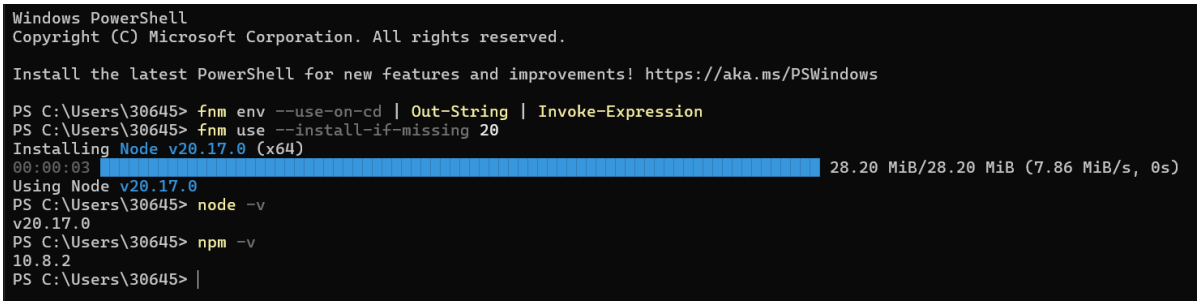
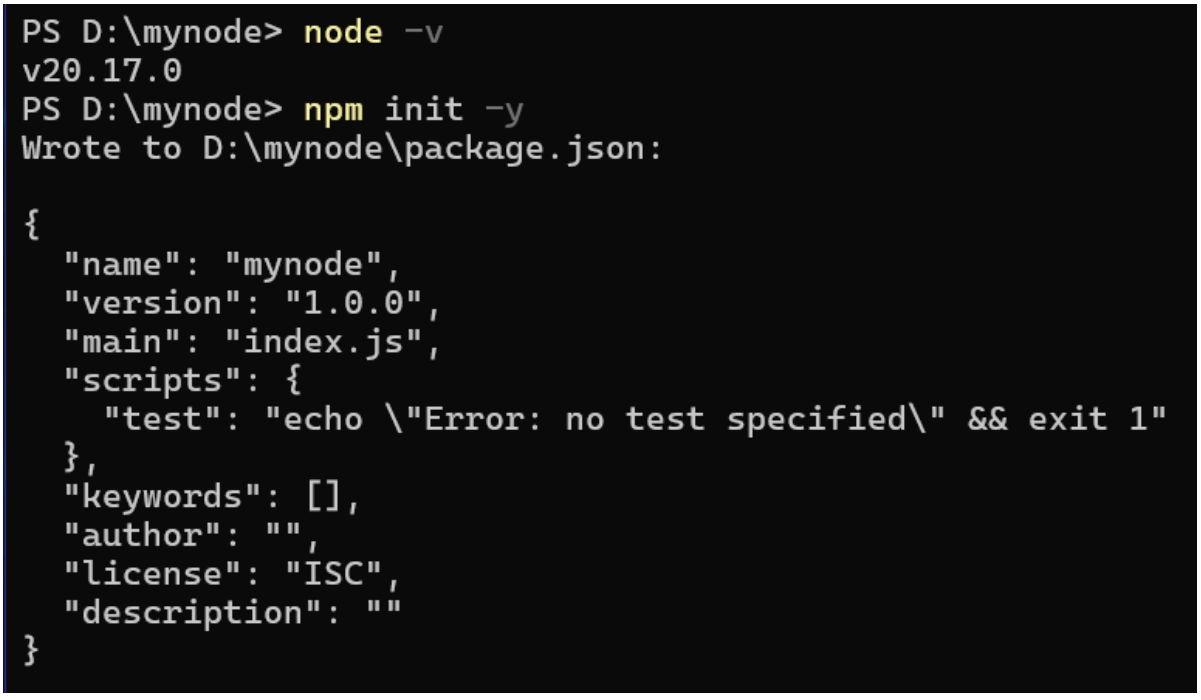
T8.Node.js

Step1：

1.下载Node.js（采用fnm包管理器）



2.初始化



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。

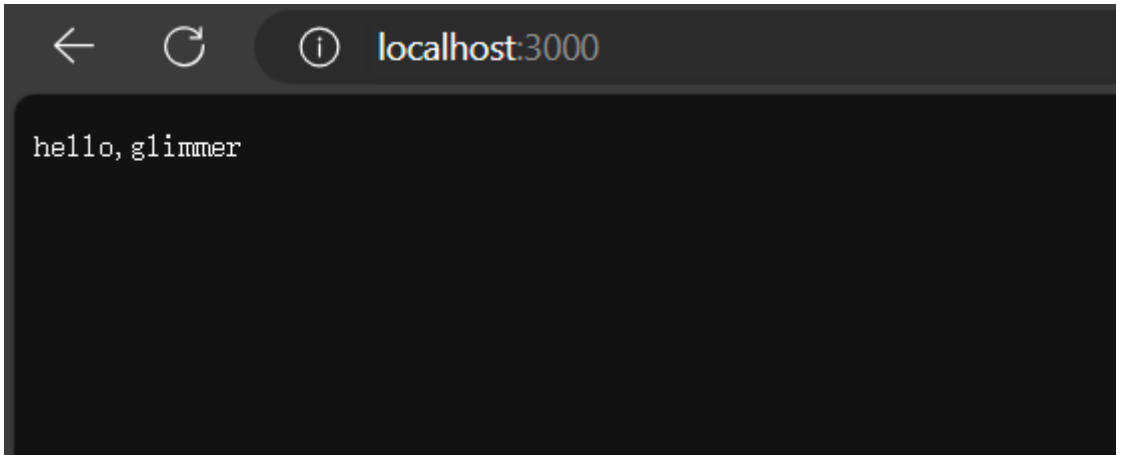
server.js

D: > mynode > server.js > server.listen() callback

```
1
2
3   const http=require('http');
4   const server=http.createServer((req,res)=>{
5       res.writeHead(200,{'Content-type':'text/plain'})
6       res.end("hello,glimmer")
7   })
8
9
10
11  const port=3000;
12  server.listen(port,()=>{
13      console.log('hello,glimmer')
14  })
```

4.运行

浏览器：

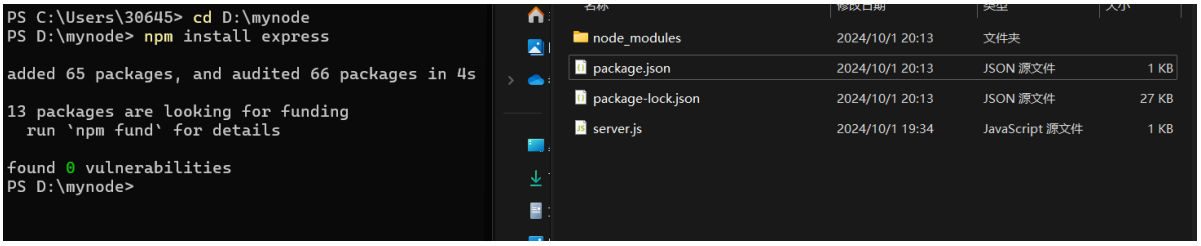


命令框：

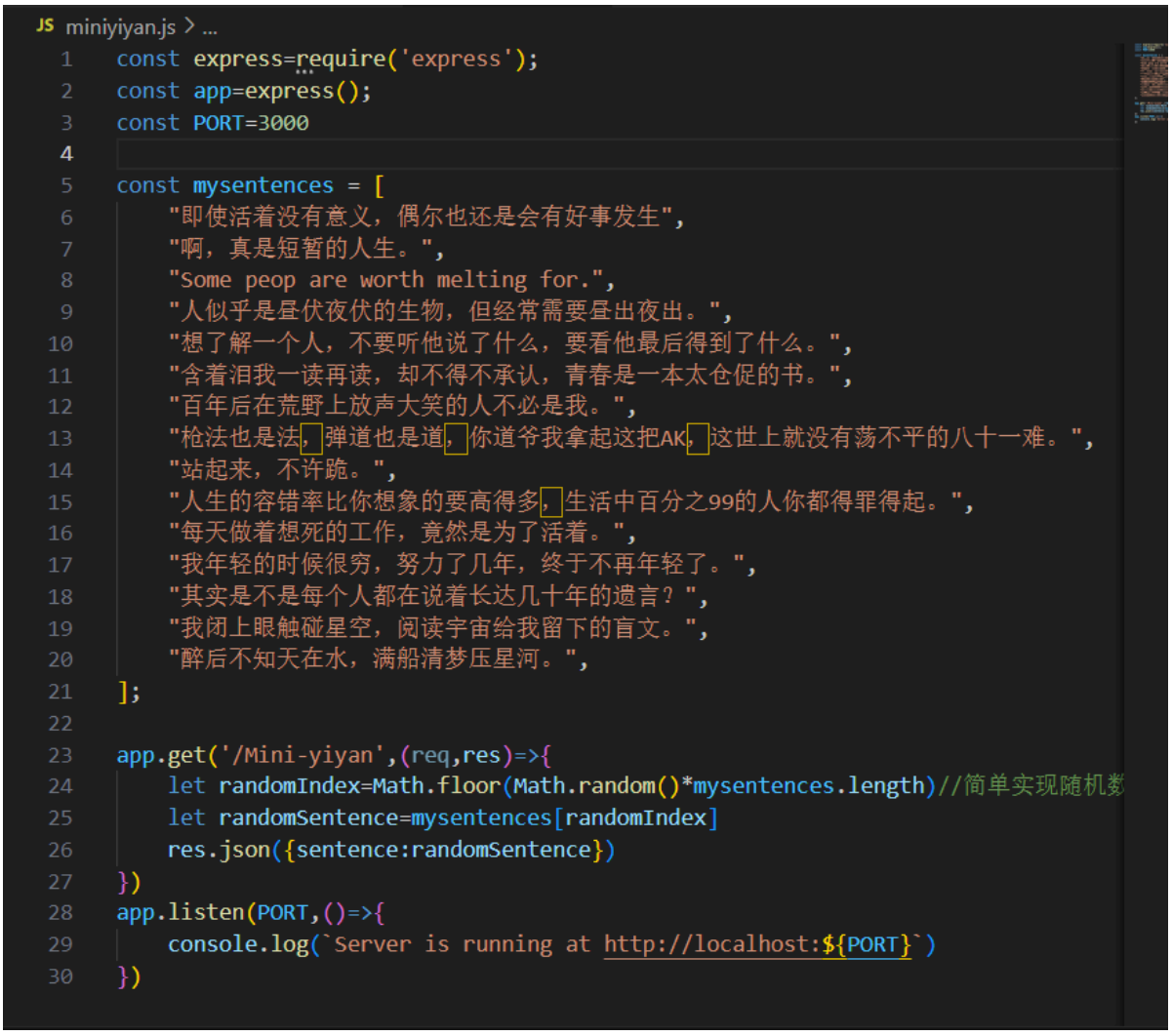
```
PS D:\mynode> node server.js
Server is running at http://localhost3000
hello,glimmer
```

Step2:

1.利用npm下载express



2.express框架：迷你版一言的制作



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。",
```

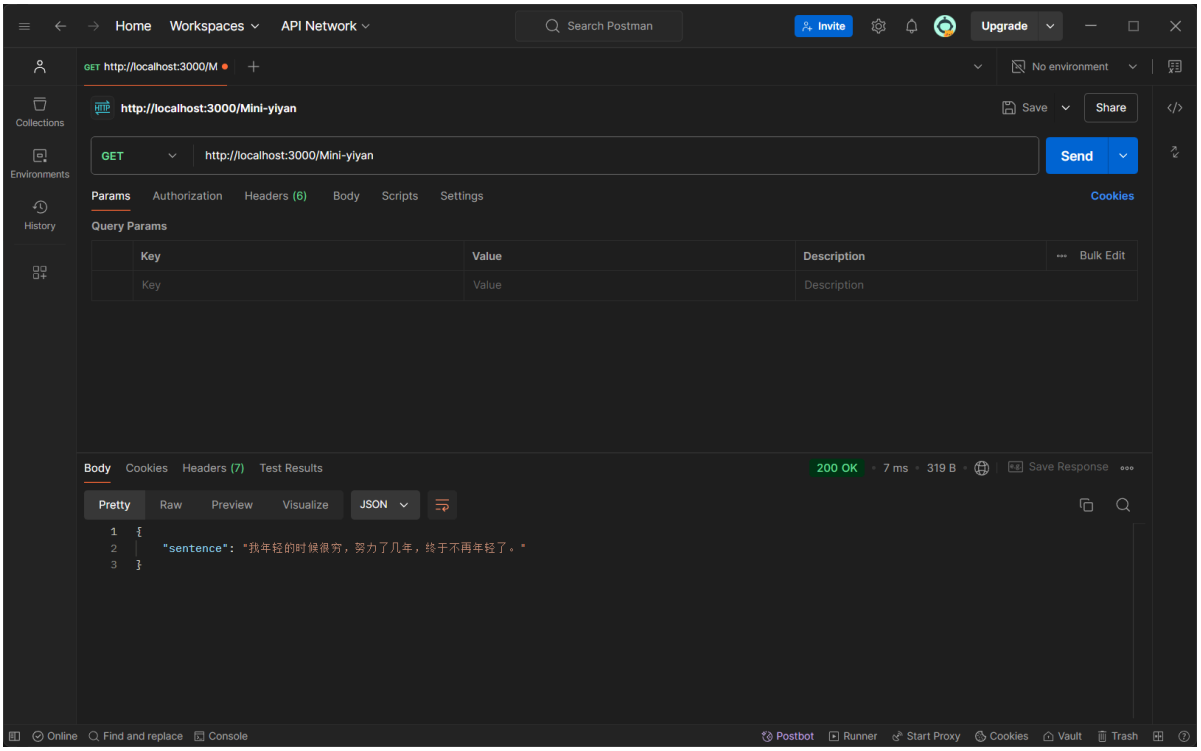
```
    "想了解一个人，不要听他说了什么，要看他最后得到了什么。",
    "含着泪我一读再读，却不得不承认，青春是一本太仓促的书。",
    "百年后在荒野上放声大笑的人不必是我。",
    "枪法也是法，弹道也是道，你道爷我拿起这把AK，这世上就没有荡不平的八十一难。",
    "站起来，不许跪。",
    "人生的容错率比你想象的要高得多，生活中百分之99的人你都得罪得起。",
    "每天做着想死的工作，竟然是为了活着。",
    "我年轻的时候很穷，努力了几年，终于不再年轻了。",
    "其实是不是每个人都在说着长达几十年的遗言？",
    "我闭上眼触碰星空，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
├── models.py
├── forms.py
└── templates/
    ├── login.html
    ├── register.html
    └── index.html
```

app.py:

```
from flask import Flask, render_template, redirect, url_for, flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, UserMixin, login_user,
login_required, logout_user, current_user
from forms import RegistrationForm, LoginForm
from models import db, User #models和forms在下面

app = Flask(__name__)
app.config['SECRET_KEY'] = 'your_secret_key'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
#资料：SQLAlchemy 是一个对象关系映射库，以此用Python对象操作数据库。
#另外可以使用MySQL这样的数据库，或者干脆直接手动创建database.db，在pycharm下载插件来管理（淘宝6块钱买的专业版激活码😍）
db.init_app(app)

login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'

@login_manager.user_loader
```

```
def load_user(user_id):
    return User.query.get(int(user_id))

@app.route('/')
@login_required
def index():
    return render_template('index.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        user = User(username=form.username.data,
password=form.password.data)
        db.session.add(user)
        db.session.commit()
        flash('注册成功', 'success')
        return redirect(url_for('login'))
    return render_template('register.html', form=form)

@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user =
User.query.filter_by(username=form.username.data).first()
        if user and user.password == form.password.data:
            login_user(user)
            return redirect(url_for('index'))
        else:
            flash('登陆失败，用户名或密码不正确', 'danger')
    return render_template('login.html', form=form)

@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))
#这一步让它自动创建数据库，一开始帮它创建了一个db结果报错，不明白为什么，最后上网找了一个生成数据库的样例，扒了这段代码
if __name__ == '__main__':
    with app.app_context():
        db.create_all()
    app.run(debug=True)
```

models.py用于定义用户模型：

```
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin

db = SQLAlchemy()

class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(150), unique=True,
nullable=False)
    password = db.Column(db.String(150), nullable=False)

    def __repr__(self):
        return f'<User {self.username}>'
```

forms.py用于定义注册/登陆表单

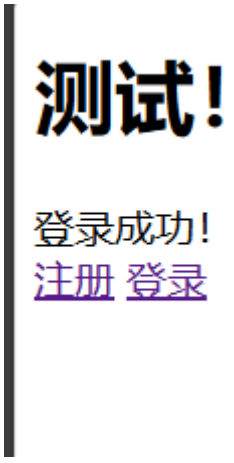
```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import DataRequired, Length, EqualTo

class RegistrationForm(FlaskForm):
    username = StringField('用户名', validators=[DataRequired(),
Length(min=2, max=150)])
    password = PasswordField('密码', validators=[DataRequired()])
    #10.3补充：添加再次确认密码
    confirm_password = PasswordField('再次确认', validators=
[DataRequired(), EqualTo('password')])
    submit = SubmitField('注册')

class LoginForm(FlaskForm):
    username = StringField('用户名', validators=[DataRequired()])
    password = PasswordField('密码', validators=[DataRequired()])
    submit = SubmitField('登陆')
```

再写好登陆和注册网页后，运行：

返回登陆界面，输入信息：



能够保存登陆状态（session）。不过登陆成功后没有跳转，因为没有这个功能qwq。

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）

最终我还是没有成功实现这个功能，也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。Python是学机器学习时入门的，现在来看根本没有“成功入门”，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 😭