

T8.Node.js

Step1:

1.下载Node.js（采用fnm包管理器）

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\30645> fnm env --use-on-cd | Out-String | Invoke-Expression
PS C:\Users\30645> fnm use --install-if-missing 20
Installing Node v20.17.0 (x64)
00:00:03 28.20 MiB/28.20 MiB (7.86 MiB/s, 0s)
Using Node v20.17.0
PS C:\Users\30645> node -v
v20.17.0
PS C:\Users\30645> npm -v
10.8.2
PS C:\Users\30645> |
```

2.初始化

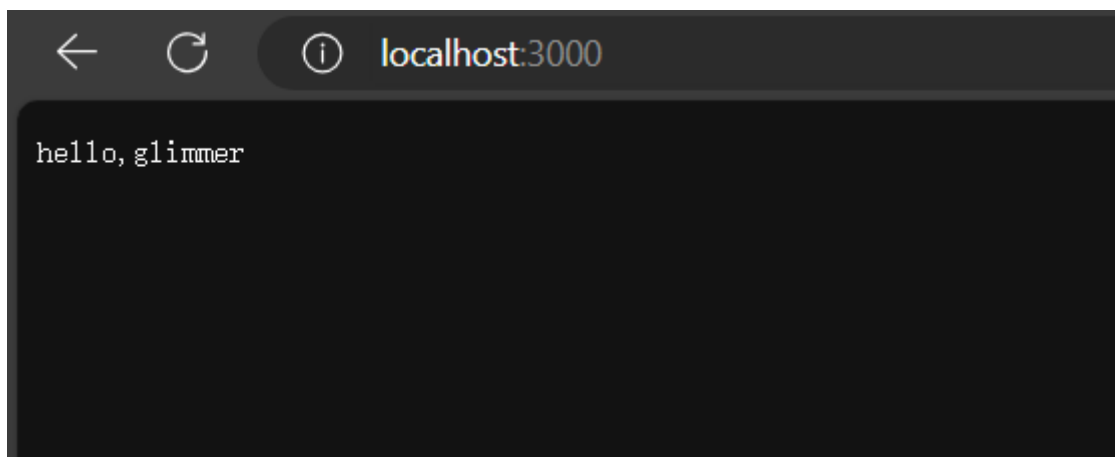
```
PS D:\mynode> node -v
v20.17.0
PS D:\mynode> npm init -y
Wrote to D:\mynode\package.json:

{
  "name": "mynode",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

3.生成一个server.js并且创建服务器，同时将信息输入到命令行的控制台和浏览器。

4.运行

浏览器：

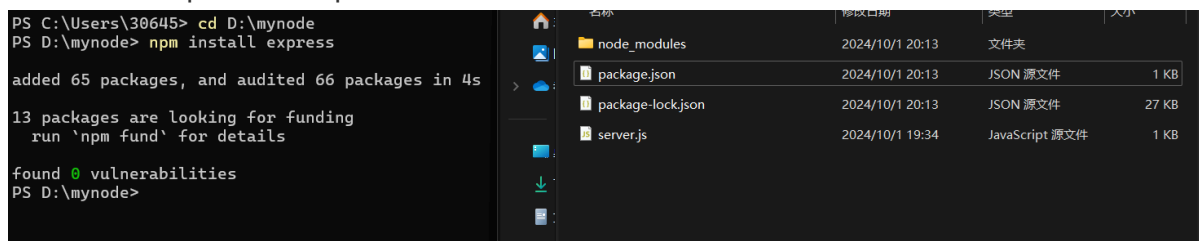


命令框：

```
PS D:\mynode> node server.js
Server is running at http://localhost3000
hello, glimmer
```

Step2:

1.利用npm下载express!



2.express框架：迷你版一言的制作

```

JS miniyiyan.js > ...
1  const express=require('express');
2  const app=express();
3  const PORT=3000
4
5  const mysentences = [
6      "即使活着没有意义，偶尔也还是会有好事发生",
7      "啊，真是短暂的人生。",
8      "Some peop are worth melting for.",
9      "人似乎是昼伏夜伏的生物，但经常需要昼出夜出。",
10     "想了解一个人，不要听他说了什么，要看他最后得到了什么。",
11     "含着泪我一读再读，却不得不承认，青春是一本太仓促的书。",
12     "百年后在荒野上放声大笑的人不必是我。",
13     "枪法也是法，弹道也是道，你道爷我拿起这把AK，这世上就没有荡不平的八十一难。",
14     "站起来，不许跪。",
15     "人生的容错率比你想象的要高得多，生活中百分之99的人你都得罪得起。",
16     "每天做着想死的工作，竟然是为了活着。",
17     "我年轻的时候很穷，努力了几年，终于不再年轻了。",
18     "其实是不是每个人都在说着长达几十年的遗言？",
19     "我闭上眼触碰星空，阅读宇宙给我留下的盲文。",
20     "醉后不知天在水，满船清梦压星河。",
21 ];
22
23 app.get('/Mini-yiyan',(req,res)=>{
24     let randomIndex=Math.floor(Math.random()*mysentences.length)//简单实现随机数
25     let randomSentence=mysentences[randomIndex]
26     res.json({sentence:randomSentence})
27 })
28 app.listen(PORT,()=>{
29     console.log(`Server is running at http://localhost:${PORT}`)
30 })

```

代码部分：

```

const express=require('express');
const app=express();
const PORT=3000

const mysentences = [
    "即使活着没有意义，偶尔也还是会有好事发生",
    "啊，真是短暂的人生。",
    "Some peop are worth melting for.",
    "人似乎是昼伏夜伏的生物，但经常需要昼出夜出。",
    "想了解一个人，不要听他说了什么，要看他最后得到了什么。",
    "含着泪我一读再读，却不得不承认，青春是一本太仓促的书。",
    "百年后在荒野上放声大笑的人不必是我。",
    "枪法也是法，弹道也是道，你道爷我拿起这把AK，这世上就没有荡不平的八十一
难。",
    "站起来，不许跪。",
    "人生的容错率比你想象的要高得多，生活中百分之99的人你都得罪得起。",
    "每天做着想死的工作，竟然是为了活着。",
    "我年轻的时候很穷，努力了几年，终于不再年轻了。",
    "其实是不是每个人都在说着长达几十年的遗言？",
    "我闭上眼触碰星空，阅读宇宙给我留下的盲文。",

```

```

    "醉后不知天在水，满船清梦压星河。",
  ];

  app.get('/Mini-yiyan',(req,res)=>{
    let randomIndex=Math.floor(Math.random()*mysentences.length)//
    简单实现随机数字并向下取整、
    let randomSentence=mysentences[randomIndex]
    res.json({sentence:randomSentence})
  })
  app.listen(PORT,()=>{
    console.log(`Server is running at http://localhost:${PORT}`)
  })

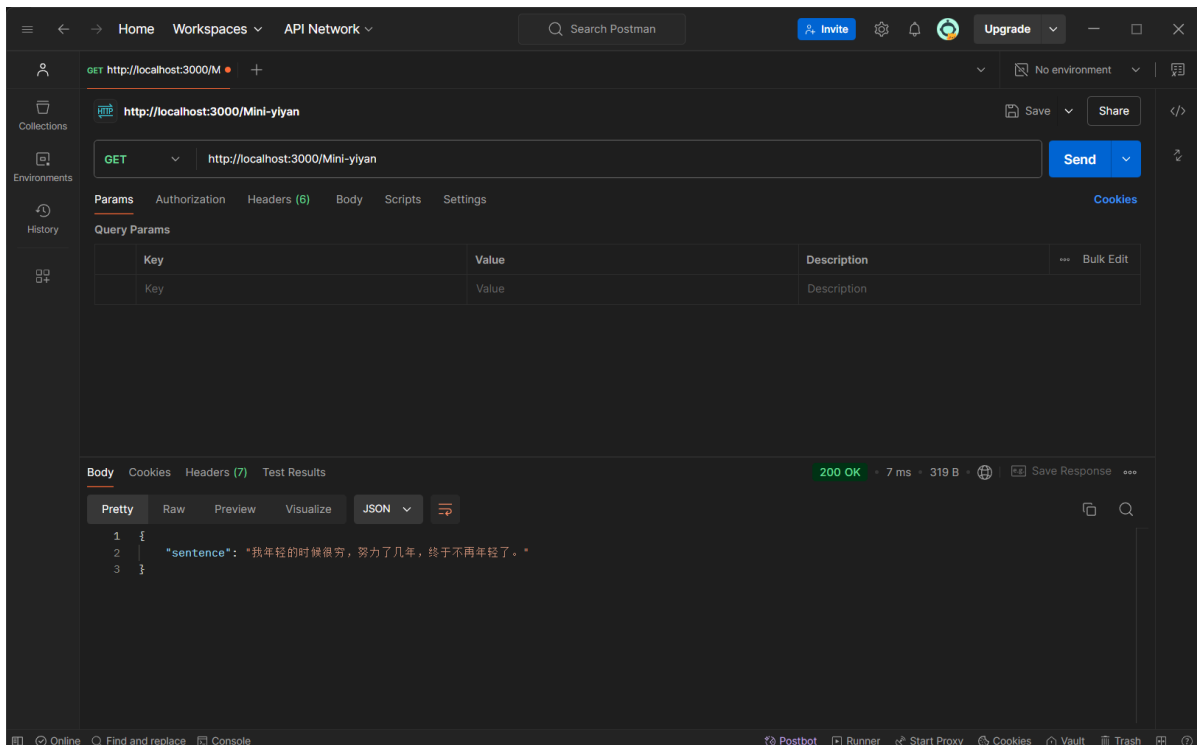
```

本地运行成功：



Step3:

在Postman上面进行测试：



英文看不懂啦（当然后来安装了汉化包qwq）

Step4:

可不可以自己向自己请求？

使用curl可以做到这一点：

```
curl http://localhost:3000/Mini-yiyan
```

输出：

```
PS C:\Users\30645> curl http://localhost:3000/Mini-yiyan
StatusCode      : 200
StatusDescription : OK
Content         : {"sentence":"想了解一个人，不要听他说了什么，要看他最后得到了什么。"}
RawContent      : HTTP/1.1 200 OK
                  Connection: keep-alive
                  Keep-Alive: timeout=5
                  Content-Length: 96
                  Content-Type: application/json; charset=utf-8
                  Date: Tue, 01 Oct 2024 15:35:01 GMT
                  ETag: W/"60-IzJ7E9Nqeej5JCDMcR2w..."
Forms           : {}
Headers         : {[Connection, keep-alive], [Keep-Alive, timeout=5], [Content-Length, 96], [Content-Type, applicatio
                  n/json; charset=utf-8]...}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 96
```

和postman的主要区别是curl命令行直接使用底层的网络协议，通过底层libcurl库发送HTTP请求，而postman通常通过发送请求到本地的nodejs服务器来实现，更加直观、方便测试。

Step5:

了解登陆和注册的原理？

登陆和注册通常使用cookie和session来实现：A向服务器B请求，B生成唯一的sessionID并储存，向A返回以对应session的cookie，在A第二次（带着cookie）向B请求时，B能够辨识出属于A的session。

下面是通过flask框架实现登陆与注册的功能（不大会nodejs，止步于使用express，而且flask感觉也好难 🤖）

创建如下结构

```
flask/
|
├─ app.py
├─ test.py（我在这里测试了一下mysql数据库，干脆直接打包给app了）
└─ templates/
    ├─ login.html
    ├─ register.html
    └─ index.html（以后注册了路由再跳转）
```

app.py:

```
from flask import Flask, request, render_template
from test import con_my_sql
app=Flask(__name__)
#这个是打算跳转的
@app.route('/index')
#def index():
#    return render_template('index.html')

#定义几个路由
@app.route('/')
def index():
    return render_template('login.html')

@app.route('/register')
def index_register():
    return render_template('register.html')

@app.route('/login' ,methods=["post"])
def login():
    name = request.form.get('username')
    pwd=request.form.get('password')

    code = "select * from login_user where username='%s'" % (name)
    cursor_ans = con_my_sql(code)
    cursor_select = cursor_ans.fetchall()
    if len(cursor_select) > 0:
        if pwd == cursor_select[0]['password']:
            return "登陆成功"
        else:
            return '密码错误 <a href="/">返回登录</a>'
    else:
        return '用户不存在 <a href="/">返回登录</a>'

@app.route('/register', methods=["post"])
def register():
    name = request.form.get('username')
    pwd = request.form.get('password')

    code = "select * from login_user where username='%s'" %
(name)
    cursor_ans = con_my_sql(code)
    cursor_select = cursor_ans.fetchall()
```

```

        if len(cursor_select) > 0:
            return '用户已存在 <a href="/">返回登录</a>'
        else:
            code = "INSERT INTO `login_user` (`username`,
`password`) VALUES ('%s', '%s')" % (name, pwd)
            con_my_sql(code)
            return '注册成功 <a href="/">返回登录</a>'

if __name__ == '__main__':
    app.run(debug=True)

```

test.py:

```

import pymysql

conn = pymysql.connect(host='localhost', port=3306,
                        user='root', password='TXYq123456789q',
                        database='demo01', charset='utf8mb4')

def con_my_sql(sql_code):
    try:
        conn.ping(reconnect=True)
        print(sql_code)
        cursor = conn.cursor(pymysql.cursors.DictCursor)
        cursor.execute(sql_code)
        conn.commit()
        cursor.close()
        return cursor
    #回滚
    except pymysql.MySQLError as err_message:
        conn.rollback()
        conn.close()
        return type(err_message), err_message

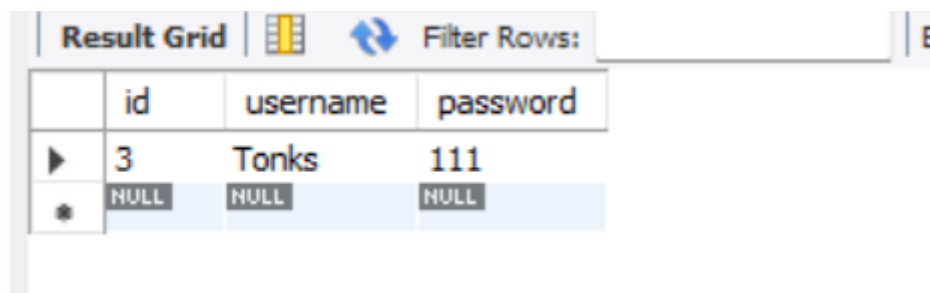
```

数据库配置:



```
Query 1 x login_user
1 • CREATE SCHEMA demo01 default character set utf8mb4 ;
2 • use demo01;
3 • create table `demo01`.`login_user` (
4     `id` int not null auto_increment,
5     `username` varchar(45) not null,
6     `password` varchar(45) not null,
7     primary key (`id`),
8     unique index `idlogin_user_UNIQUE` (`id` ASC),
9     unique index `username_UNIQUE` (`username` ASC));
```

这样的话，注册中输入的字符和数据能被传入数据库，获得专属id，等待登录时检查：



	id	username	password
▶	3	Tonks	111
•	NULL	NULL	NULL

我暂时没有搞懂主页路由的“根据用户的登陆状态显示内容”具体怎么实现，不过猜测应该是通过在相关路由检查用户的登陆状态，比如

```
if 'username' not in session:
    flash('请先登录!', 'danger')
    return redirect(URL_for('login'))
```

但是我没能做到运行它，可能和我的py水平有关系（≈ 0）

我做不到未登录时不跳转、只显示部分内容（对py的理解太浅了，很多功能并不知道怎么实现）。

学习使用flask框架的过程中遇到了许多困难，连最基本的语句都要上网查一下功能用途，感觉好难受。我原以为Python是学机器学习时入门的，现在来看根本没有“入门”，还要抽空学习vue、js和css.....完全安排不过来。

辛苦的日子还在等着我.jpg 🤖