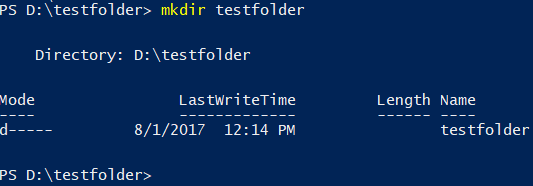
进入官网，点击下载最新msi文件（for windows）。用来安装node.js和npm。

安装完成后在任意目录输入node -v和npm -v



**Express**

1. 在成功安装node以后，在某个路径新建自己的project folde，或直接在terminal中cd到该路径然后输入mkdir folder名，按回车以后电脑自动创建文件夹并直接cd到新的文件夹。



1. 在project folder路径下输入npm init，按回车依次输入合适的信息，或者直接输入npm init -y，直接跳过设置步骤全部使用默认选项。
2. 此时project folder中已经成功新建了package.json文件
3. 在terminal中输入

npm install express --save

1. 成功在当前project中安装express
2. 新建entry point文件（默认名为index.js），你可以在npm init步骤中设置，也可以在package.json文件中设置。这个文件就是我们写服务器代码的地方。
3. 打开该entry point文件（index.js），在开头输入

const express = require(‘express’)

1. 成功将express module引入当前project

**想要持续监听server文件修改，可以使用npm全局下载nodemon**

**MongoDB**

**配置MongoDB**

在windows电脑中，打开mongodb官网，下载合适的文件，解压缩，完成。

运行MangoDB则需要打开该文件夹中的mongod文件（Mongo Daemon后台程序）。

\*\*在打开mongod文件以后才可以打开mongo，也就是mongo shell，在这里才可以输入query。

**连接Express**

下载mongoose

npm install mongoose –save

在project文件中引入mongoose

const mongoose = require('mongoose');

连接database实例

// model/db.js



接下来任何需要连接myApp数据库的文件只需要在开头引入db.js文件，如下图

const db = require('./model/db.js');

引入该文件后数据库会自动连接。

**Angular 4**

**下载安装Angular 4**

1. 使用npm全局下载angular cli

npm install -g @angular/cli

1. 下载完成后，cd到node project路径，在terminal输入

ng new angular-src

1. angular-src就是新建的文件夹名，其中包含了所有Angular需要的文件。

**连接Angular 4和Express**

1. 目前这个文件夹在node project的根目录，我们知道Angular集成了webpack，默认在build 该project以后它会将前端所有需要的HTML，js和css文件打包，然后输出到一个dist文件夹
2. 我们还知道在Express中我们需要设置一个folder用来放所有static files（一般命名为public或者client），这个folder中的文件会直接发给前端。
3. 那么为了能将打包好的Angular文件放进Express中设置的public（或client）folder，给我们需要改动Angular project（我们用的名字是angular-src）根目录的.angular-cli.json文件。
4. 将其中的apps.outDir属性，从”dist” 改为 "../public"



1. 那么从现在开始，在terminal中输入ng serve，打包的文件会保存至public文件夹（ng serve不会显示文件，而是持续监听angular文件的修改）。在开发完成以后输入ng build，打包的文件才存入public，在deploy时使用。

**开发时的监听问题**

输入ng serve以后，Angular会默认在4200端口运行一个dev server，监听任何project路径中文件的改变。显然假如我们既想监听Angular project file的改变，又想监听node project file的改变，无法在一个端口上进行。

但是在前端代码中一定会向server请求data，这种情况下也就是<http://localhost:4200>跨域向<http://localhost:3000>请求data。

默认情况下浏览器会因为安全原因阻止向不同domain发出的XMLHttpRequest，浏览器控制台报错：



解决方法

在node project中下载cors middleware

npm install cors –save

在node 的entry point文件（index.js或app.js或server.js）中引入cors middleware

const cors = require('cors');

在node 的entry point文件中使cors middleware 运行，输入

app.use(cors())；

截至目前

* 我们可以在node project路径下输入nodemon，运行node server并持续监听server文件修改
* 我们可以在Angular project路径下输入ng serve，运行dev server打包前端文件并持续监听前端文件修改