云计算与大数据应用开发

实验五：云计算应用开发（一）

丁烨
dingye@dgut.edu.cn
计算机科学与技术学院

2024-05-13

東莞理工學院
DONGGUAN UNIVERSITY OF TECHNOLOGY

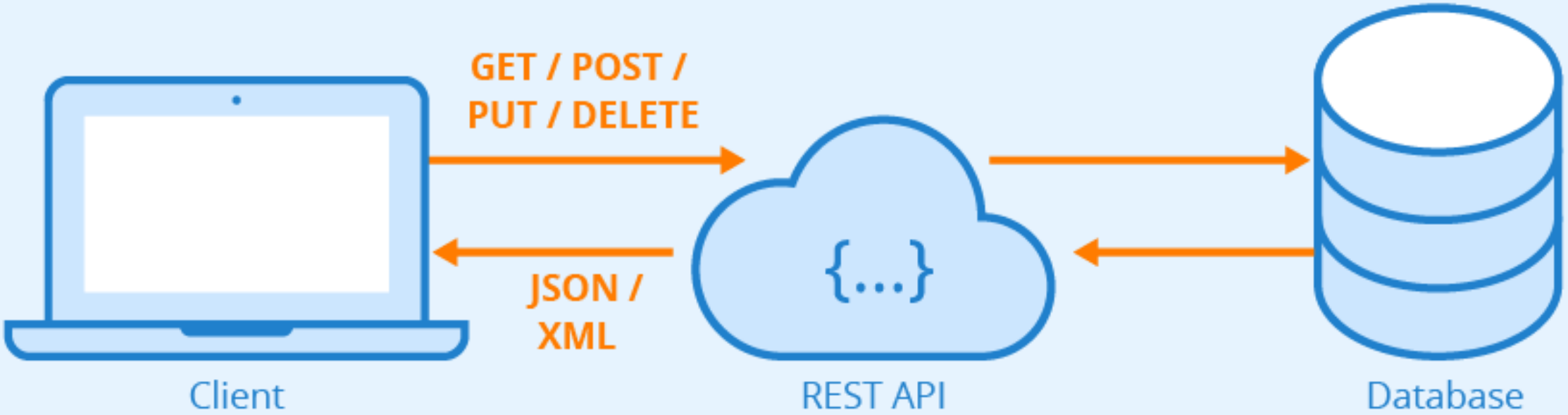
应用程序接口概述

REST API

- 表现层状态转换（Representational State Transfer，REST）
- 由 Roy Thomas Fielding 于 2000 年提出来的一种互联网应用程序架构
- 目的是便于不同应用程序在互联网中互相传递信息
- 允许客户端发出统一的、无状态的资源标识符访问和操作网络资源
- 相对于其它种类的互联网应用程序架构，例如 SOAP（Simple Object Access Protocol），REST 的无状态特性降低了多平台应用程序的开发强度
- REST 是目前 SaaS 的主流技术架构

应用程序接口概述

REST API



应用程序接口概述

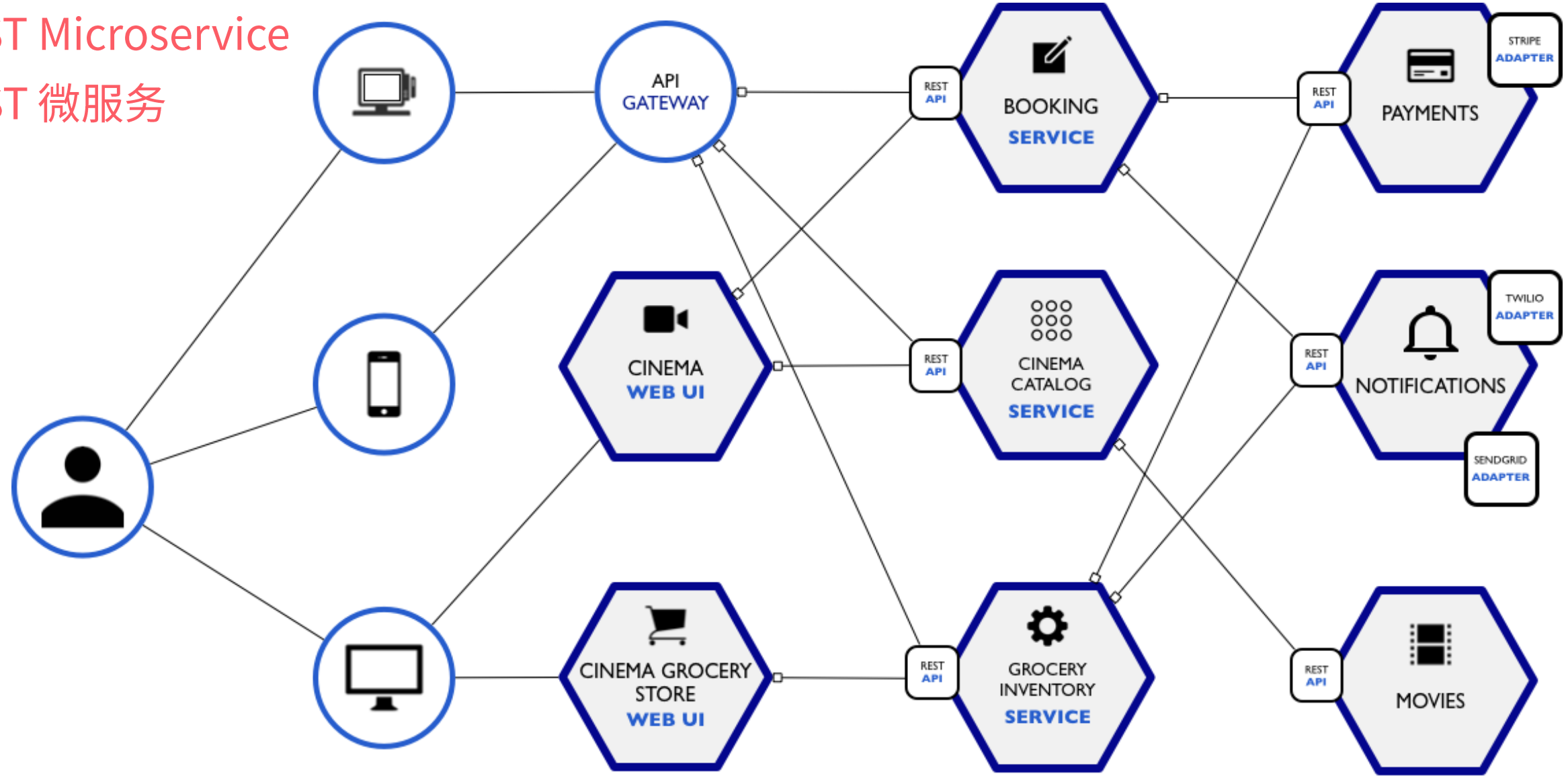
REST API

- 应用程序接口（Application Programming Interface，API）
- 又称“后端”、“服务端”
- 响应客户端请求，并返回数据的 REST 架构组件
- API 可以连接数据库获取数据，也可以连接其他 API 获取数据
- 客户端和 API 的统一接口通常为 JSON 或 XML
- API 和数据库的统一接口通常为 ORM

应用程序接口概述

REST API

- REST Microservice
- REST 微服务



Flask 简介

- Flask
- https://flask.palletsprojects.com/

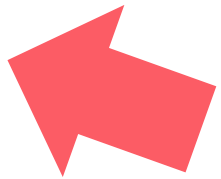
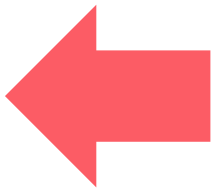
- 一个使用 Python 编写的轻量级互联网应用 API 框架
- 基于 Werkzeug WSGI 工具箱和 Jinja2 模板引擎，使用 BSD 授权
- Flask 被称为“微框架（Micro Framework）”，因为它的核心非常简单
- Flask 可以通过插件的形式增加复杂的功能
- 例如数据库（ORM）、文件上传、各种开放式身份验证技术（OAuth）等

Flask 使用指南

- 安装 Flask
- `pip3 install flask`

❖ 测试 Flask

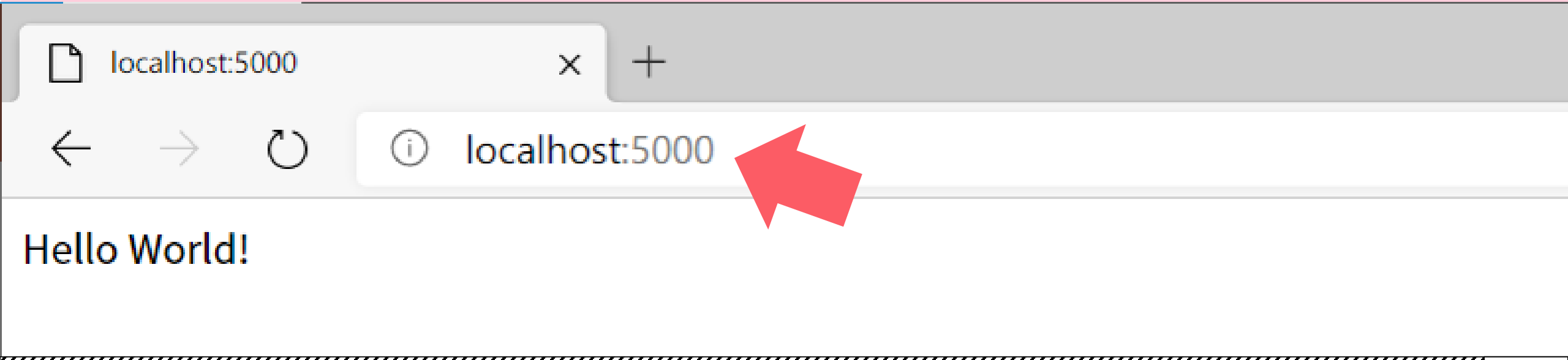
```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```


Flask 使用指南

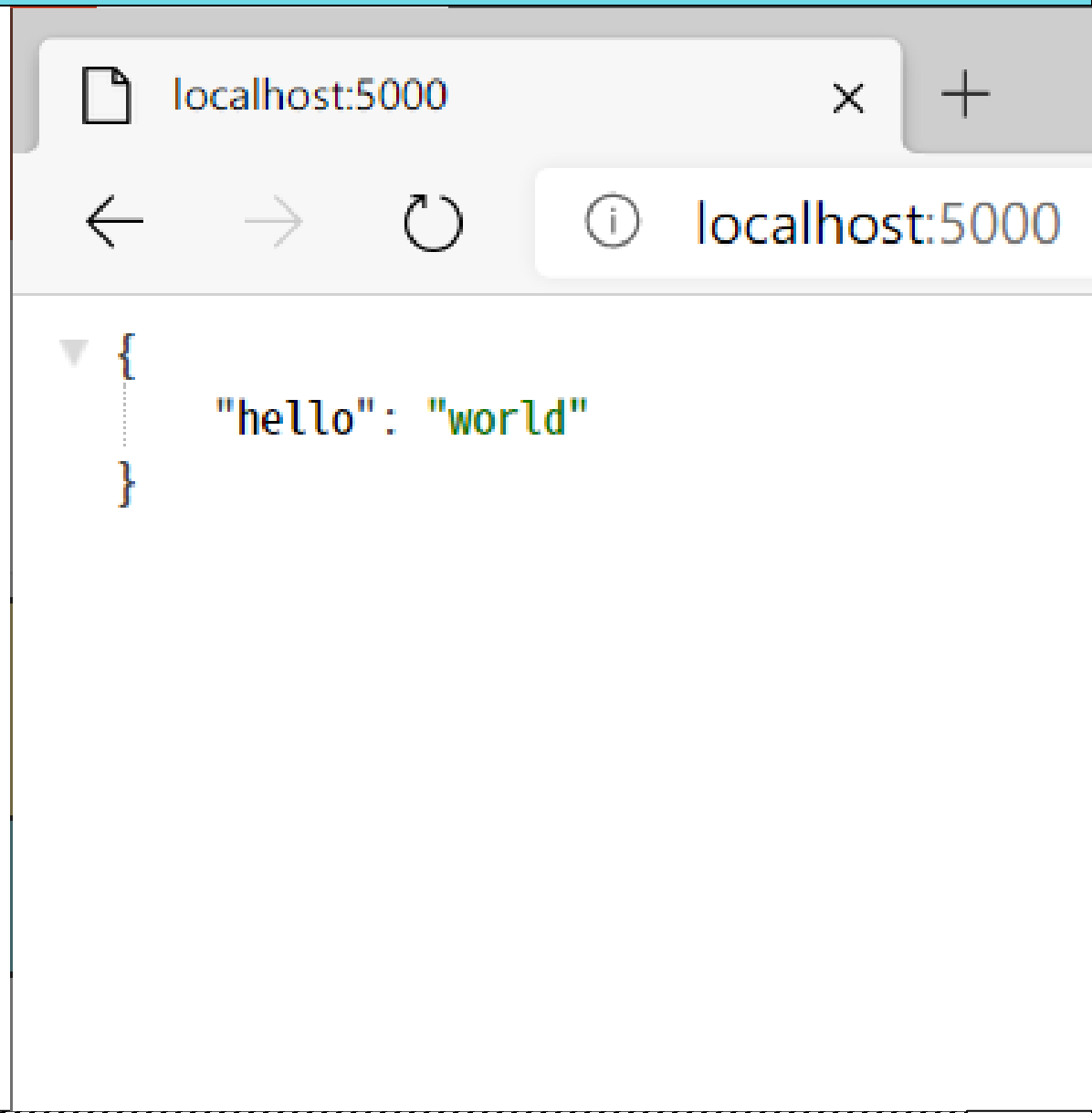
python3

```
~/Workspace/course » python3 api.py                                    valency@aorus-master
 * Serving Flask app "api" (lazy loadi
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [05/May/2020 17:12:53] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/May/2020 17:12:53] "GET /favicon.ico HTTP/1.1" 404 -
```



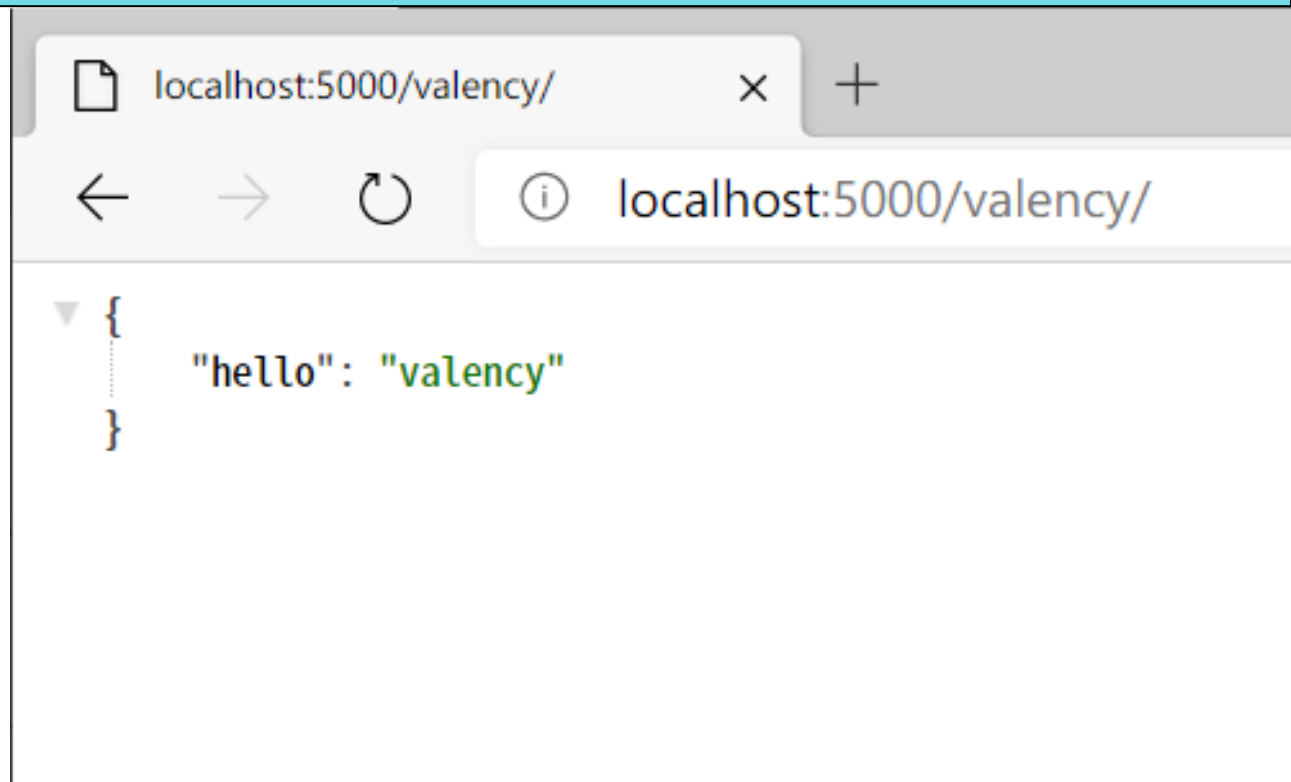
❖ 使用 JSON 与客户端通信

```
from flask import Flask, jsonify
...
@app.route('/')
def hello():
    return jsonify({
        'hello': 'world'
    })
...
```



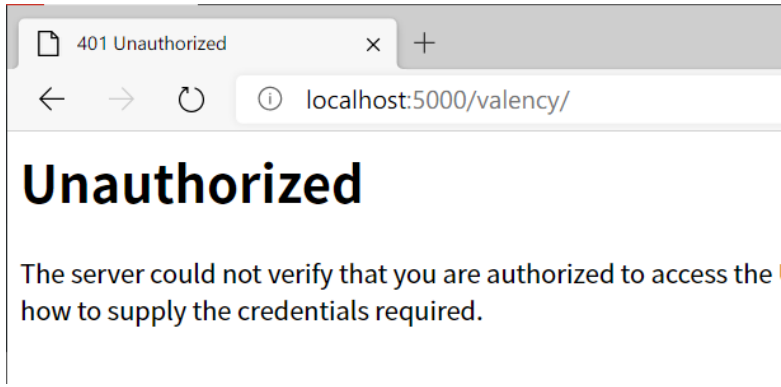
❖ 基础路由（Routing）

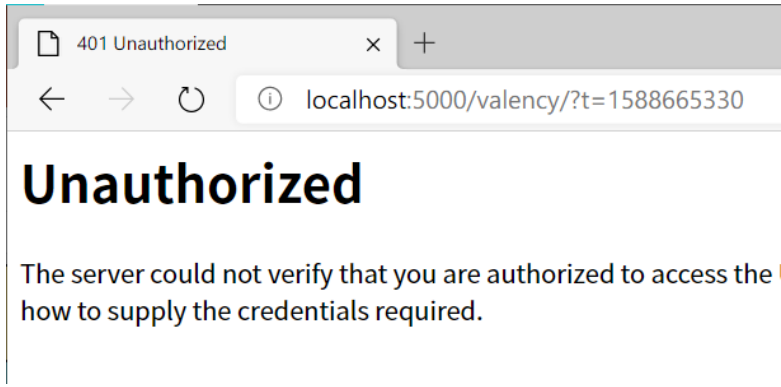
```
...
@app.route('/<string:name>/')
def hello(name: str = None):
    return jsonify({
        'hello': name
    })
...
```

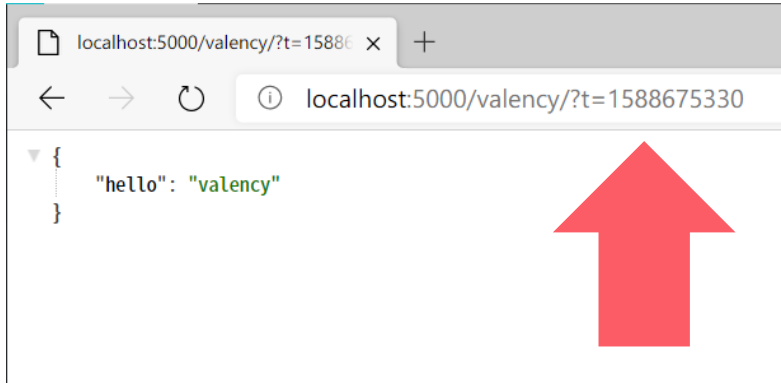


❖ 处理客户端请求参数（GET）

```python
from datetime import datetime
from flask import Flask, jsonify, request, abort

...
def hello(name: str = None):
    t = request.args.get('t')
    if t is not None and datetime.now().timestamp() - int(t) < 3600:
        return jsonify({
            'hello': name
        })
    else:
        abort(401)
...
```

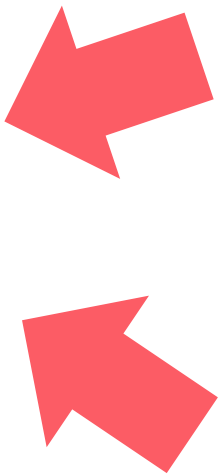




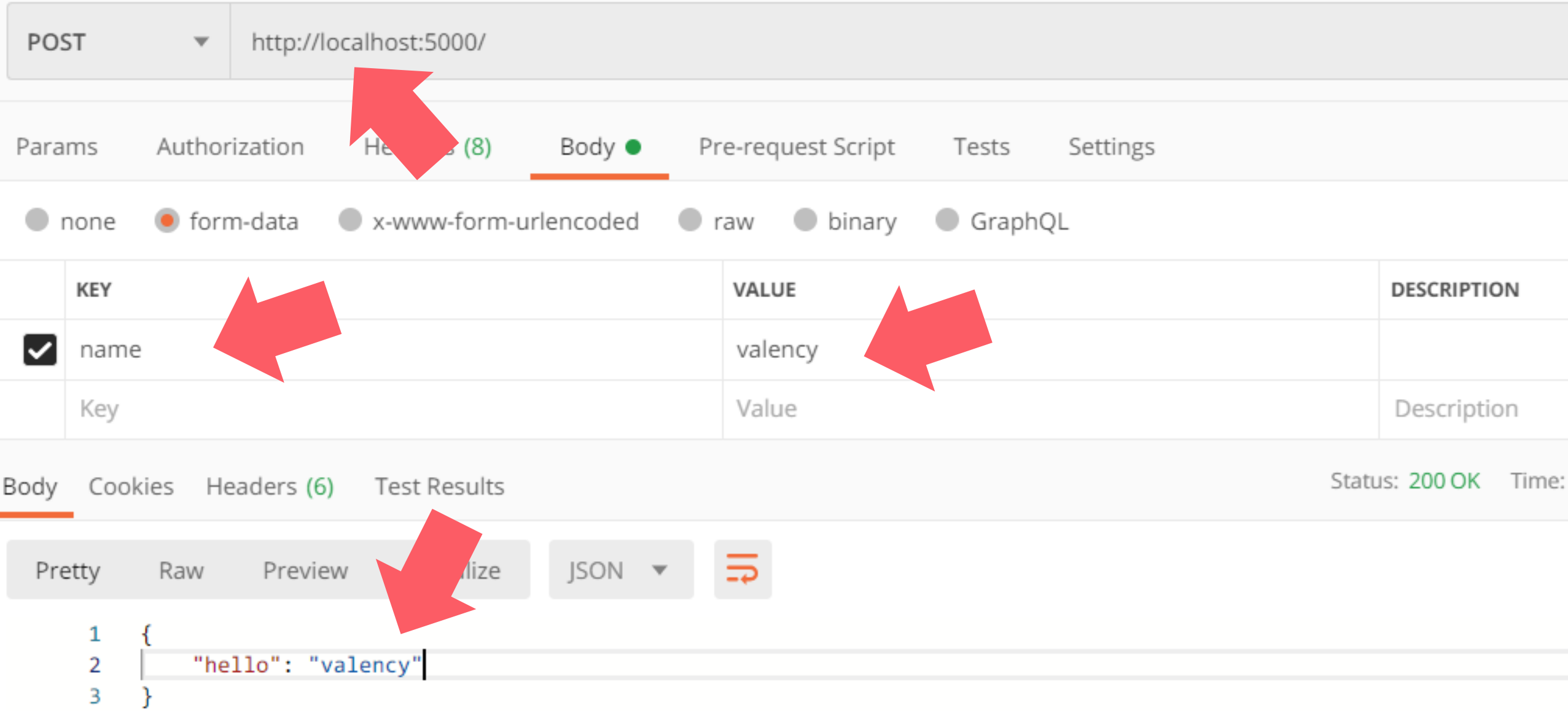


❖ 处理客户端请求参数（POST）

```
from flask import Flask, jsonify, request, abort
...
@app.route("/", methods=('POST',))
def user_create():
    name = request.form.get('name')
    if name:
        return jsonify({
            'hello': name
        })
    else:
        abort(400)
...
```



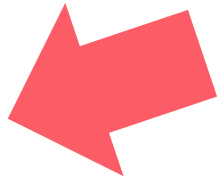
Flask 使用指南

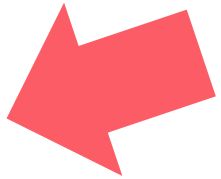

POST
http://localhost:5000/
Params
Authorization
(8)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
KEY
VALUE
DESCRIPTION
name
valency
Key
Value
Description
Body
Cookies
Headers (6)
Test Results
Status: 200 OK
Time:
Pretty
Raw
Preview
JSON
1 {
2 "hello": "valency"
3 }

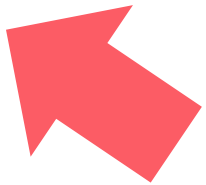
Flask 使用指南

❖ 连接数据库

```
...
engine = create_engine('sqlite:///rest.db')
Base = declarative_base()
...
class User(Base):
    ...
...
if __name__ == "__main__":
    Base.metadata.create_all(engine)
    Session = sessionmaker(bind=engine)
    app.run()
...
```

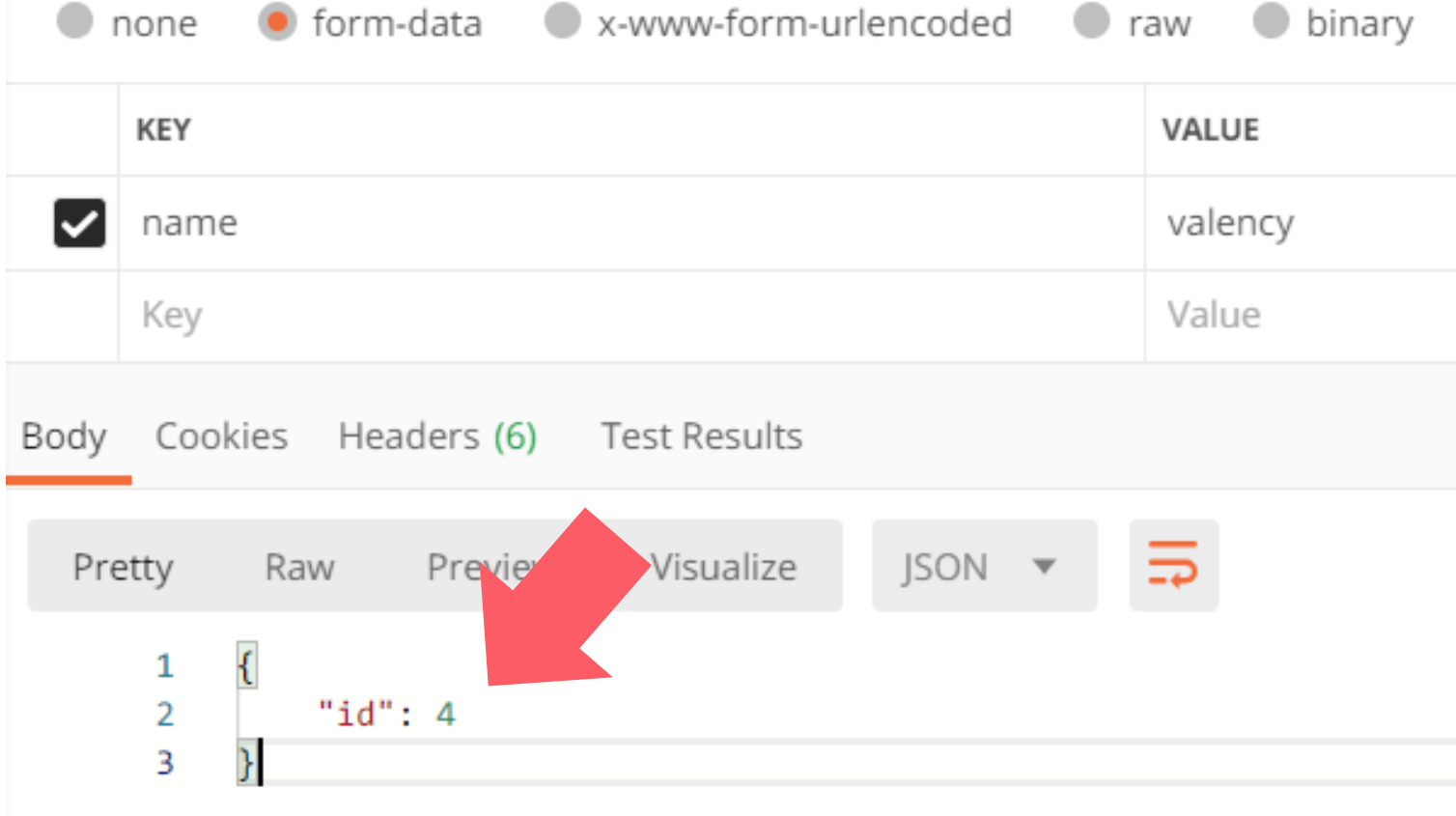




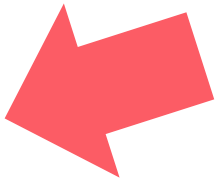


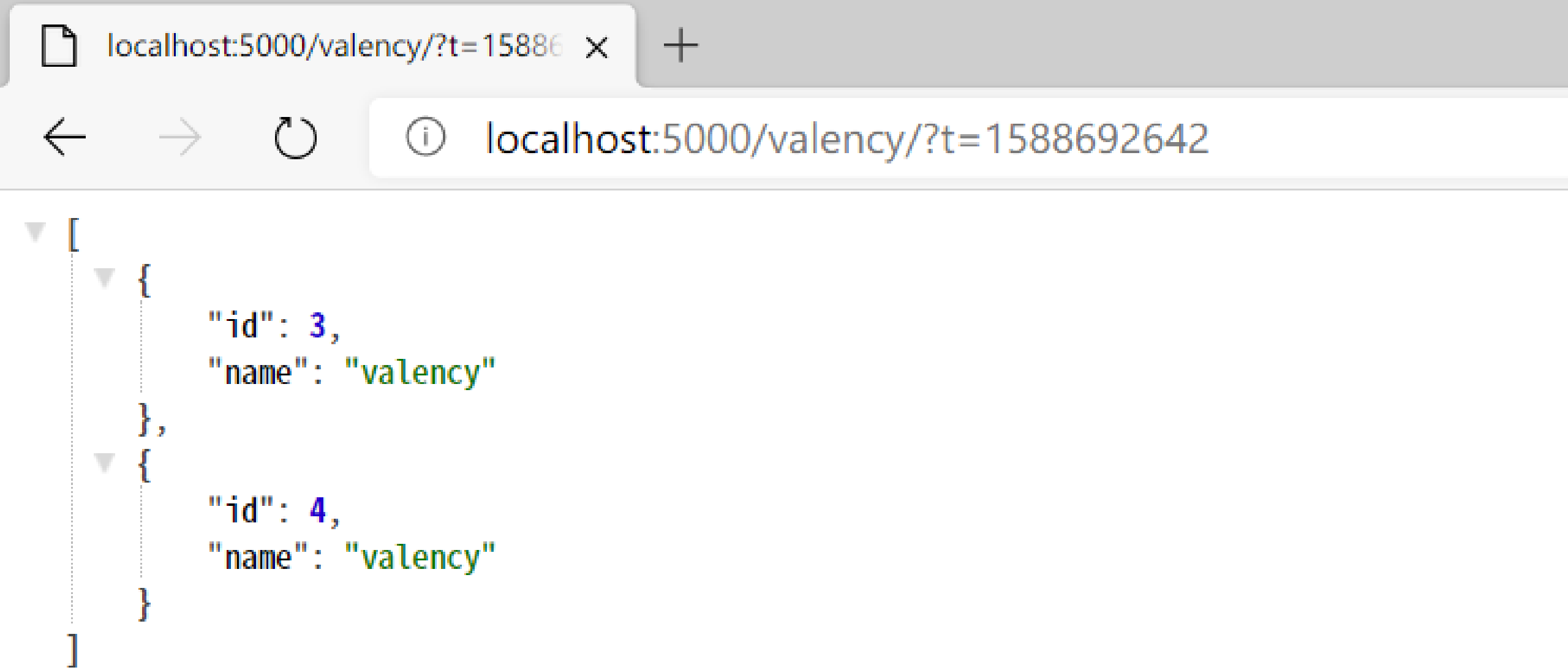
Flask 使用指南

```
...
@app.route("/", methods=('POST',))
def user_create():
    name = request.form.get('name')
    if name:
        session = Session()
        user = User(name=name)
        session.add(user)
        session.commit()
        return jsonify({
            'id': user.id
        })
    else:
        abort(400)
```



```
...


@app.route('/<string:name>/')
def hello(name: str = None):
    t = request.args.get('t')
    if t is not None and datetime.now().timestamp() - int(t) < 3600:
        return jsonify([{
            'id': i.id,
            'name': i.name
        } for i in Session().query(User).filter(User.name == name)])
    else:
        abort(401)


...
```




localhost:5000/valency/?t=15886
localhost:5000/valency/?t=1588692642
[
{
"id": 3,
"name": "valency"
},
{
"id": 4,
"name": "valency"
}
]

- Flask 官方教程：
- https://flask.palletsprojects.com/

- Flask 快速入门：
- https://www.jianshu.com/p/cc916366567e/

Express

Express 简介

- Express
- https://expressjs.com/

Express 4.17.1

Fast, unopinionated, minimalist web framework for Node.js

- 一个使用 JavaScript 编写的互联网应用服务器框架
- Express 提供基本的 API 框架，也提供对 UI 框架的支持
- Express.js 由 TJ Holowaychuk 首次发布于 2010 年 5 月 22 日
- 2016 年 1 月，IBM 宣布将 Express 置于 Node.js 基金会孵化器的管理之下
- Express 已经是 Node.js 服务器框架的事实标准
- 大部分使用 JavaScript 开发的 API 和 UI 框架都是基于 Express 的
- 包括著名的 UI 框架 Vue 和 React

Express 使用指南

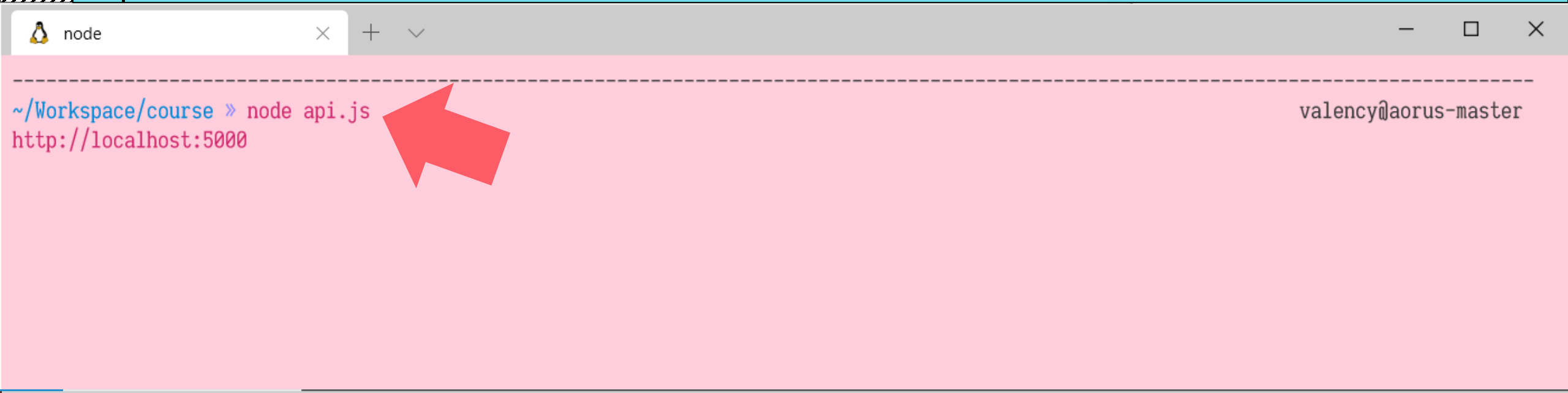
❖ 安装 Express

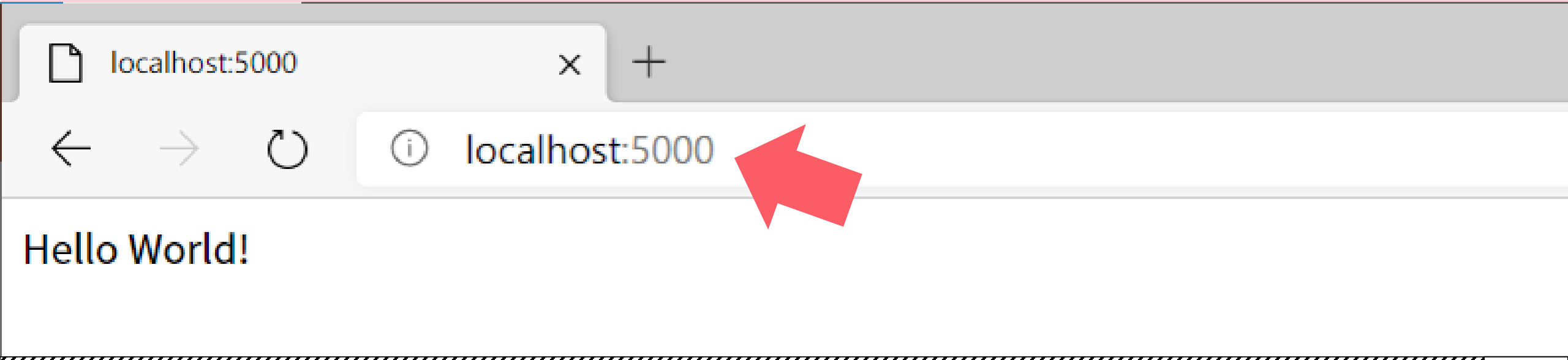
❖ npm i --save express

❖ 测试 Express

```
const express = require('express')
const app = express()
const port = 5000

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => console.log(`http://localhost:${port}`))
```

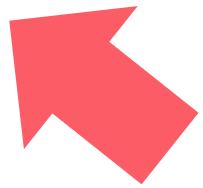
Express 使用指南

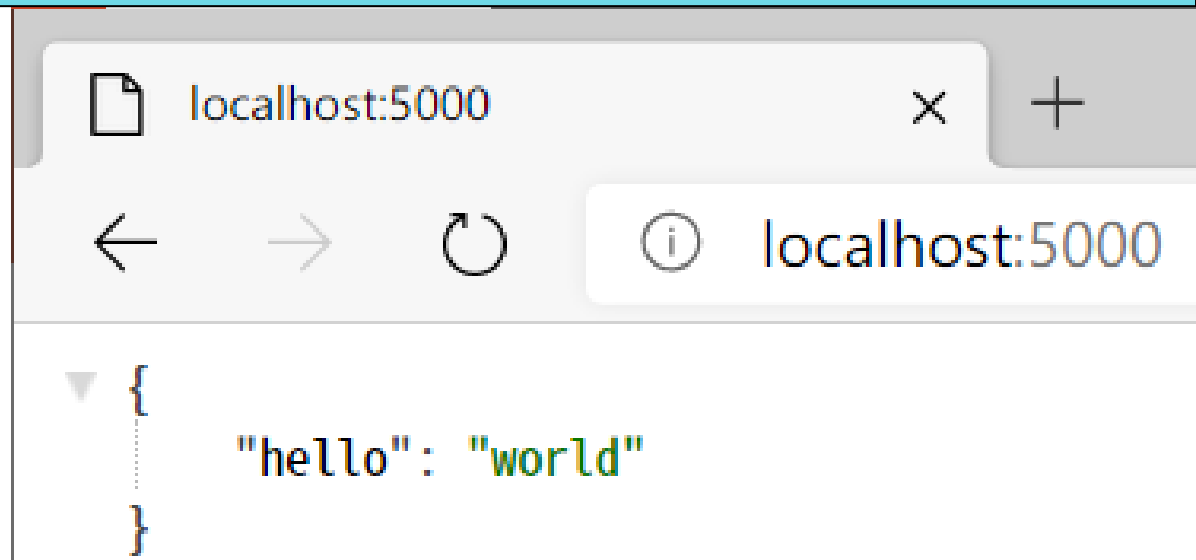




❖ 使用 JSON 与客户端通信

```
...

app.get('/', (req, res) => res.send({
    'hello': 'world'
}))

...
```

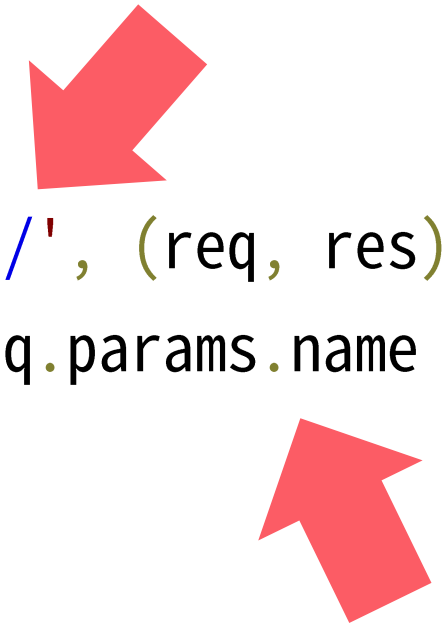


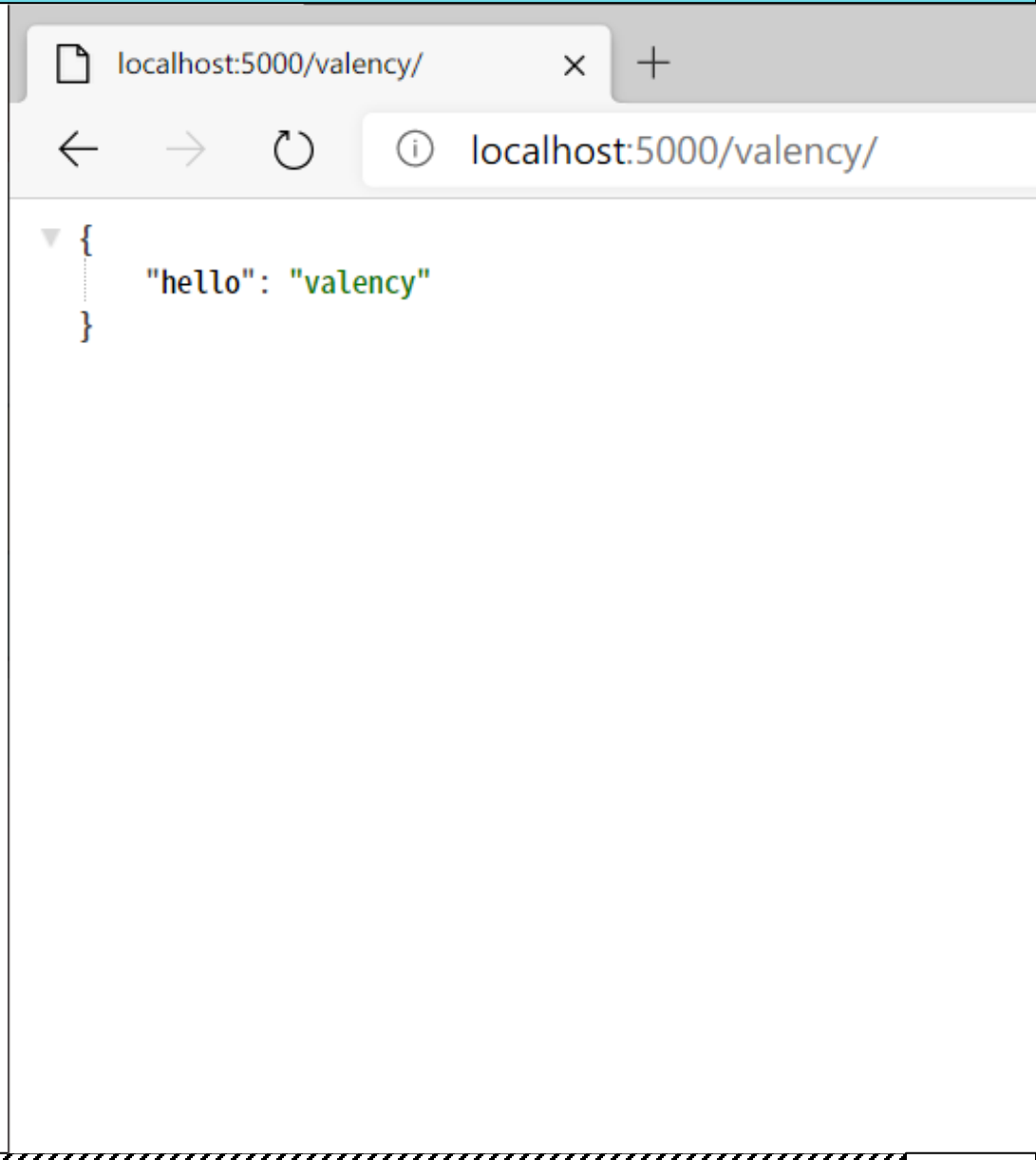


Express 使用指南

❖ 基础路由（Routing）

```
...
app.get('/:name/', (req, res) => res.send({
    'hello': req.params.name
}))
...
```

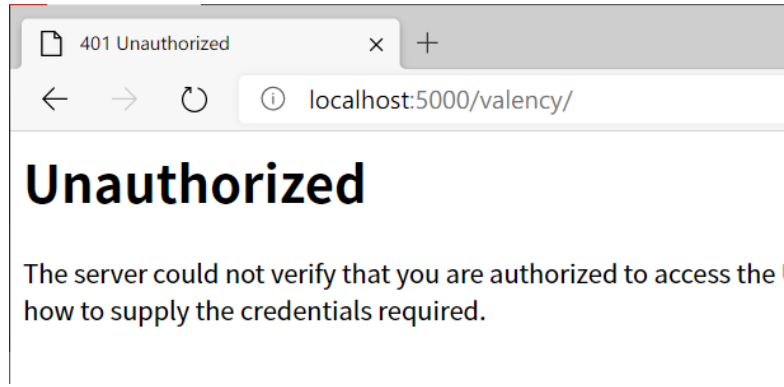


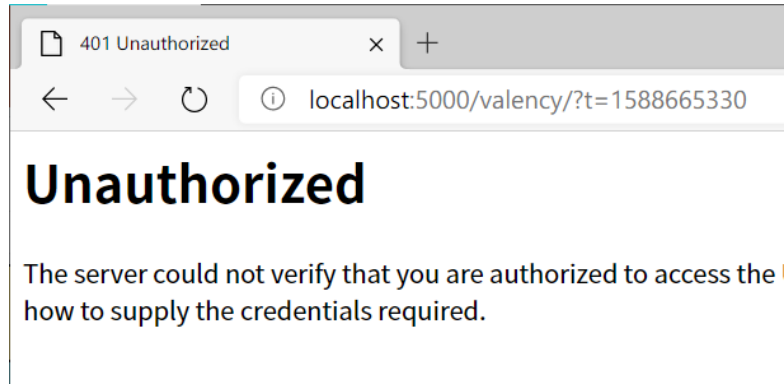


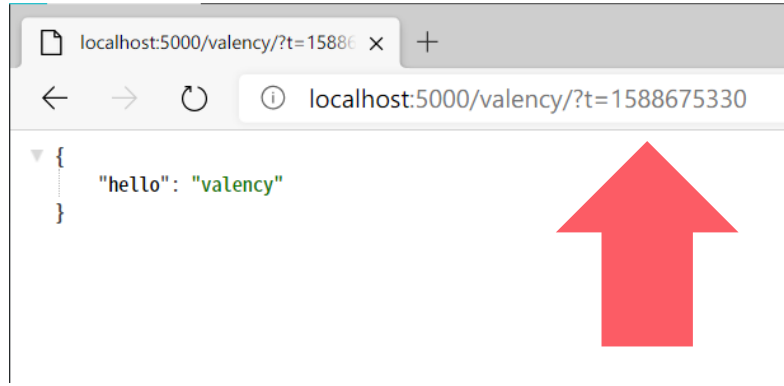
Express 使用指南

❖ 处理客户端请求参数（GET）

```
...
app.get('/:name/', (req, res) => {
    let name = req.params.name
    let t = req.query.t
    if (t && new Date().getTime() / 1000 - parseInt(t) < 3600) {
        res.send({
            'hello': name
        })
    } else {
        res.status(401).send()
    }
})
...
```

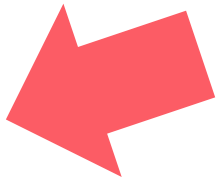


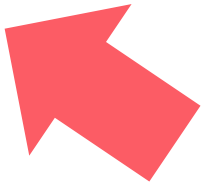




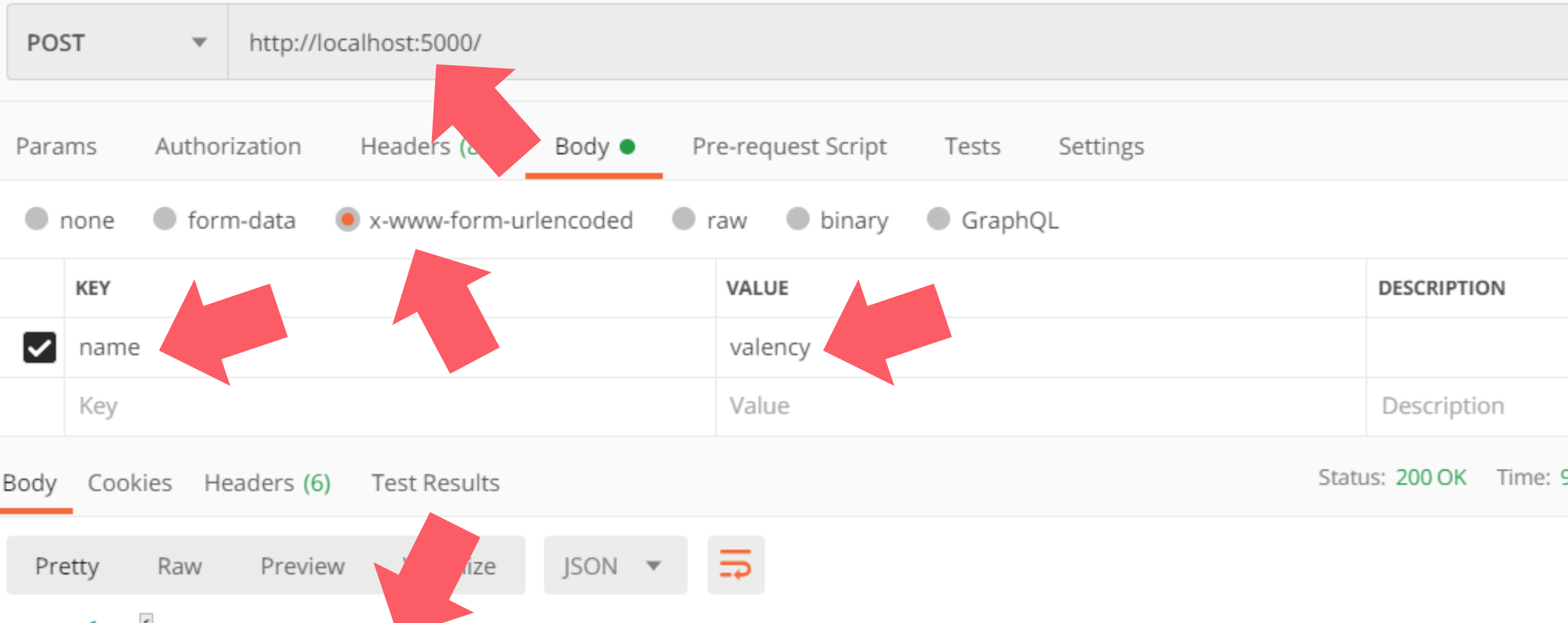
❖ 处理客户端请求参数（POST）

```
...
app.use(express.json());
app.use(express.urlencoded());
app.post('/', (req, res) => {
    let name = req.body.name
    if (name) {
        res.send({
            'hello': name
        })
    } else {
        res.status(400).send()
    }
})
```





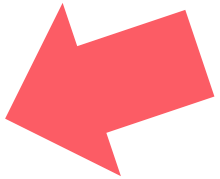
Express 使用指南

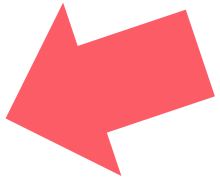


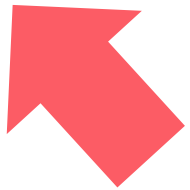
Express 使用指南

❖ 连接数据库

```
...
const {Sequelize, Model, DataTypes} = require('sequelize');
const sequelize = new Sequelize('sqlite:rest.db');

class User extends Model {
}

User.init({
    name: DataTypes.STRING
}, {sequelize, modelName: 'user'});

...
```

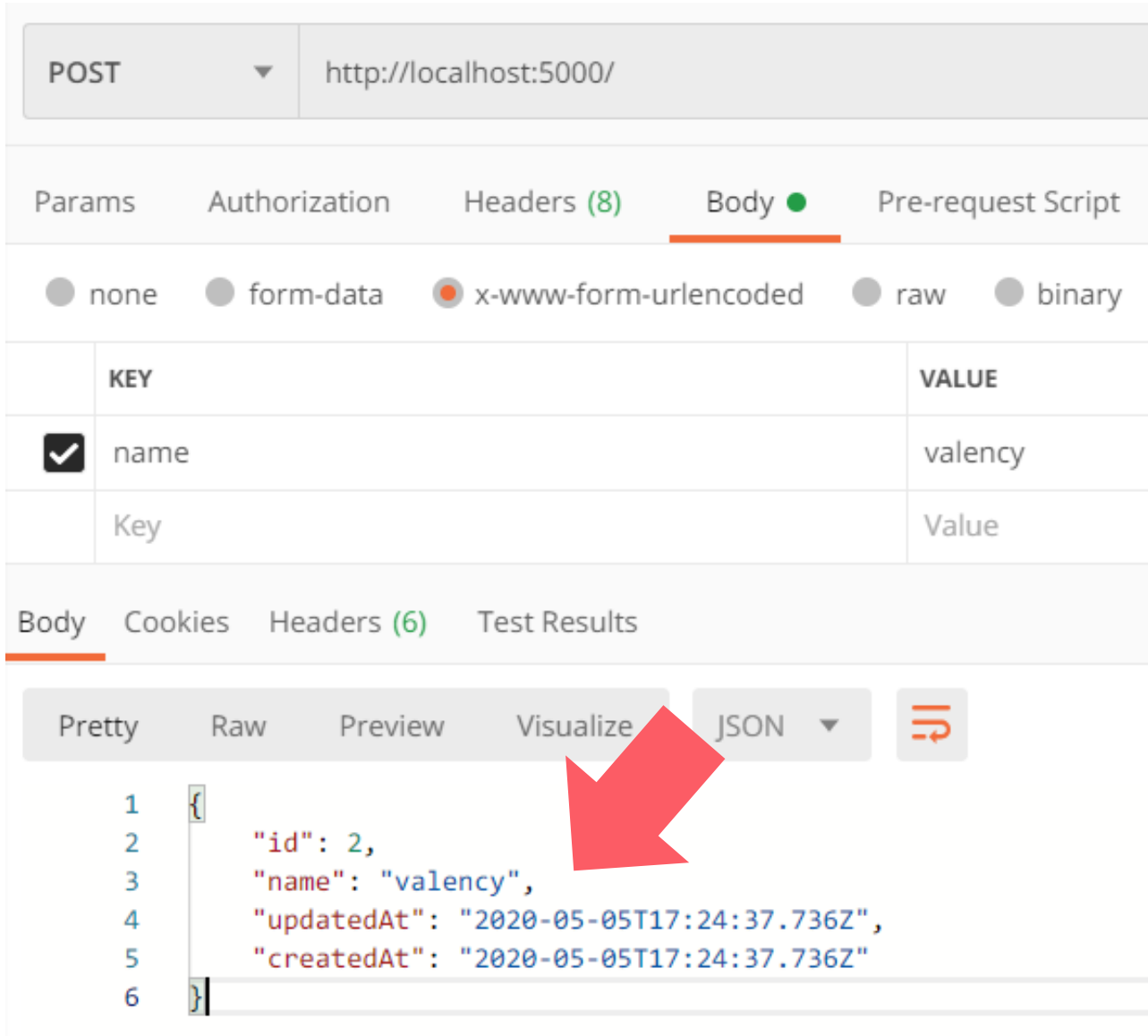






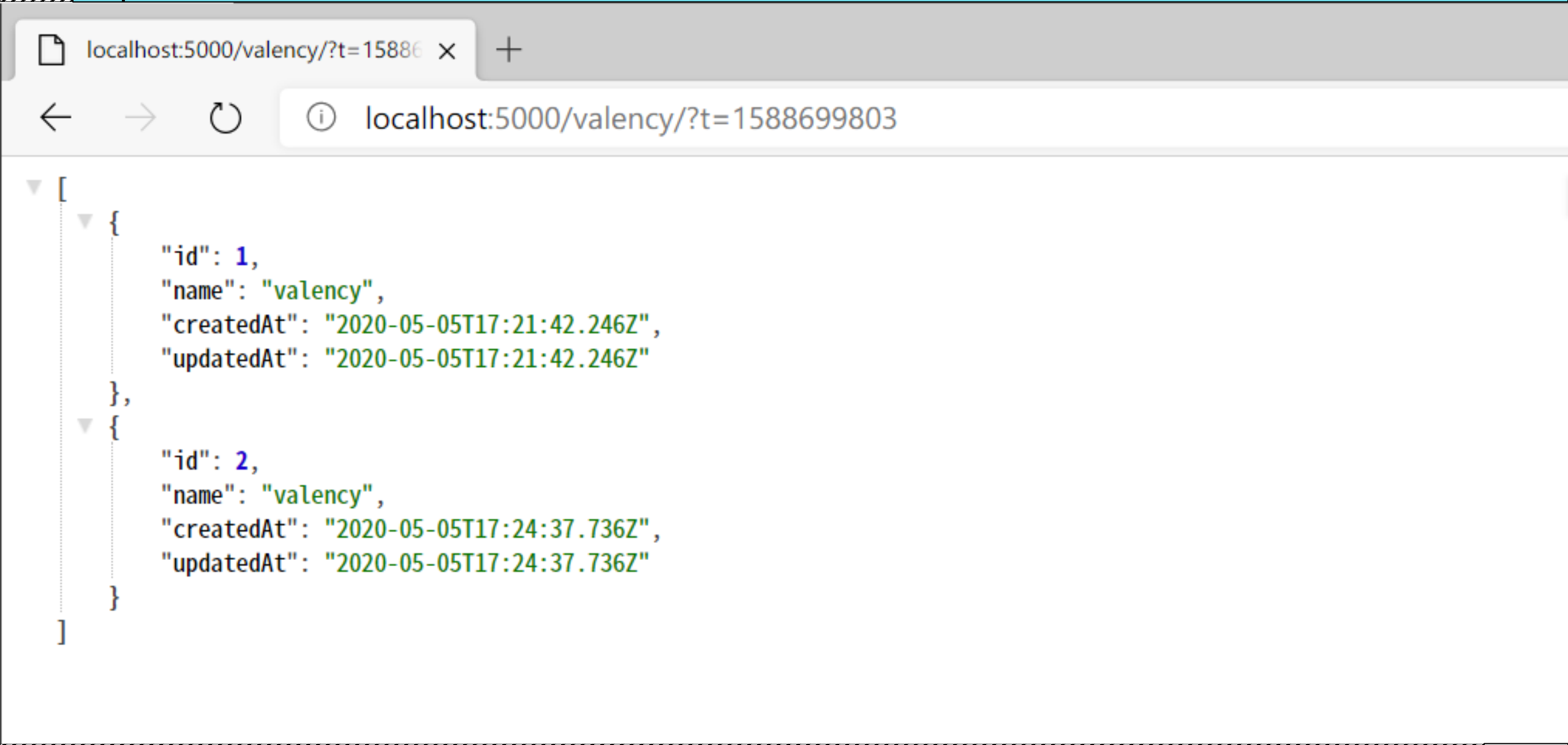
Express 使用指南

```
...
app.post('/', (req, res) => {
    let name = req.body.name
    if (name) {
        sequelize.sync().then(() => User.create({
            name: name
        })).then((user) => {
            res.send(user)
        })
    } else {
        res.status(400).send()
    }
})
...
```



```
...

app.get('/:name/', (req, res) => {
    let name = req.params.name
    let t = req.query.t
    if (t && new Date().getTime() / 1000 - parseInt(t) < 3600) {
        User.findAll({where: {name: name}}).then(data => {
            res.send(data)
        });
    } else {
        res.status(401).send()
    }
})
```

Express 使用指南



Express 进阶

- Express 官方教程：
- https://expressjs.com/en/starter/installing.html

- Express 简明教程：
- https://www.jianshu.com/p/18f06a4ac4dd

Imagine what you could build if you learned Ruby on Rails...

Learning to build a modern web application is daunting. Ruby on Rails makes it much easier and more fun. It includes everything you need to build fantastic applications, and you can learn it with the support of our large, friendly community.


Gin Web Framework
Documentation
Blog
GitHub
English
Search this site...
Gin Web Framework
Learn More
Download
The fastest full-featured web framework for Go. Crystal clear.

Type Safe

Forget about stringly typed objects, from request to response, everything has types.

Feature Rich

Actix provides a lot of features out of box. HTTP/2, logging, etc.

```
use actix_web::{web, App, HttpRequest, HttpServer, Responder};

async fn greet(req: HttpRequest) -> impl Responder {
    let name = req.match_info().get("name").unwrap_or("World");
    format!("Hello {}!", &name)
}

#[actix_rt::main]
```

Laravel

Documentation Forge Ecosystem News Partners

Search Docs

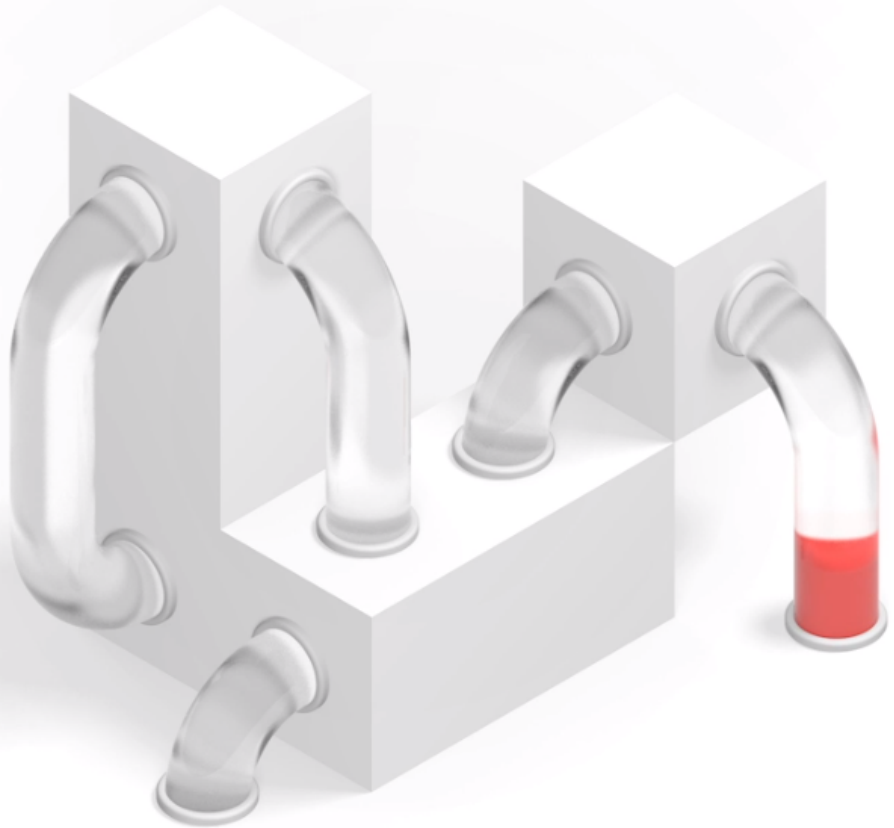
Laravel Vapor is now available! Sign up today! →

The PHP Framework for Web Artisans

Laravel is a web application framework with expressive, elegant syntax. We've already laid the foundation — freeing you to create without sweating the small things.

Documentation

Watch Laracasts



Spring makes Java cloud-ready.

WHY SPRING QUICKSTART

NEWS Announcing: The NEW Spring Website! | SpringOne 2020 goes virtual

What Spring can do

ASP.NET MVC Pattern

A design pattern for achieving a clean separation of concerns

Get Started

Supported on Windows, Linux, and macOS

Model View Controller (MVC)

MVC is a design pattern used to decouple user-interface (view), data (model), and application logic (controller). This pattern helps to achieve separation of concerns.

总结

- 适合云计算和大数据应用的互联网应用程序框架（Web Framework）
- 接口统一：方便随时更换框架，而不影响整体架构
- 前后端分离：采用 REST 或类似的架构，适合小团队开发，并可通过微服务部署
- 无状态：单一容器或服务器崩溃不会影响整体服务状态，可以随时增加、删除微服务
- MVC：采用 MVC（Model View Controller）或类似的架构，提高开发效率
- 结构简单：支持 CI/CD 平台，容易实施自动化测试

作业

- 使用任意一个 API 框架撰写一个简单的用户管理系统
- 不限制编程语言，可以使用 Python、JavaScript、Ruby、Rust、Go、Java 等
- 至少需要包含以下几个功能：
- 注册：POST 一套用户名和密码，在数据库中写入该用户名和密码，并返回用户 ID
- 登录：GET 一套用户名和密码，在数据库中更新该用户的 Token，并返回该 Token
- 验证登录状态：GET 一套用户 ID 和 Token，比对数据库中的记录，并返回是否正确
- 登出：PUT 一套用户 ID 和 Token，将数据库中对应的 Token 记录移除

- 用户管理系统中会出现很多错误处理的问题
- 例如：用户名重复、登录密码错误、Token 过期、用户名不合法等
- 如有兴趣，可以自行处理这些错误

作业

- 示范案例
- 注册：POST 一套用户名和密码，在数据库中写入该用户名和密码，并返回用户 ID
- POST(user=a, pwd=81dc9bdb52d04dc20036dbd8313ed055)
- RETURN(status=201,data={
- id:'1',
- user: 'a'
- })

作业

- 示范案例
- 登录：GET 一套用户名和密码，在数据库中更新该用户的 Token，并返回该 Token
- GET(user=a, pwd=81dc9bdb52d04dc20036dbd8313ed055)
- RETURN(status=200,data={
- id:'1',
- token: 'f37a6071-0b97-465c-bf88-6eae6e12991c'
- })

- 生成随机 Token 可以使用 UUID：https://baike.baidu.com/item/UUID/5921266

作业

- 示范案例
- 验证登录状态：GET 一套用户名和 Token，比对数据库中的记录，并返回是否正确
- GET(id=1, token=f37a6071-0b97-465c-bf88-6eae6e12991c)
- RETURN(status=200)

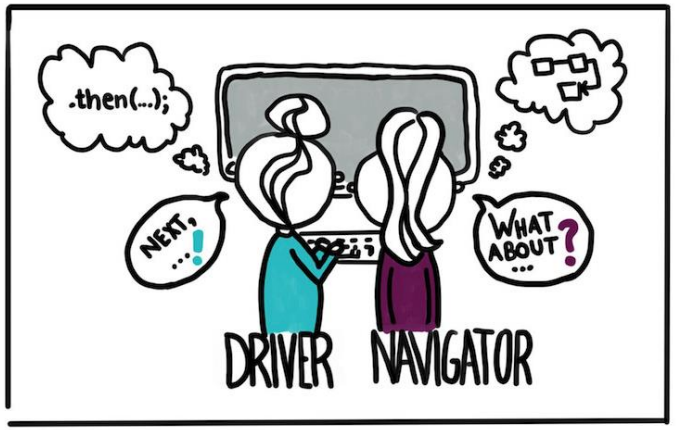
- GET(id=1, token=something-is-wrong)
- RETURN(status=401)

作业

- 示范案例
- 登出：PUT 一套用户名和 Token，将数据库中对应的 Token 记录移除
- PUT(id=1, token=f37a6071-0b97-465c-bf88-6eae6e12991c)
- RETURN(status=204)

作业

- 结对编程（Pair Programming）
- 是一种敏捷软件开发的方法，两个程序员在一个计算机上共同工作。一个人输入代码，而另一个人审查他输入的每一行代码。输入代码的人称作驾驶员（Driver），审查代码的人称作导航员（Navigator）。两个程序员经常互换角色。
- 在结对编程中，观察员同时考虑工作的战略性方向，提出改进的意见，或将来可能出现的问题以便处理。这样使得驾驶者可以集中全部注意力在完成当前任务的“战术”方面。观察员当作安全网和指南。结对编程对开发程序有很多好处。比如增加纪律性，写出更好的代码等。
- 结对编程是极限编程的组成部分。



- 考虑到本次实验较为复杂，两人一组提交实验报告即可
- 注意：两人都需要提交实验报告，内容务必保持一致
- 如果不愿意结对编程，也可以一人独立完成实验

作业

- 在作业系统中下载并完成本实验课对应实验报告
- https://hw.dgut.edu.cn/
- 注意：所有标识为 * 的地方都需要填写
- 截止日期：2024-05-20 23:59:59

课程名称：云计算与大数据应用开发　　学期：2023 年春季

实验名称	虚拟化技术			实验序号	1
姓　名	***	学　号	***	班　级	***
实验地点	***	实验日期	***	指导老师	丁烨
教师评语	-			实验成绩	-
				分制	100
同组同学	无				

四、　实验作业及分析

4.1　实验过程

1) *** 请将详细实验过程填写在此处 ***

4.2　实验结果

*** 请将实验结果截图填写在此处 ***

五、　实验总结

*** 请撰写一段 200 字左右的实验总结 ***


GOOD
LUCK!