# 1.Express是一个nodejs框架

## express的使用比例

|  |
| --- |
|  |

### 遥遥领先其他nodejs框架

## 安装Express

### npm i express

## 一个简单的express项目

|  |
| --- |
|  |

### package.json

|  |
| --- |
| {    "name": "3.1-express-server",    "version": "1.0.0",    "description": "",    "main": "index.js",    "type": "module",    "scripts": {      "test": "echo \"Error: no test specified\" && exit 1"    },    "keywords": [],    "author": "",    "license": "ISC",    "dependencies": {      "express": "^4.21.2"    }  } |

### index.js

|  |
| --- |
| import express from "express";  const app = express();  const port = 3000;  app.get("/",(req,res)=>{    res.end("<h1>Welcome</h1>")  })  app.listen(port, () => {    console.log(`Server ready: http://localhost:${port}/`);  }); |

### 注意:如果你没有添加”/”的路由,运行访问这个网站就会得到一个错误

|  |
| --- |
|  |

## 小技巧:使用node 来运行js文件在开发过程中并不好用,每一次都需要关闭服务然后重新启动服务,我们可以安装nodemon用它来启动,因为他又热更新功能

### 可以全局安装nodemon方便我们调试: npm install -g nodemon

### 安装后,就可以使用nodemon index.js来运行我们的网站项目了

# 2.使用纯nodejs和使用express来创建项目的区别

|  |  |
| --- | --- |
| 纯nodejs | express |
|  |  |

## 使用express的好处:

### 1.代码容易阅读,

### 2.写的代码量比较少,

### 3.方便使用中间件

## 小技巧:检查电脑在监听哪些端口

|  |
| --- |
|  |
|  |

# 3.使用Postman来测试我们的网站

## 下载: <https://dl.pstmn.io/download/latest/win64>

## 下载后点击安装,会要求登录,我们使用Google账号登陆,登录后的界面如下

|  |
| --- |
|  |

## 小知识,HTTP状态码

|  |
| --- |
|  |

## 小知识2: http请求

|  |
| --- |
|  |

## 小知识3:纯后端渲染项目

|  |
| --- |
|  |

## 练习.

### 进入: G:\C以及其他教程\web开发2024-2025\完整 Web 开发训练营-上（中文字幕）\The Complete 2024 Web Development Bootcamp 2024-1\24 - Express.js with Node.js\3.3 Postman,先运行npm install 安装依赖,然后用nodemon index.js来运行项目,服务器启动成功后,我们可以使用postman来测试,因为这个服务器的路由大多数都是返回状态码,用postman比较方便

|  |
| --- |
|  |

### 测试效果如下

|  |
| --- |
|  |
|  |

### 注意:为例防止混淆,每一个请求最好使用一个独立的选项卡.然后可以保存下来

# 4.express中间件的使用

## 在中间可以插入指定的路由,这样子当这个路由匹配上了,会先执行我们的中间件.有内置的中间件和我们的自定义中间件,也就是自己来写中间件.中间件可以做一些预处理,用户数据验证,日志和错误处理等等许多功能.注意,中间件处理完成后需要放行,否则后面的程序不能得到执行的机会.例如我们需要用到一个非常有用的中间件body-parser

|  |
| --- |
|  |

## 3.4中间件练习,这里使用的express提供的response.sendFile方法,可以直接返回一个html文件

|  |
| --- |
|  |

### 任务1,用index1.js添加/submit路由,然后获取表单数据并且输出到控制台或者返回给浏览器,需要使用body-parser

|  |
| --- |
|  |

#### 效果

|  |
| --- |
|  |

#### 点击Submit按钮后,跳转到/submit路由,返回用户提交的数据给浏览器

|  |
| --- |
|  |

### 任务2.使用日志中间件Morgen,使用index2.js作为服务器应用

#### 1.首先需要安装morgen : npm i morgen

#### 2.然后使用模块语法导入到项目

|  |
| --- |
|  |

#### 3.使用morgen中间件

|  |
| --- |
|  |

#### 4.运行程序,访问<http://localhost:3000/> 发现它在控制台里面输出日志

|  |
| --- |
|  |

#### 可以格式化日志输出

|  |
| --- |
|  |

#### 然后日志输出就会变为

|  |
| --- |
|  |

#### 还可以这么写

|  |
| --- |
|  |

#### 效果是一样的

|  |
| --- |
|  |

#### 其实只需要使用morgan(“tiny”)就可以实现上面的效果

|  |
| --- |
|  |
|  |
|  |

#### 还可以只需要输出错误信息,其他信息不输出

|  |
| --- |
|  |

#### 运行程序,正常访问,没有日志输出

|  |
| --- |
|  |

#### 然后我们来输入一个错误的url

|  |
| --- |
|  |

#### 此时就会有日志输出

|  |
| --- |
|  |

### 任务3.使用自定义中间件,

#### 自定义中间件是一个函数,不过它比普通的回调函数多了一个next函数作为参数,这个函数就是用来放行的

|  |
| --- |
|  |

#### 一个项目可能会用到很多中间件,所以了必须注意他们的调用顺序.因为顺序不同,结果可能就不一样

#### 我们自己写一个中间件,然后使用这个中间件,index3.js的代码如下

|  |
| --- |
|  |

#### 效果

|  |
| --- |
|  |
|  |

### 任务4.用index4.js把之前学习的都复习一遍,用中间件把表单的数据进行拼接然后返回给浏览器

|  |
| --- |
| import express from "express";  import { dirname } from "path";  import { fileURLToPath } from "url";  import bodyParser from "body-parser";  let \_\_dirname = dirname(fileURLToPath(import.meta.url))  const app = express();  const port = 3000;  let bandName = ""  app.use(bodyParser.urlencoded({extended:true}))  app.get("/",(req,res)=>{     res.sendFile(\_\_dirname+"/public/index.html")  })  //自定义中间件  function bandNameMaker(req,res,next) {    // bandName = "✌"+ req.body.street + req.body.pet+"✌"    bandName = "✌"+ req.body["street"] + req.body["pet"]+"✌"    next()  }  app.use(bandNameMaker)  app.post("/submit",(req,res)=>{     res.send(`<h1>New Band Name Generated</h1><br><h2>${bandName}</h2>`)  })  app.listen(port, () => {    console.log(`Server ready:http://localhost:${port}/`);  }); |

#### public/index.html的内容如下

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Band Name Generator</title>  </head>  <body>    <h1>Band Name Generator</h1>    <form action="/submit" method="POST">      <label for="street">Street Name:</label>      <input type="text" name="street" required>      <label for="pet">Pet Name:</label>      <input type="text" name="pet" required>      <input type="submit" value="Submit">    </form>  </body>  </html> |

#### 效果,先访问<http://localhost:3000>,然后填写数据

|  |
| --- |
|  |

#### 点击Submit按钮,效果如下

|  |
| --- |
|  |

##### 注意这个案例使用了2给中间件一个是bodyParser,一个是我们的bandNameMaker,一定要先定义bodyParser,再调用bandNameMaker,否则会报错,因为如果调用bandNameMaker在bodyParser的前面,req.body是undefined,就抛异常因为不能获取undefined的属性

|  |
| --- |
|  |

# 5.Secrets Access案例

## public/index.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>    <title>Secrets</title>  </head>  <body>    <h1>Secrets</h1>    <form action="/check" method="POST">      <label for="password">Password:</label>      <input type="text" id="password" name="password" required>      <input type="submit" value="Submit">    </form>  </body>  </html> |

## public/secret.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Secrets</title>  </head>  <body>    <h1>Secrets</h1>    <ul>      <li>When making chocolate desserts, always add a little salt. It makes the chocolate flavour more intense.</li>      <li>Whenever you travel abroad bring a new soundtrack for each place you visit, preferably one you have never heard        before.        In the future, every time you listen to each soundtrack again they will bring you vivid memories of the places you        have        visited.</li>      <li>Do not try to be the person your parents would want you to be. Be the person you would like your child to be be.        It more        clearly        defines your own convictions, desires, goals, and motivates you to be your best.</li>      <li>If a friend or a family member gets diagnosed with dementia or alzheimer, in the early stages try to find out        what their        favorite songs of all time are. In this way you would be able to create a playlist for them that could be of great        benefit in the later stages of the disease.</li>      <li>When a friend is upset, ask them one simple question before saying anything else: 'Do you want to talk about it        or do        you want to be distracted from it?'</li>      <li>If you ever forget your WiFi password or you want to get your school WiFi password etc. Just type this command        into the        command line of a computer already connected to that WiFi: netsh wlan show profile WiFi-name key=clear</li>    </ul>  </body>  </html> |

## index.js

|  |
| --- |
| //To see how the final website should work, run "node solution.js".  //Make sure you have installed all the dependencies with "npm i".  //The password is ILoveProgramming  import express from "express"  import bodyParser from "body-parser"  import {dirname} from 'path'  import { fileURLToPath } from "url"  let \_\_dirname = dirname(fileURLToPath(import.meta.url))  let app = express()  let port =3000  app.use(bodyParser.urlencoded({extended:true}))  app.get("/",(req,res)=>{      res.sendFile(\_\_dirname+"/public/index.html")  })  function Authentication(req,res,next) {      let pwd = req.body["password"]      if(pwd!=="ILoveProgramming")      {          res.redirect("/");      } else{          next();      }  }  app.use(Authentication)  app.post("/check",(req,res)=>{      res.sendFile(\_\_dirname+"/public/secret.html")  })  app.listen(port,()=>{      console.log(`Server ready: http://localhost:${port}/`);  }) |

### 老师的代码是这样子的

|  |
| --- |
| import express from "express";  import bodyParser from "body-parser";  import { dirname } from "path";  import { fileURLToPath } from "url";  const \_\_dirname = dirname(fileURLToPath(import.meta.url));  const app = express();  const port = 3000;  var userIsAuthorised = false;  app.use(bodyParser.urlencoded({ extended: true }));  function passwordCheck(req, res, next) {    const password = req.body["password"];    if (password === "ILoveProgramming") {      userIsAuthorised = true;    }    next();  }  app.use(passwordCheck);  app.get("/", (req, res) => {    res.sendFile(\_\_dirname + "/public/index.html");  });  app.post("/check", (req, res) => {    if (userIsAuthorised) {      res.sendFile(\_\_dirname + "/public/secret.html");    } else {      res.sendFile(\_\_dirname + "/public/index.html");      //Alternatively res.redirect("/");    }  });  app.listen(port, () => {    console.log(`Listening on port ${port}`);  }); |

### 效果:运行程序,在浏览器在输入: <http://localhost:3000/> ,效果如下

|  |
| --- |
|  |

### 此时如果密码不正确,点击Submit,不会跳转到Secret,还是会留在index.html中

|  |
| --- |
|  |
|  |

### 然后我们输入正确的密码

|  |
| --- |
|  |
|  |

### 注意:bodyParse现在已经整合到express里面了,你可以不用安装

|  |
| --- |
| import express from "express";  // import bodyParser from "body-parser";  import { dirname } from "path";  import { fileURLToPath } from "url";  const \_\_dirname = dirname(fileURLToPath(import.meta.url));  const app = express();  const port = 3000;  var userIsAuthorised = false;  app.use(express.urlencoded({ extended: true })); //可以用express来代替bodyParser  function passwordCheck(req, res, next) {    const password = req.body["password"];    if (password === "ILoveProgramming") {      userIsAuthorised = true;    }    next();  }  app.use(passwordCheck);  app.get("/", (req, res) => {    res.sendFile(\_\_dirname + "/public/index.html");  });  app.post("/check", (req, res) => {    if (userIsAuthorised) {      res.sendFile(\_\_dirname + "/public/secret.html");    } else {      res.sendFile(\_\_dirname + "/public/index.html");      //Alternatively res.redirect("/");    }  });  app.listen(port, () => {    console.log(`Listening on port ${port}`);  }); |

### 效果是一样的.当然建议还是使用bodyParser

# 选项.安装Express Generator用来快速生成express项目

## 需要全局安装express-generator: npm install -g express-generator

## 安装完成后,查看版本: express --version

# Express Generator参考

# Express application generator

Use the application generator tool, express-generator, to quickly create an application skeleton.

You can run the application generator with the npx command (available in Node.js 8.2.0).

$ npx express-generator

For earlier Node versions, install the application generator as a global npm package and then launch it:

$ npm install -g express-generator

$ express

Display the command options with the -h option:

$ express -h

Usage: express [options] [dir]

Options:

-h, --help output usage information

--version output the version number

-e, --ejs add ejs engine support

--hbs add handlebars engine support

--pug add pug engine support

-H, --hogan add hogan.js engine support

--no-view generate without view engine

-v, --view <engine> add view <engine> support (ejs|hbs|hjs|jade|pug|twig|vash) (defaults to jade)

-c, --css <engine> add stylesheet <engine> support (less|stylus|compass|sass) (defaults to plain css)

--git add .gitignore

-f, --force force on non-empty directory

For example, the following creates an Express app named **myapp**. The app will be created in a folder named **myapp** in the current working directory and the view engine will be set to [Pug](https://pugjs.org/):

$ express --view=pug myapp

create : myapp

create : myapp/package.json

create : myapp/app.js

create : myapp/public

create : myapp/public/javascripts

create : myapp/public/images

create : myapp/routes

create : myapp/routes/index.js

create : myapp/routes/users.js

create : myapp/public/stylesheets

create : myapp/public/stylesheets/style.css

create : myapp/views

create : myapp/views/index.pug

create : myapp/views/layout.pug

create : myapp/views/error.pug

create : myapp/bin

create : myapp/bin/www

Then install dependencies:

$ cd myapp

$ npm install

On MacOS or Linux, run the app with this command:

$ DEBUG=myapp:\* npm start

On Windows Command Prompt, use this command:

> set DEBUG=myapp:\* & npm start

On Windows PowerShell, use this command:

PS> $env:DEBUG='myapp:\*'; npm start

Then, load http://localhost:3000/ in your browser to access the app.

The generated app has the following directory structure:

.

├── app.js

├── bin

│ └── www

├── package.json

├── public

│ ├── images

│ ├── javascripts

│ └── stylesheets

│ └── style.css

├── routes

│ ├── index.js

│ └── users.js

└── views

├── error.pug

├── index.pug

└── layout.pug

7 directories, 9 files