云计算与大数据应用开发

第六章：云计算应用开发（一）

丁烨
dingye@dgut.edu.cn
计算机科学与技术学院

2024-05-06

東莞理工學院
DONGGUAN UNIVERSITY OF TECHNOLOGY

目录

应用程序接口概述

REST 应用程序架构

- 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

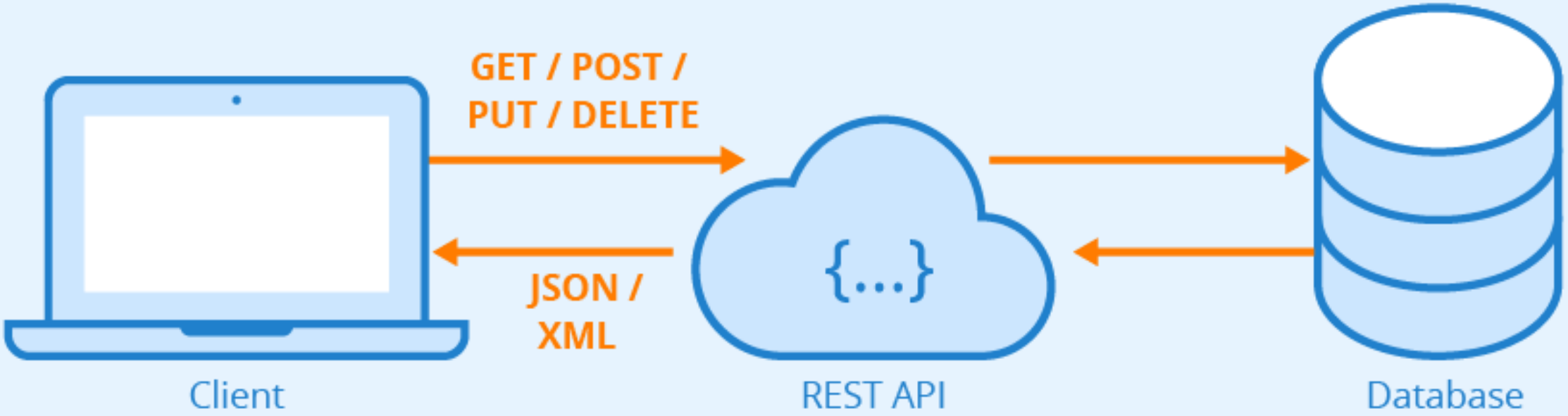
REST 应用程序架构

- 接口统一
- 客户端和服务端通常为 JSON 或 XML
- 服务端和数据库通常为 ORM

- 无状态
- 如有需要，客户端每次服务请求都需要携带验证信息
- 服务端每次数据请求都需要携带验证信息

应用程序接口概述

REST 应用程序架构



REST 应用程序架构

- REST 的优势
- 客户端、服务端、数据库可以任选架构，只要接口统一即可
- 客户端包括：PC 网页端、移动网页端，Windows / Mac / Linux 桌面应用程序、Android / iOS 移动端应用程序、微信小程序等
- 数据库包括：RDBMS 数据库，如 SQLite、MySQL、PostgreSQL、Oracle 等；NoSQL 数据库，如 Redis、MongoDB、Impala、Cassandra、HBase 等
- 服务端：即“应用程序接口”

- 不同的组件可以选择不同的运行环境，包括操作系统和硬件
- 不同的组件可以同时运行（例如，一个用户可以同时使用手机和电脑登陆“微信”）
- 只要接口统一即可

应用程序接口概述

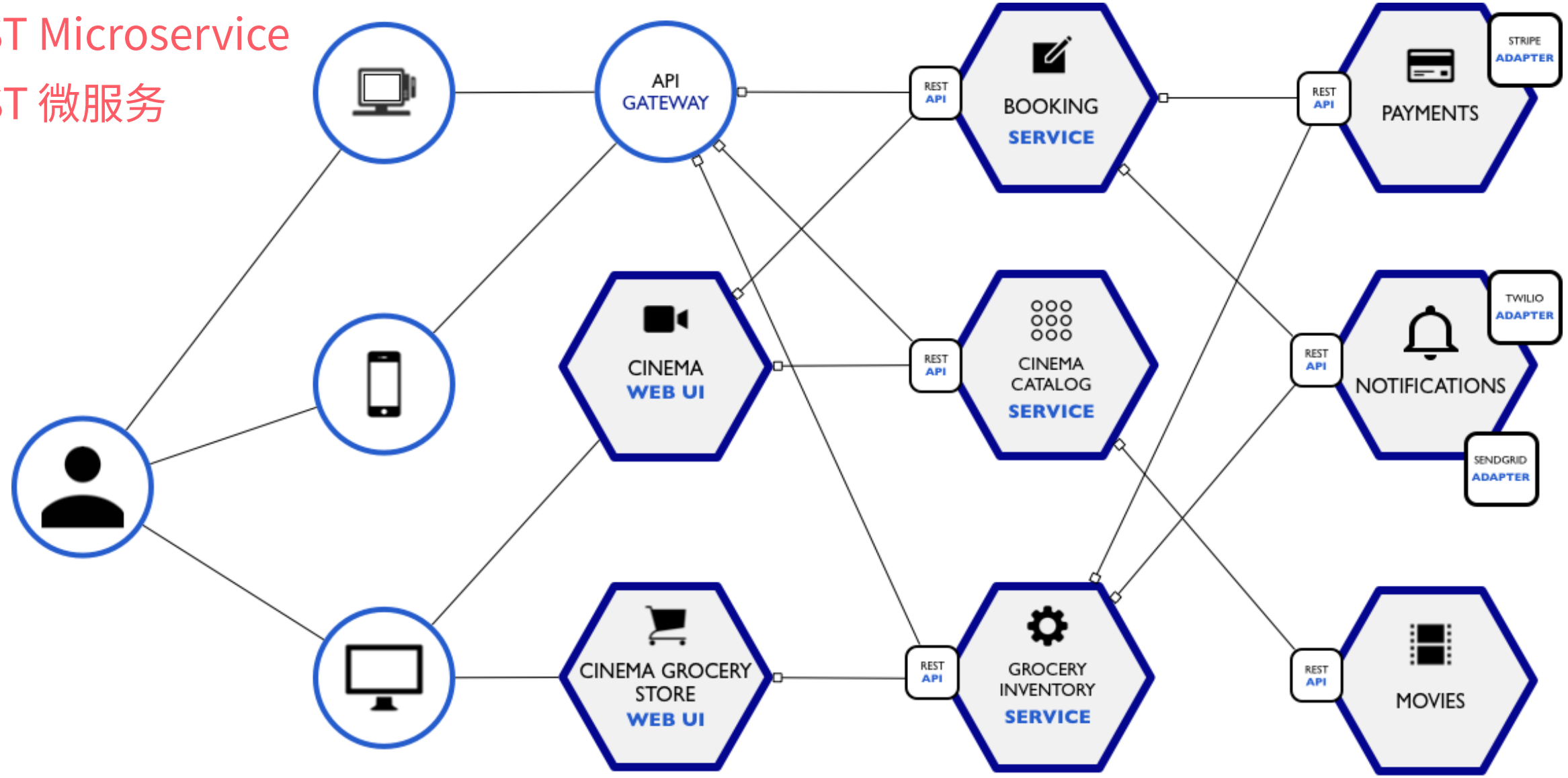
REST 应用程序架构

- REST 的优势
- REST 架构非常适合云计算时代的互联网应用
- 为了适应用户需求，大部分应用程序都包含不止一个终端
- REST 架构使得应用开发完全分离：前端工程师、后端工程师、数据库工程师
- Docker 为 REST 架构提供了非常方便的部署方案
- 在 REST 架构中，每个组件都可以非常方便的扩大服务规模（Scale Up）或减小服务规模（Scale Down）

应用程序接口概述

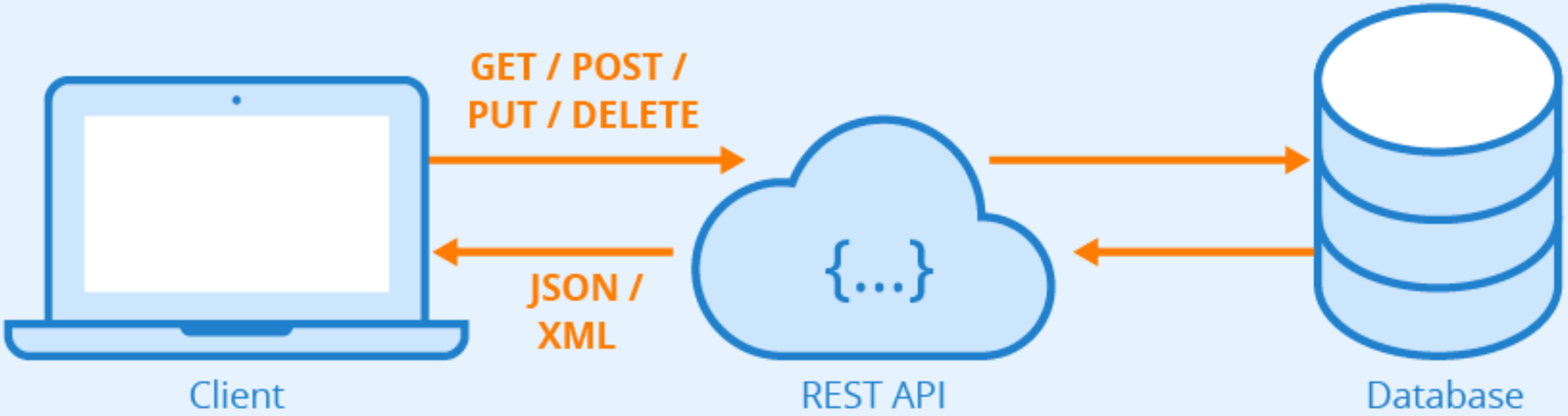
REST 应用程序架构

- REST Microservice
- REST 微服务



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

JSON

- JSON（JavaScript Object Notation）
- https://www.json.org/

- 是一种由道格拉斯·克罗克福特构想和设计的轻量级数据交换语言，该语言以易于让人阅读的文字为基础，用来传输由属性值或者序列性的值组成的数据对象
- JSON 保留了 C 的特性，大部分编程语言都可以方便的将对象转为 JSON（序列化，Serialize），或将 JSON 转为对象（反序列化，Deserialize）

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

❖ JSON 的序列化和反序列化（Python）

```
>>> import json
>>> json.dumps(['foo', {'bar': ('baz', None, 1.0, 2)}])
'["foo", {"bar": ["baz", null, 1.0, 2]}]'

>>> import json
>>> json.loads('["foo", {"bar":["baz", null, 1.0, 2]}]')
['foo', {'bar': ['baz', None, 1.0, 2]}]
```

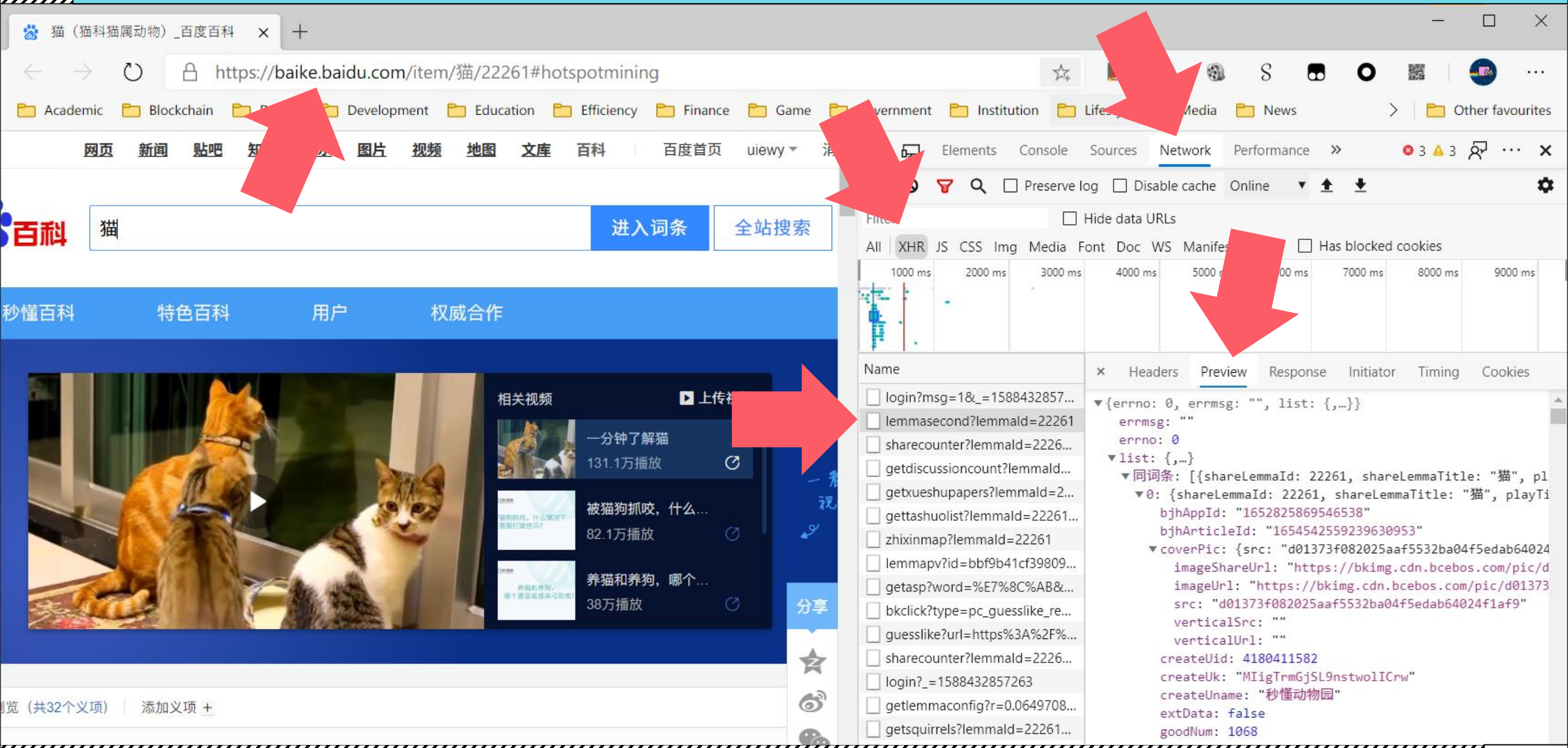
JSON

❖ JSON 的序列化和反序列化（JavaScript）

```
> var obj = { name: "John", age: 30, city: "New York" };
> var myJSON = JSON.stringify(obj);
> console.log(myJSON);
{"name":"John","age":30,"city":"New York"}

> var obj = JSON.parse('{ "name":"John", "age":30, "city":"New York"}');
> console.log(obj.name, obj.age);
John 30
```

应用程序接口概述

JSON



应用程序接口概述

JSON

- JSON 简明教程：
- https://www.w3schools.com/js/js_json_intro.asp
- https://www.jianshu.com/p/8b428e1d1564

- 在线 JSON 语法检查 / 美化器：
- http://json.parser.online.fr/

应用程序接口概述

ORM

- 对象关系映射（Object Relational Mapping，ORM）
- 一种程序设计技术，用于实现面向对象编程语言里不同类型系统的数据之间的转换
- 从效果上说，它其实是创建了一个可在编程语言里使用的“虚拟对象数据库”
- 使用 ORM 可以让数据查询更“自然”：不需要在一个编程语言中（例如 Python 或 JavaScript）拼装另一个编程语言（例如 SQL）

应用程序接口概述

ORM

sql中的表

创建表:

```
CREATE TABLE employee(
            id INT PRIMARY KEY auto_increment ,
            name VARCHAR (20),
            gender BIT default 1,
            birthday DATA ,
            department VARCHAR (20),
            salary DECIMAL (8,2) unsigned,
          );
```

sql中的表纪录

添加一条表纪录:

```
INSERT employee (name,gender,birthday,salary,department)
        VALUES   ("alex",1,"1985-12-12",8000,"保洁部");
```

查询一条表纪录:

```
SELECT * FROM employee WHERE age=24;
```

更新一条表纪录:

```
UPDATE employee SET birthday="1989-10-24" WHERE id=1;
```

删除一条表纪录:

```
DELETE FROM employee WHERE name="alex"
```

ORM

python的类

```
class Employee(models.Model):
    id=models.AutoField(primary_key=True)
    name=models.CharField(max_length=32)
    gender=models.BooleanField()
    birthday=models.DateField()
    department=models.CharField(max_length=32)
    salary=models.DecimalField(max_digits=8,decimal_places=2)
```

python的类对象

添加一条表纪录:

```
emp=Employee(name="alex",gender=True,birthday="1985-12-12",
        department="保洁部")
emp.save()
```

查询一条表纪录:

```
Employee.objects.filter(age=24)
```

更新一条表纪录:

```
Employee.objects.filter(id=1).update(birthday="1989-10-24")
```

删除一条表纪录:

```
Employee.objects.filter(name="alex").delete()
```

应用程序接口概述

ORM

- SQLAlchemy
- https://www.sqlalchemy.org/

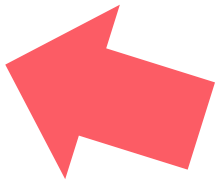
- 一个为 Python 设计的 ORM 框架

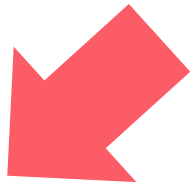
- 安装 SQLAlchemy：
- `pip install -U sqlalchemy`

应用程序接口概述

ORM

```
from datetime import datetime

from sqlalchemy import create_engine, Column, Integer, String
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

engine = create_engine('sqlite:///rest.db')
Base = declarative_base()


class User(Base):
    __tablename__ = 'user'
    id = Column(Integer, primary_key=True)
    name = Column(String)
```

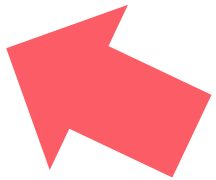


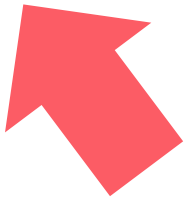


应用程序接口概述

ORM

```
if __name__ == '__main__':
    Base.metadata.create_all(engine)
    Session = sessionmaker(bind=engine)
    session = Session()

    user = User(name='Valency@{}'.format(datetime.now()))
    session.add(user)
    session.commit()

    for i in session.query(User).all():
        print(i.id, i.name)
```

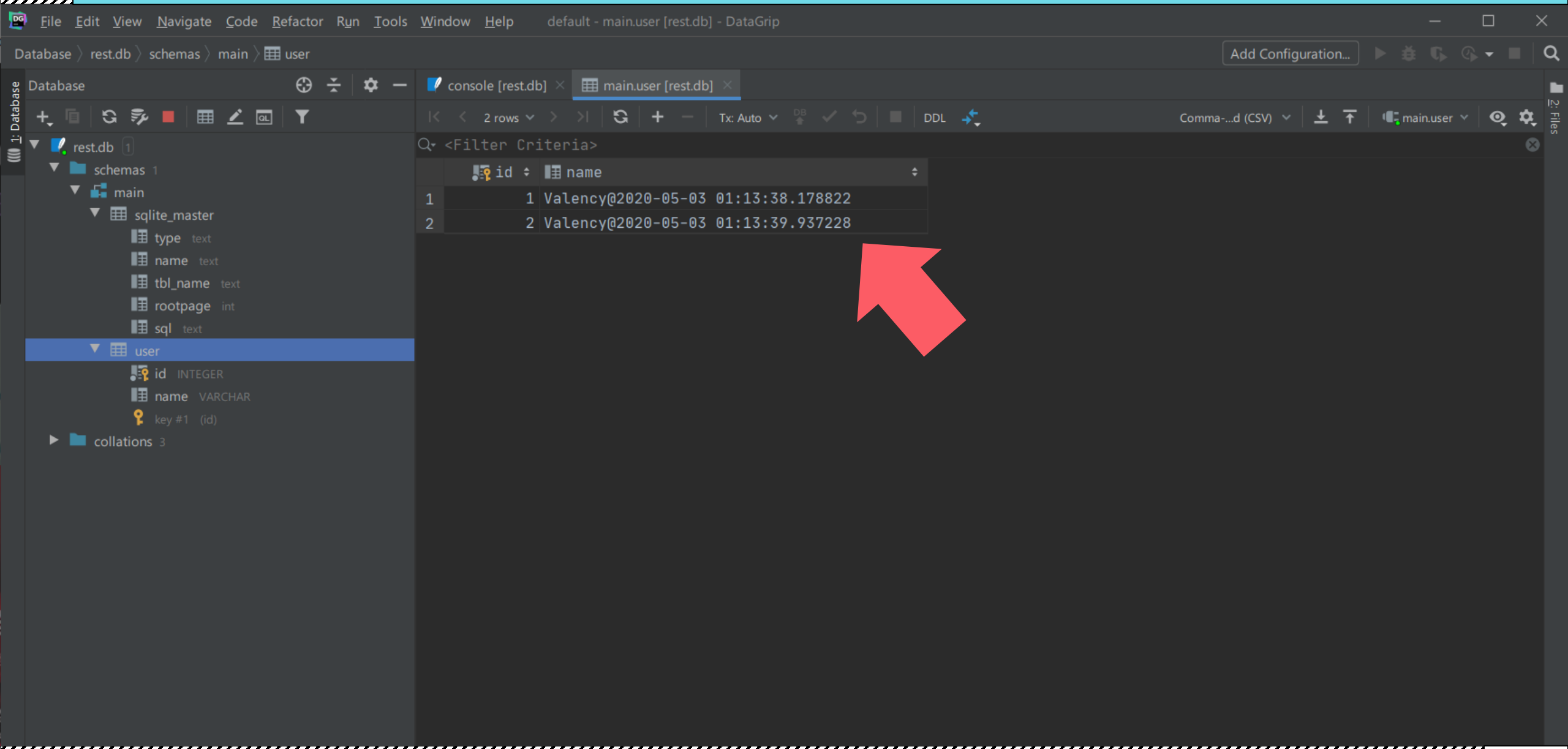




```
~/Workspace/course » python3 rest.py
1 Valency@2020-05-03 00:25:48.986668
2 Valency@2020-05-03 00:25:55.761180
```

应用程序接口概述

ORM



应用程序接口概述

ORM

- SQLAlchemy 简明教程：
- https://docs.sqlalchemy.org/en/13/orm/tutorial.html
- https://www.jianshu.com/p/264ceec89652

ORM

- Sequelize
- https://sequelize.org/

Sequelize

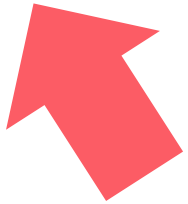
- 一个为 JavaScript 设计的 ORM 框架

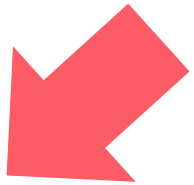
- 安装 Sequelize：
- `npm i --save sequelize`
- `npm i --save sqlite3`

应用程序接口概述

ORM

```
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'});
```

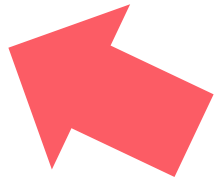


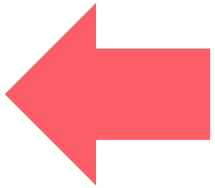


应用程序接口概述

ORM

```
sequelize.sync().then(() => User.create({
    name: 'Valency@' + new Date()
})).then(() => {
    User.findAll().then(data => {
        data.forEach(i => {
            console.log(i.id, i.name);
        });
    });
});
```





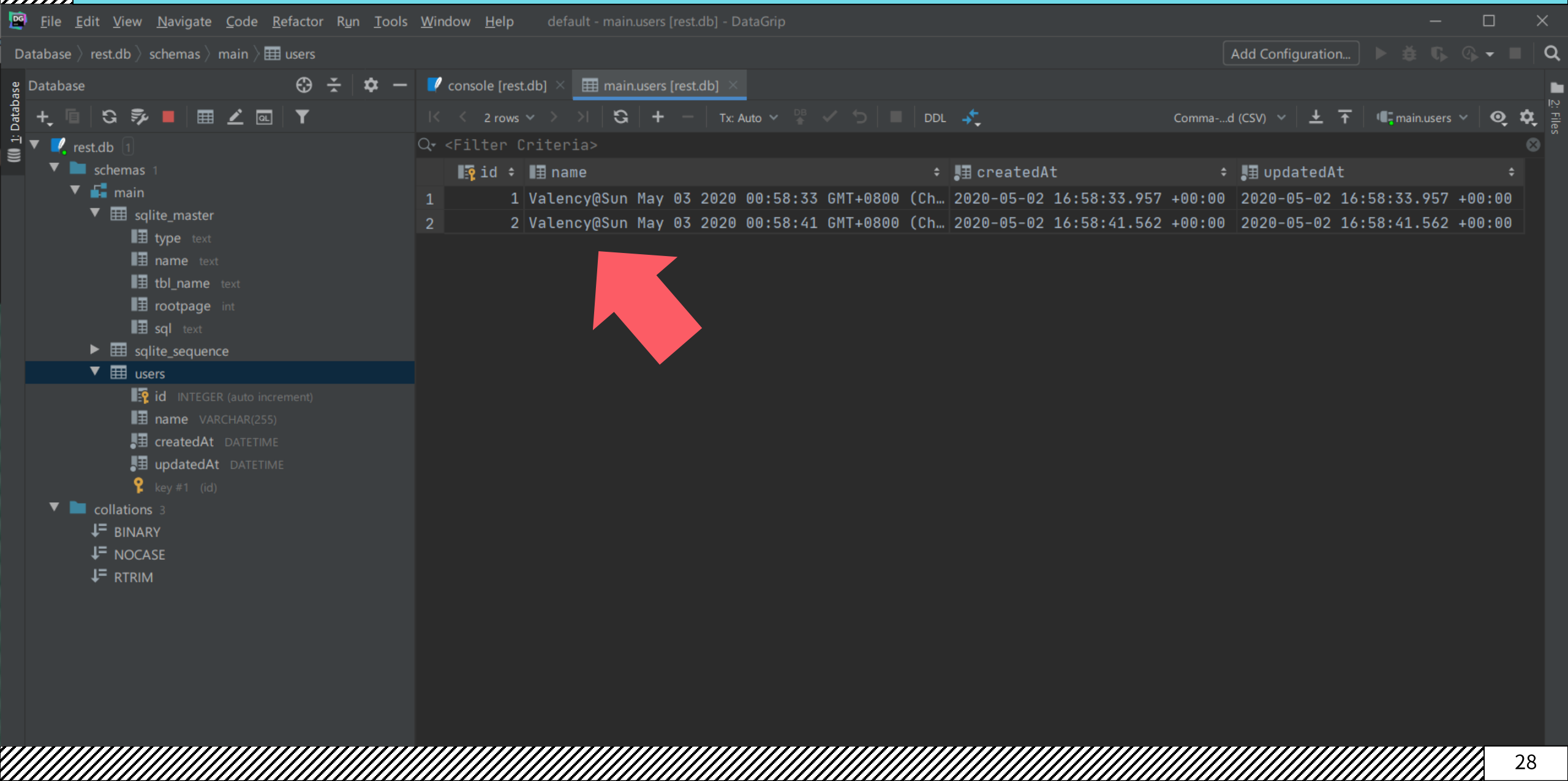
应用程序接口概述

ORM

```
~/Workspace/course » node rest.js
Executing (default): CREATE TABLE IF NOT EXISTS `users` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `name` VARCHAR(255) NOT NULL, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL);
Executing (default): PRAGMA INDEX_LIST(`users`)
Executing (default): INSERT INTO `users` (`id`,`name`,`createdAt`,`updatedAt`) VALUES (NULL,$1,$2,$3);
Executing (default): SELECT `id`, `name`, `createdAt`, `updatedAt` FROM `users` AS `user`;
1 Valency@Sun May 03 2020 00:58:33 GMT+0800 (China Standard Time)
2 Valency@Sun May 03 2020 00:58:41 GMT+0800 (China Standard Time)
```

应用程序接口概述

ORM



应用程序接口概述

ORM

- Sequelize 简明教程：
- https://sequelize.org/v5/manual/getting-started.html
- https://www.jianshu.com/p/9059f42477ab

目录

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

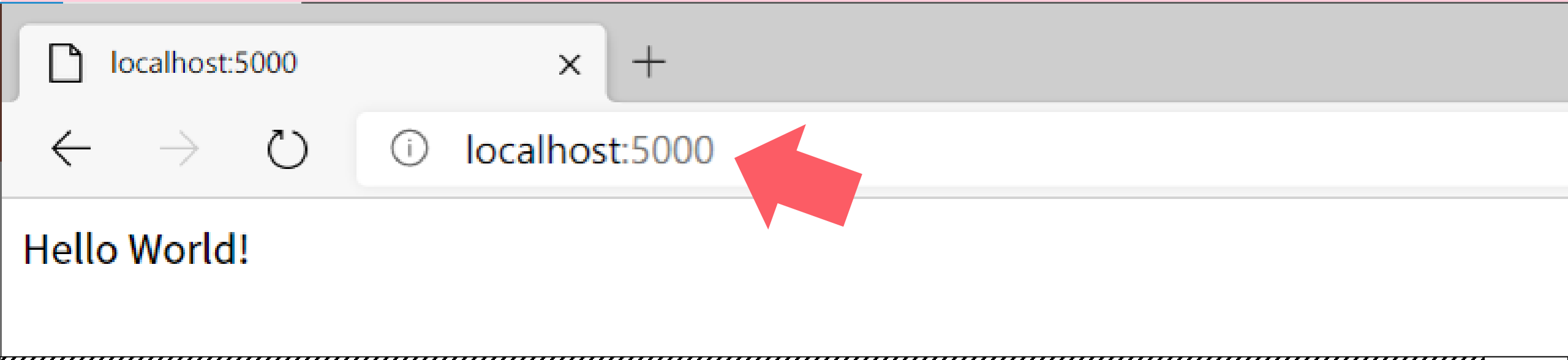
- 安装 Flask
- `pip3 install flask`

❖ 测试 Flask

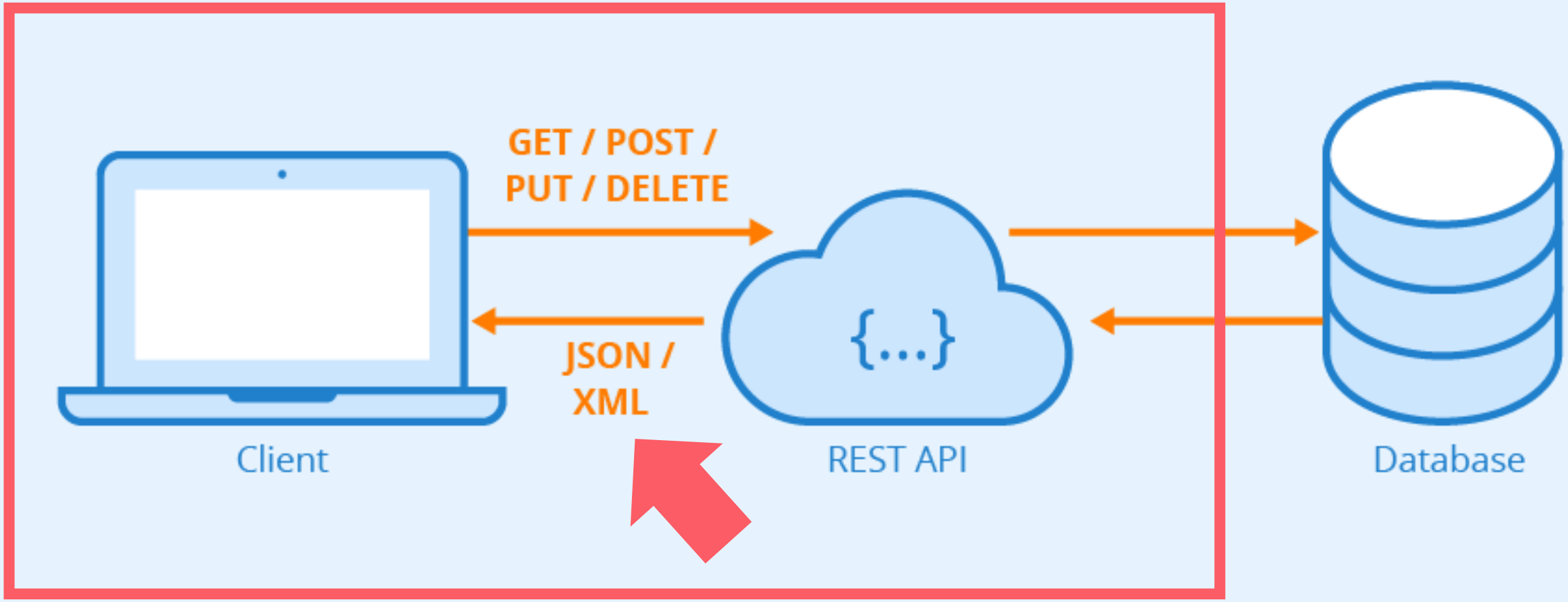
```
from flask import Flask
app = Flask(__name__)

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

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

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 -
```

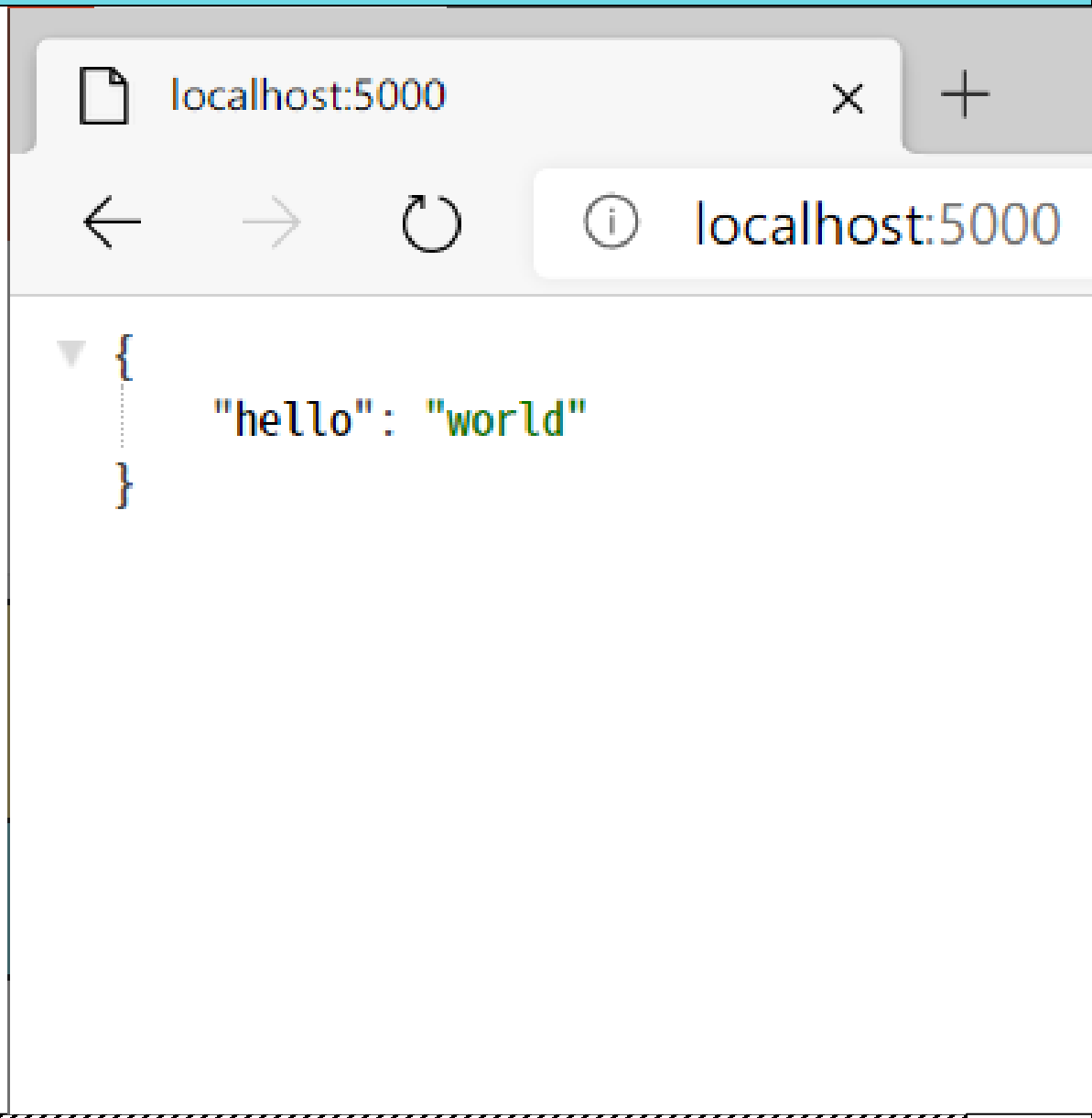


Flask 使用指南

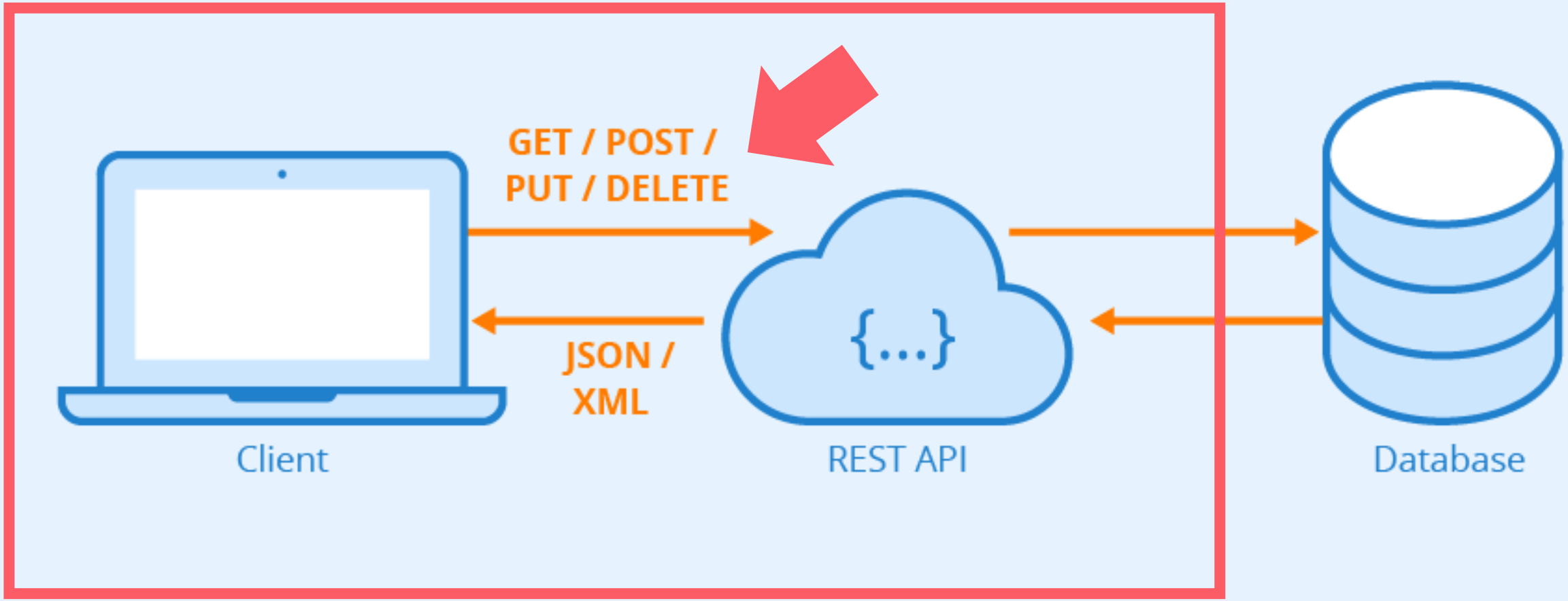


❖ 使用 JSON 与客户端通信

```
from flask import Flask, jsonify

...
@app.route('/')
def hello():
    return jsonify({
        'hello': 'world'
    })

...
```

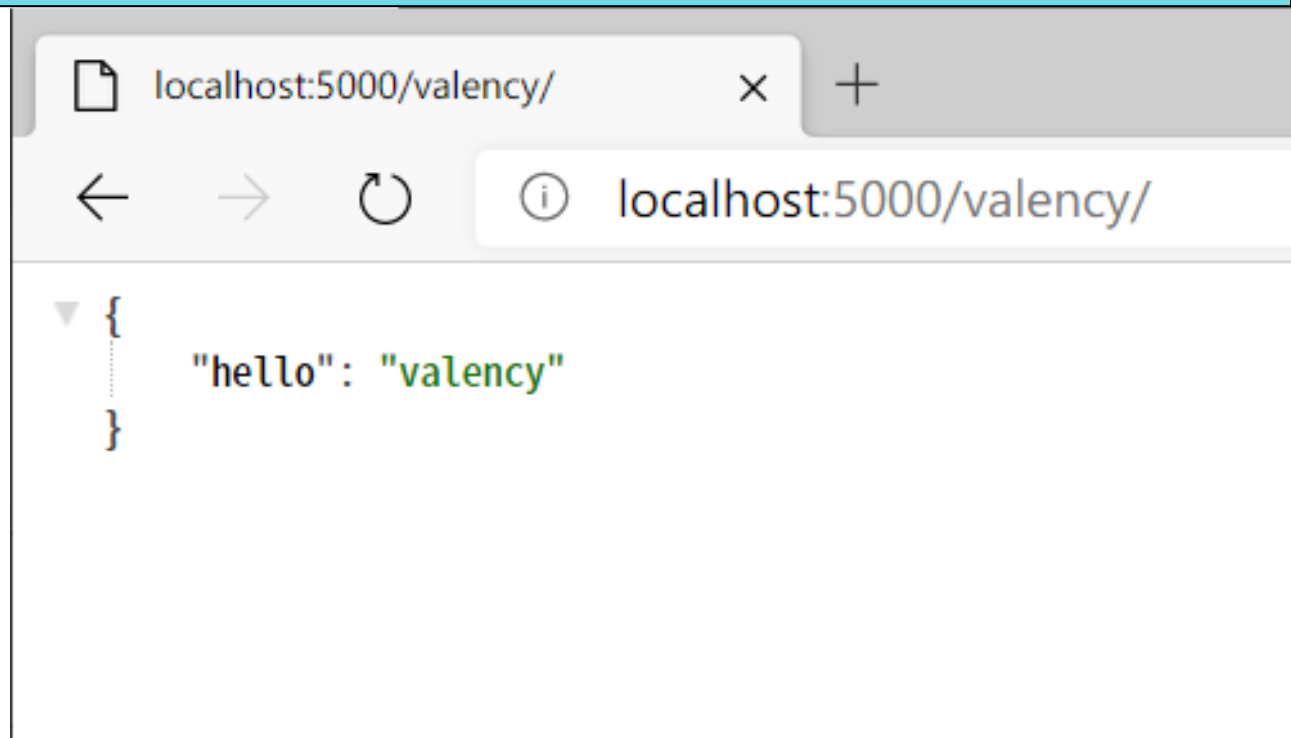


Flask 使用指南



❖ 基础路由（Routing）

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



Flask 使用指南

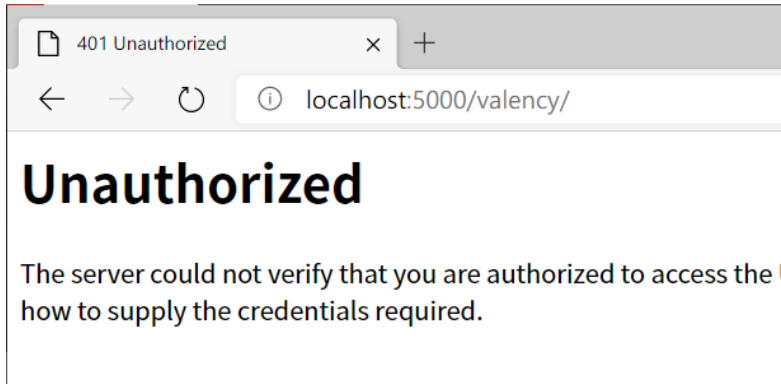
- HTTP 请求方法（HTTP Request Method）
- https://www.w3.org/Protocols/rfc2616/rfc2616.html
- HTTP 协议定义了多种请求方法，用于以不同方式操作指定的资源（资源描述符，Uniform Resource Identifier，URI）

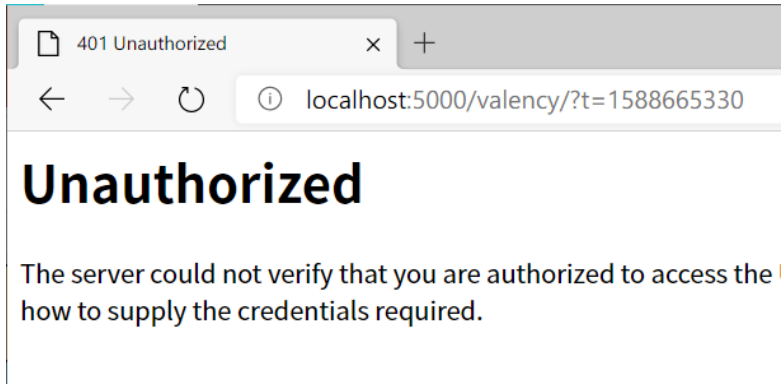
- 四种最常见的请求方法：
- GET：请求服务器“读取”指定的资源
- POST：向指定资源“提交”数据，请求服务器进行处理（例如提交表单或者上传文件）
- PUT：为指定资源“更新”其最新内容
- DELETE：请求服务器“删除”指定的资源

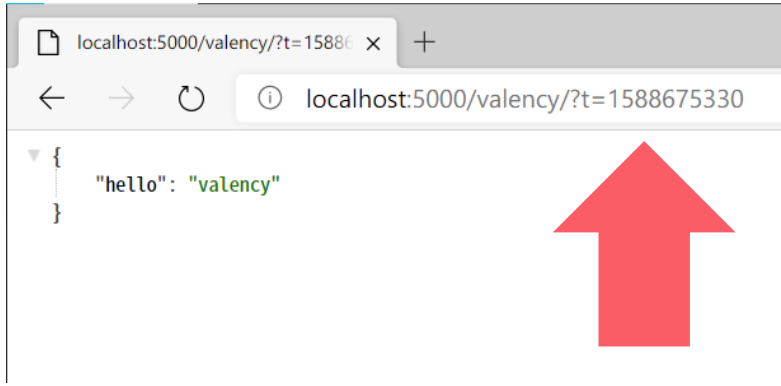
- HTTP 服务器至少应该实现 GET 和 HEAD 方法，其他方法都是可选的

❖ 处理客户端请求参数（GET）

```
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)
...
```







Flask 使用指南

- HTTP 状态码（HTTP Status Code）
- https://tools.ietf.org/html/rfc7231#section-6
- https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

- 用于返回 HTTP 的响应状态，常见的有：
- 1xx：服务器消息，通常不需要实现
- 2xx：成功，例如：200 OK、201 Created、202 Accepted、204 No Content
- 3xx：重定向，通常不需要实现
- 4xx：客户端错误，例如：400 Bad Request、401 Unauthorized、403 Forbidden、404 Not Found、406 Not Acceptable、409 Conflict
- 5xx：服务器错误，例如：500 Internal Server Error、501 Not Implemented

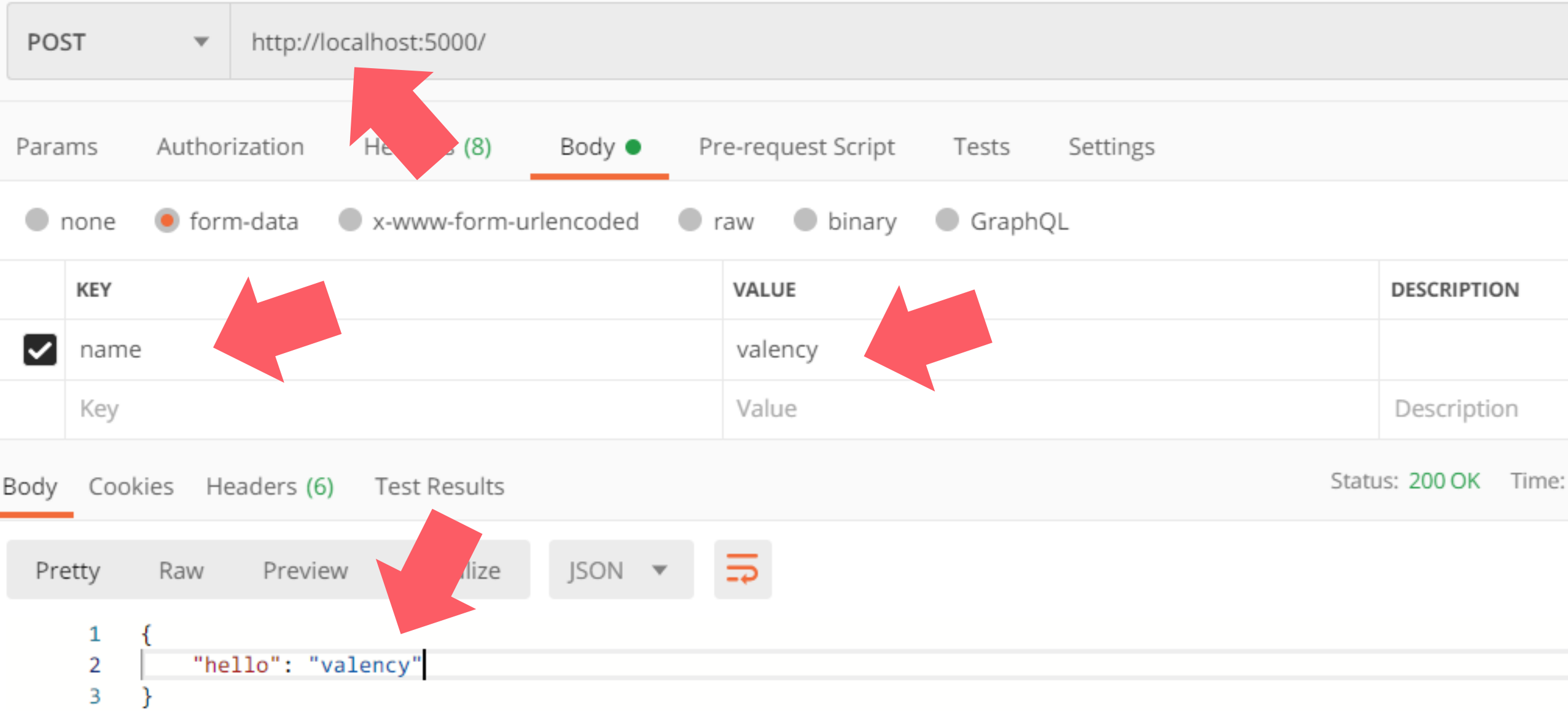
❖ 处理客户端请求参数（POST）

```
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)
...
```

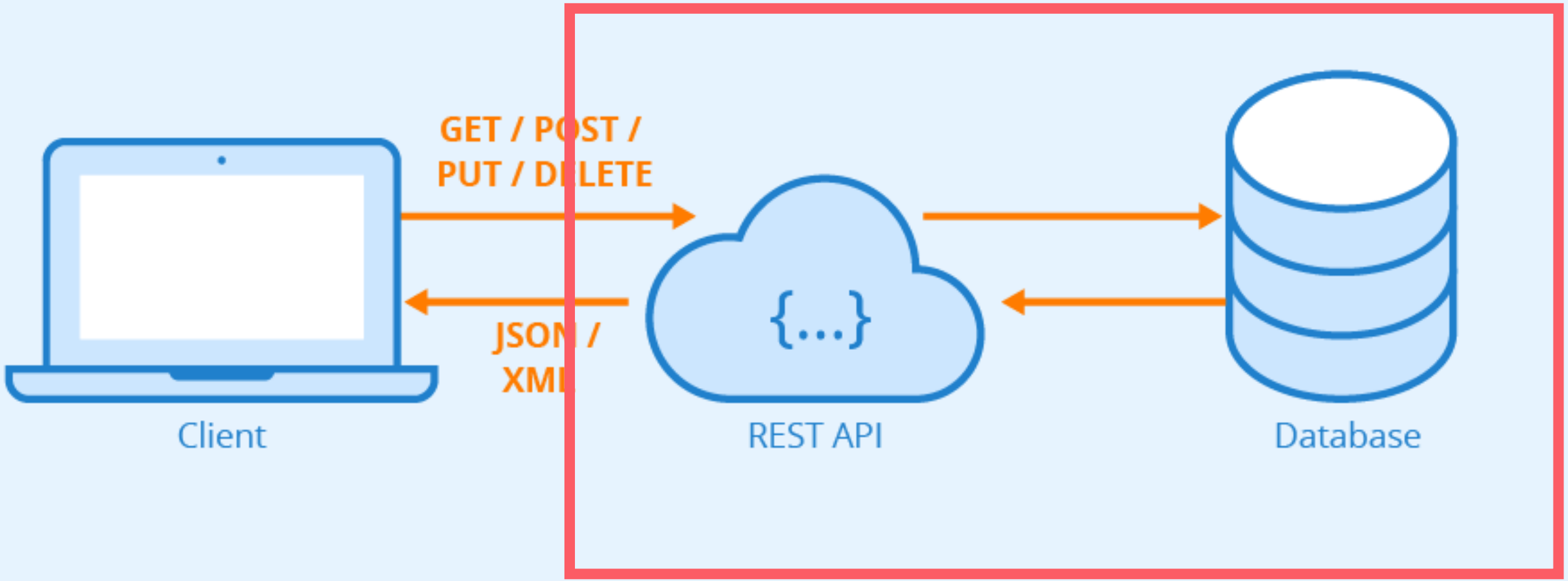
Flask 使用指南

- Postman
- https://www.postman.com/
- 非常出名的 API 测试 / 自动化测试软件

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

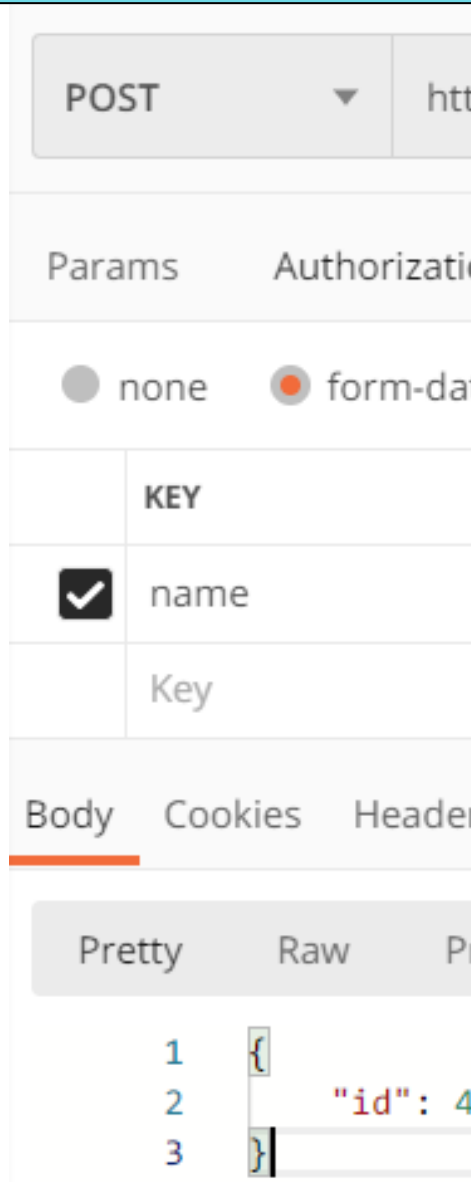


❖ 连接数据库

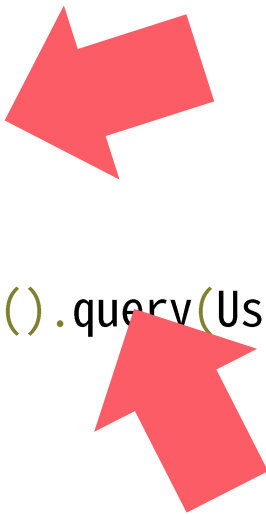
```
...
engine = create_engine('sqlite:///rest.db')
Base = declarative_base()
...
class User(Base):
    ...
...
if __name__ == "__main__":
    Base.metadata.create_all(engine)
    Session = sessionmaker(bind=engine)
    app.run()
...
```

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/en/1.1.x/

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

- Django
- https://www.djangoproject.com/

- 一个使用 Python 编写的全栈互联网应用框架
- Django 采用了 MVT 的软件设计模式，非常全面且细致

- 个人观点：
- Flask 比较适合学习和测试 API 框架，或是开发简单的互联网应用
- 从软件工程的角度而言，Django 更适合团队协作，且功能更完善

- Django vs. Flask

- https://medium.com/@SteelKiwiDev/flask-vs-django-how-to-understand-whether-you-need-a-hammer-or-a-toolbox-39b8b3a2e4a5

- https://hackr.io/blog/flask-vs-django

- https://www.zhihu.com/question/33538127

- https://baijiahao.baidu.com/s?id=1652854891500064989&wfr=spider&for=pc

目录

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}`))
```

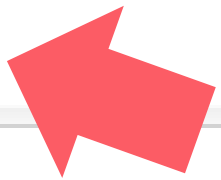
Express 使用指南

node

```
~/Workspace/course » node api.js                                valency@aorus-master
http://localhost:5000
```

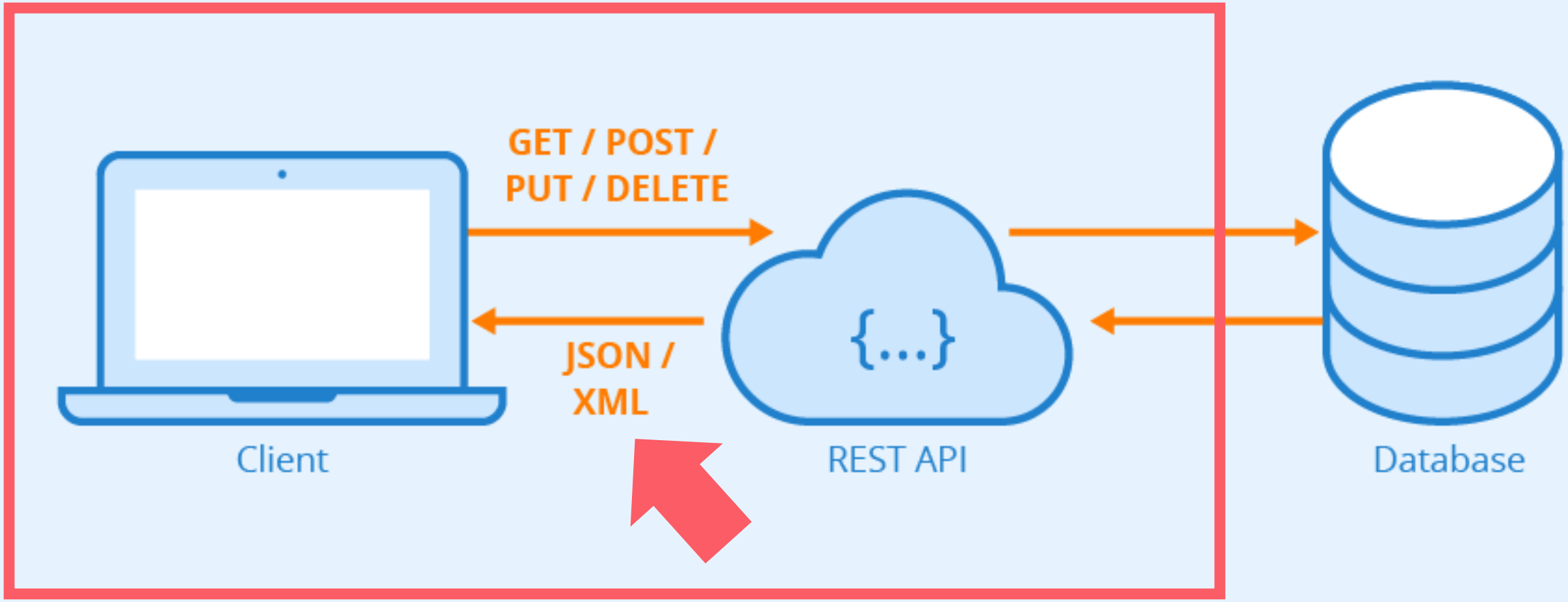
localhost:5000

localhost:5000



Hello World!

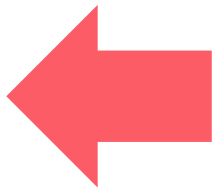
Express 使用指南

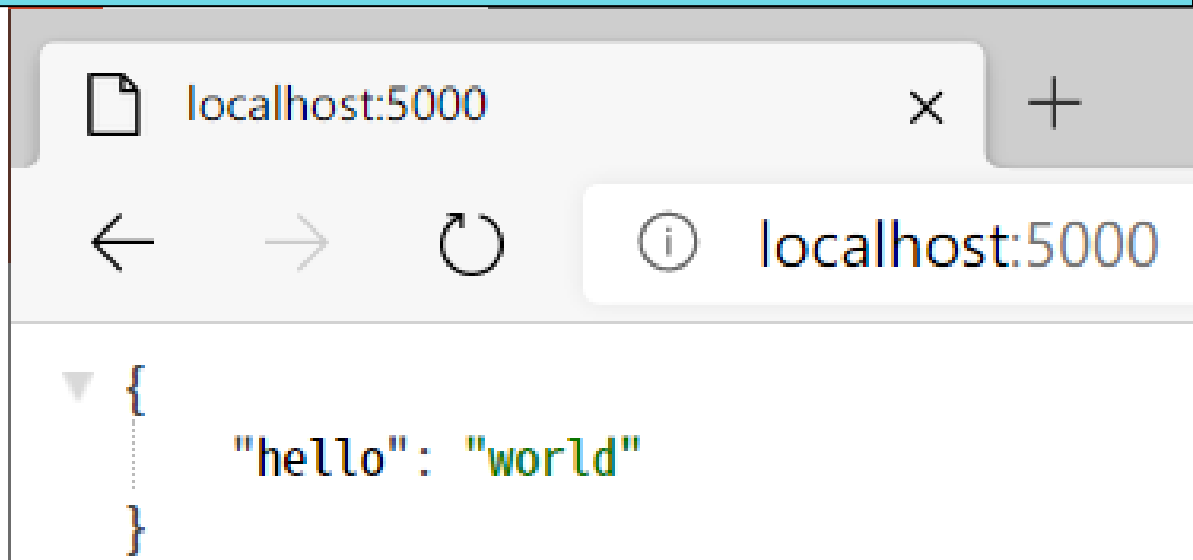


Express 使用指南

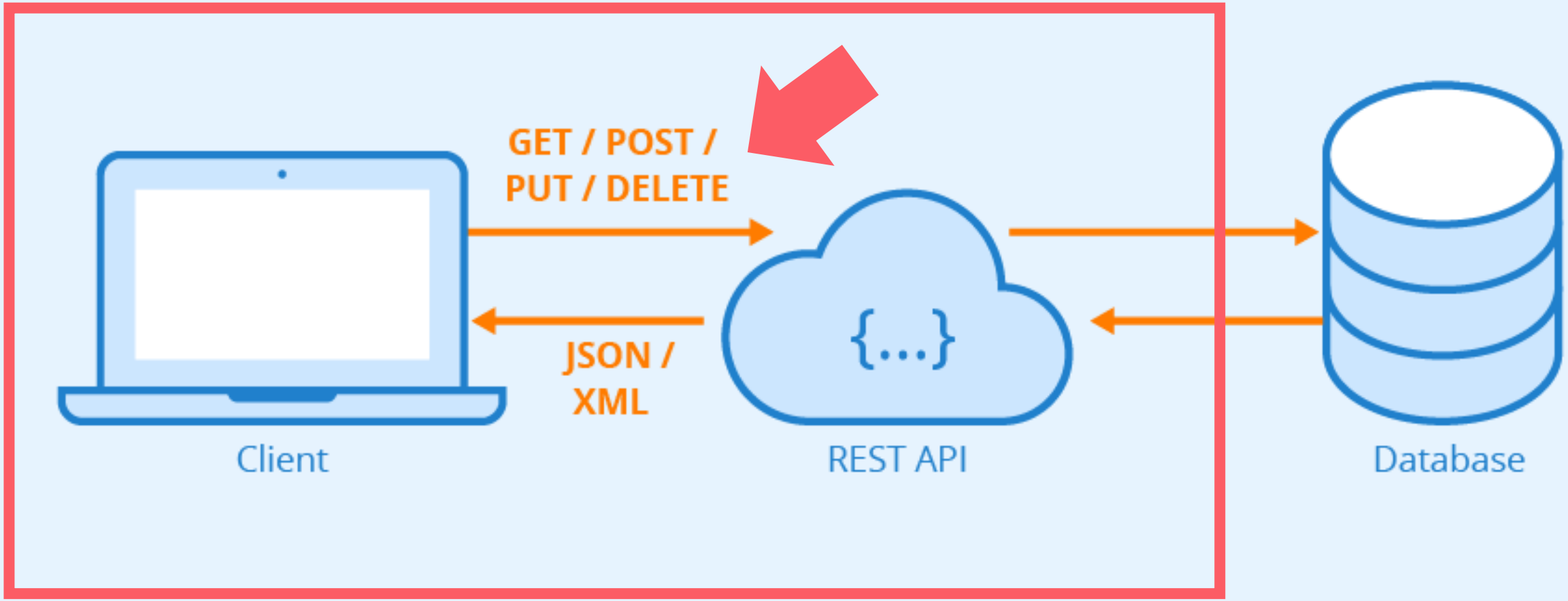
❖ 使用 JSON 与客户端通信

```
...

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

...
```





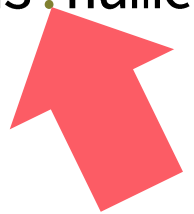
Express 使用指南

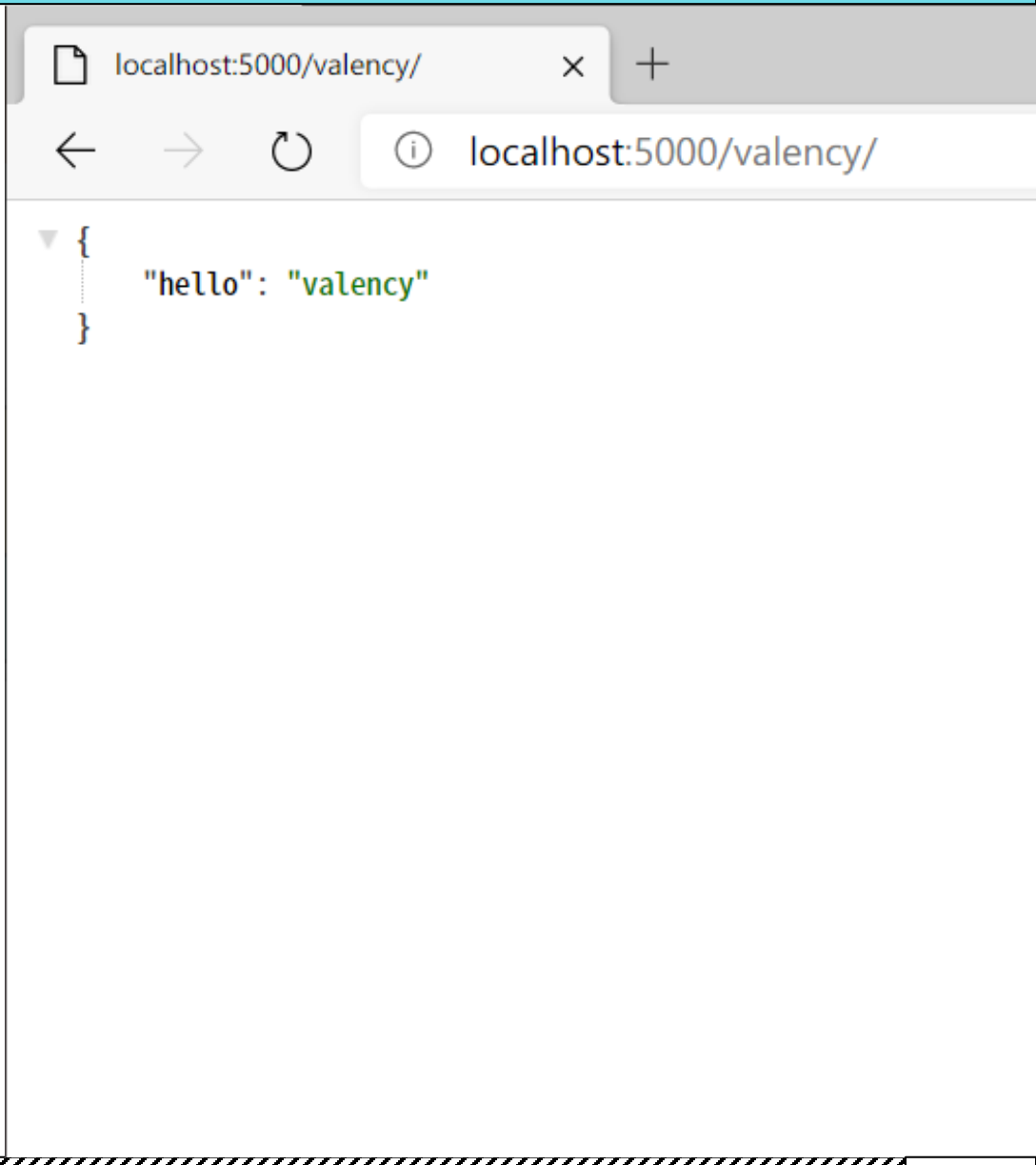


Express 使用指南

❖ 基础路由（Routing）

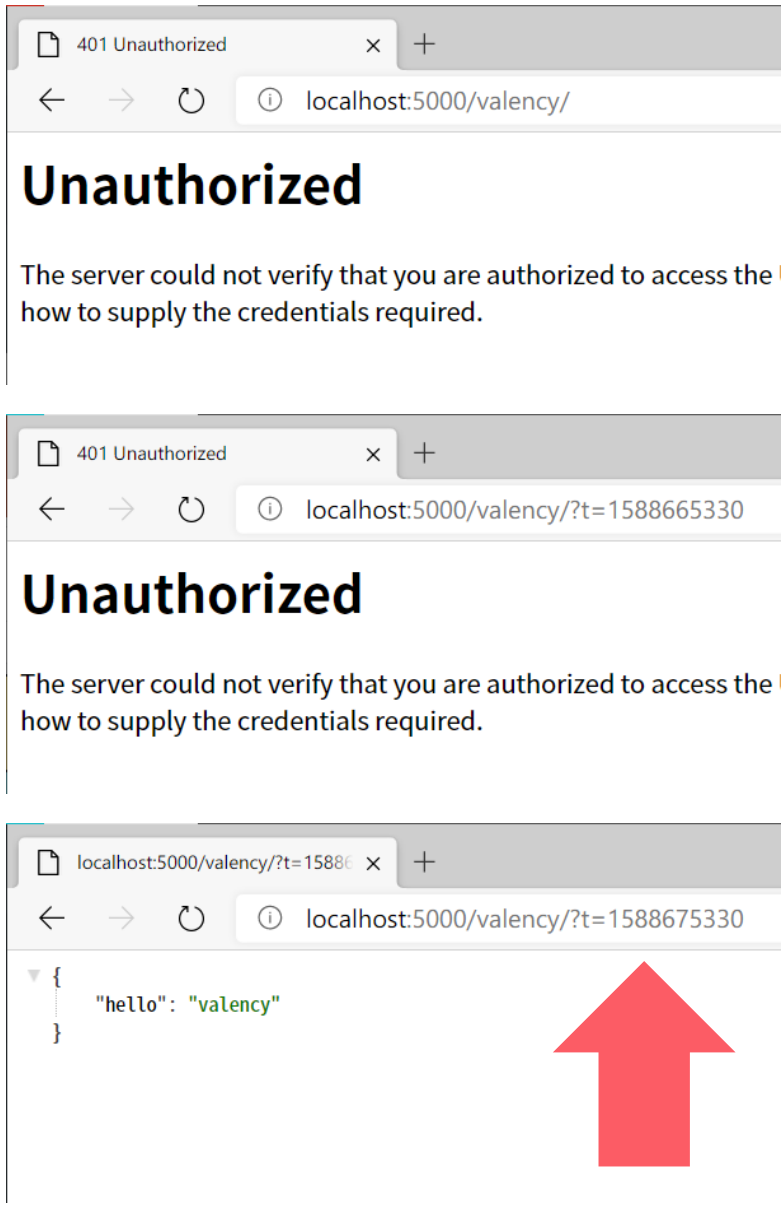
```
...
app.get('/:name/', (req, res) => res.send({
    'hello': req.params.name
}))
...
```





❖ 处理客户端请求参数（GET）

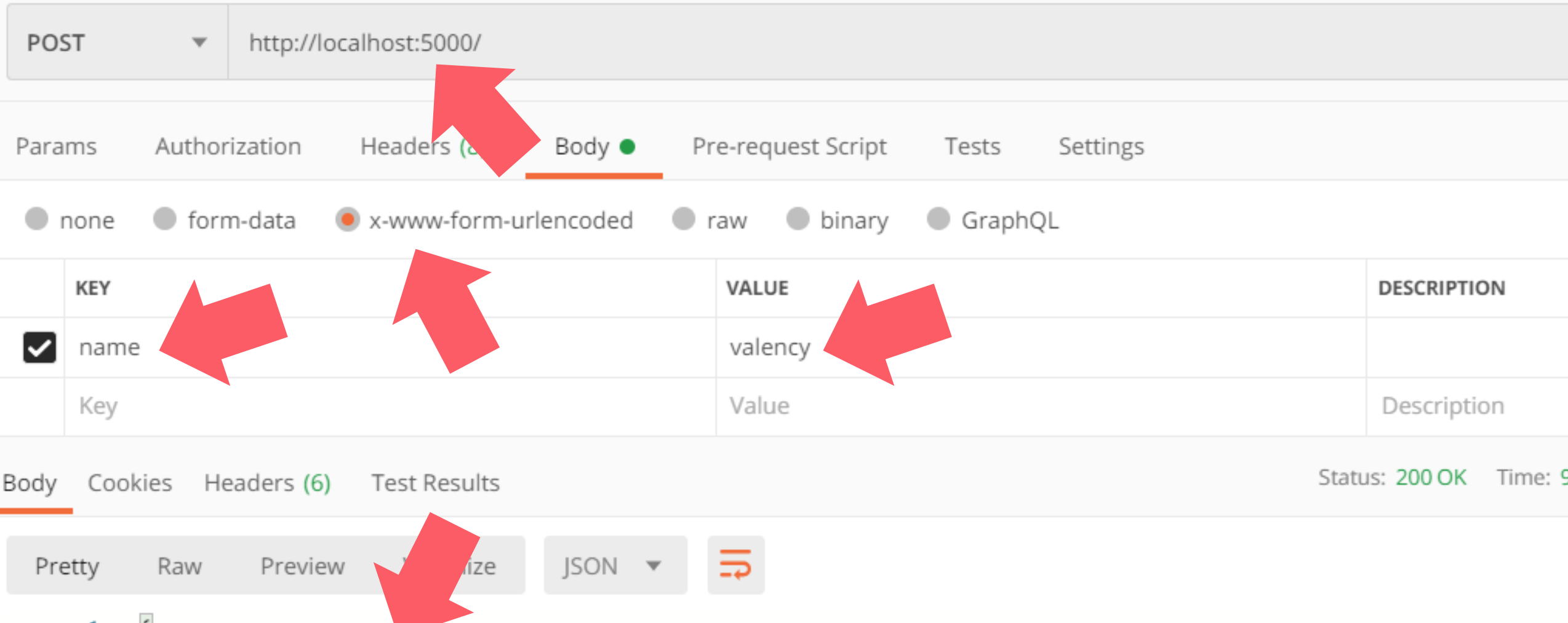
```
...
app.get('/:name/', (req, res) => {
    let name = req.params.name
    let t = req.query.t
    if (t && new Date().getTime() / 1000 - parseInt(t) < 3600) {
        res.send({
            'hello': name
        })
    } else {
        res.status(401).send()
    }
})
...
```



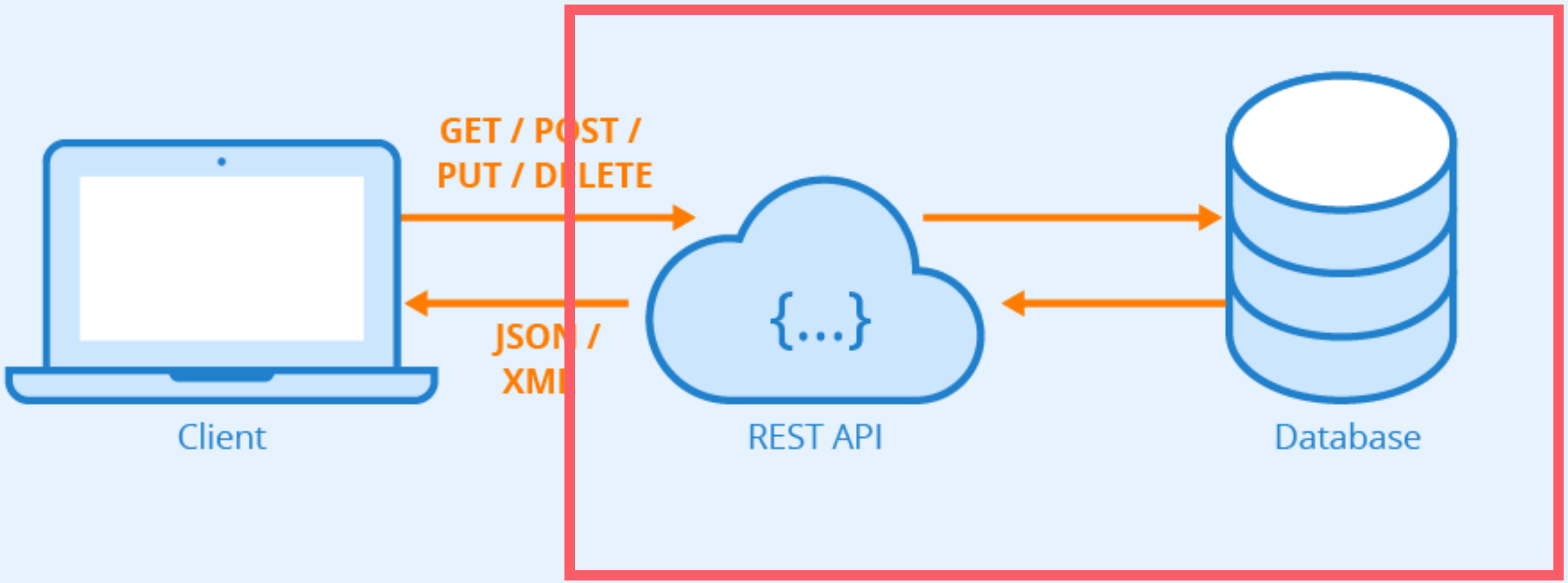
❖ 处理客户端请求参数（POST）

```
...
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

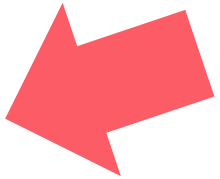
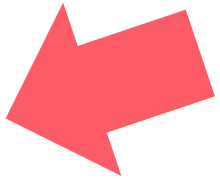


Express 使用指南



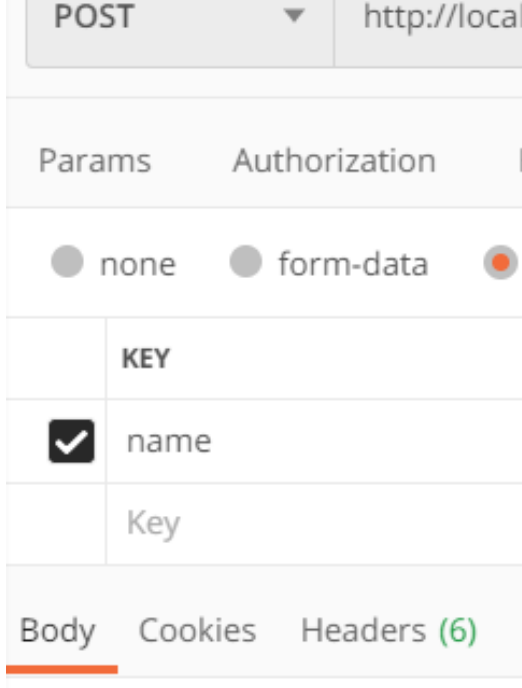
❖ 连接数据库

```
...
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'});

...
```

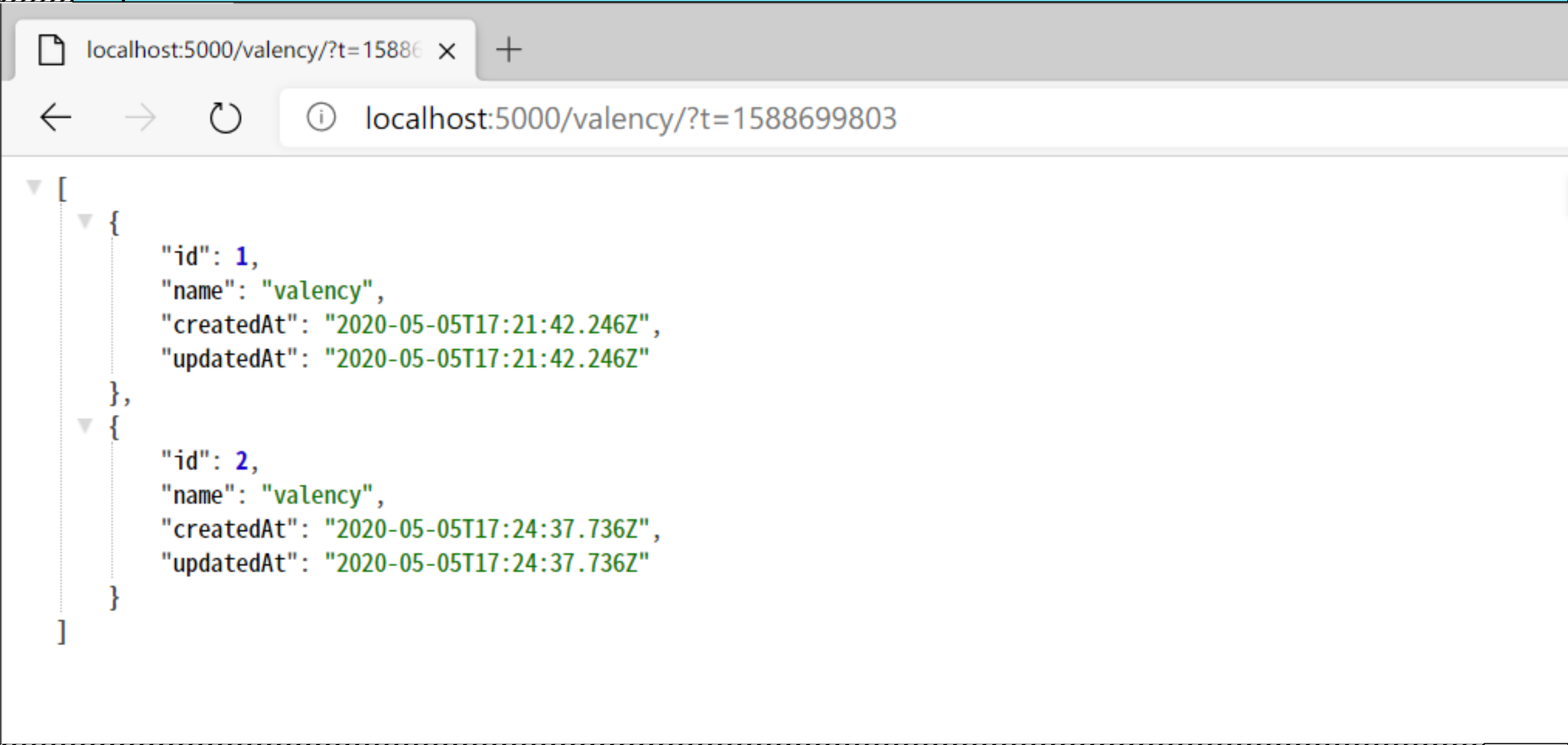

Express 使用指南

```
...
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

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

总结

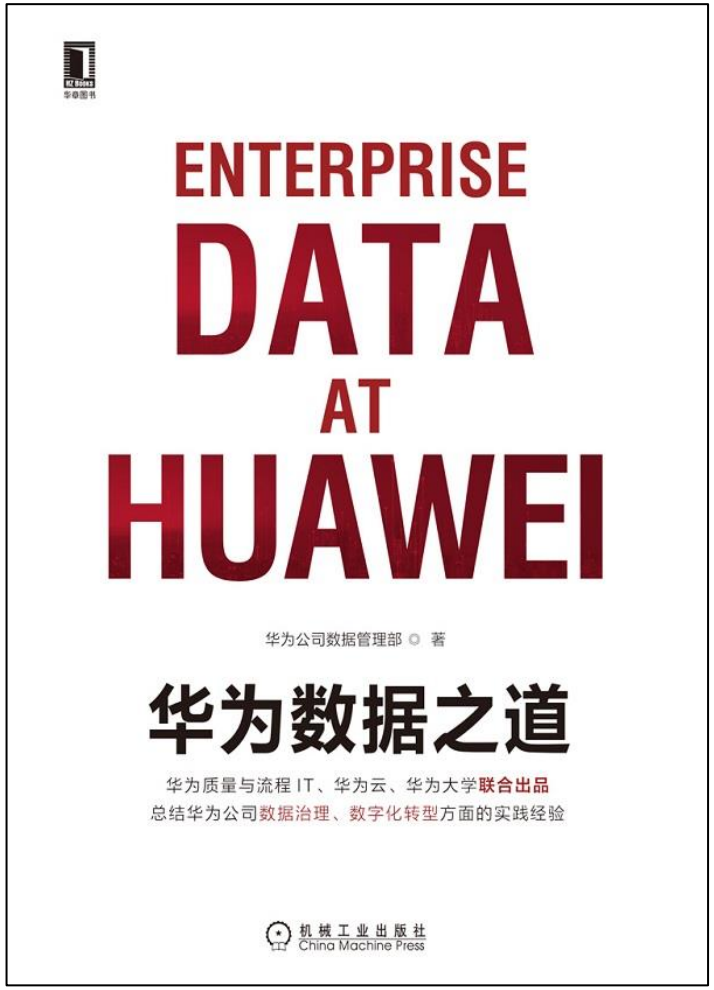
Google Cloud
cloud.google.com

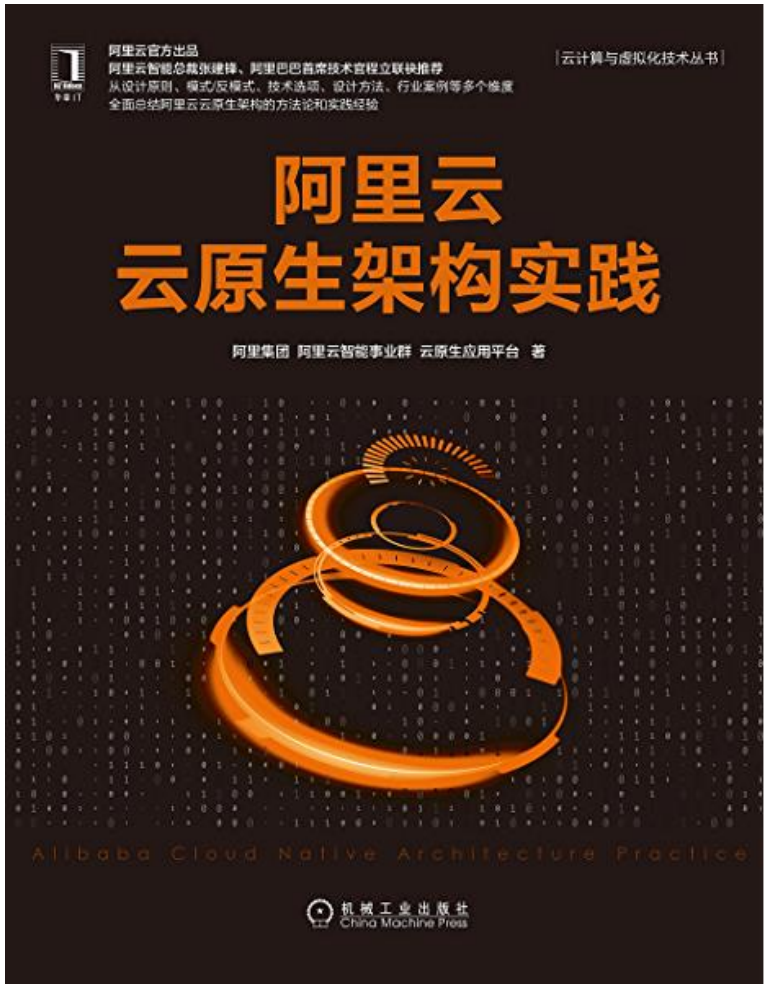
Amazon Web Services
aws.amazon.com

阿里云
aliyun.com

总结


大数据技术与应用专业规划教材
AWS-教育部-AWS产学合作专业综合改革项目规划教材
云计算与大数据
技术理论及应用
林伟伟 彭绍亮 编著
清华大学出版社


ENTERPRISE
DATA
AT
HUAWEI
华为公司数据管理部 著
华为数据之道
华为质量与流程IT、华为云、华为大学联合出品
总结华为公司数据治理、数字化转型方面的实践经验
机械工业出版社
China Machine Press


阿里云官方出品
阿里云智能总裁张建锋、阿里巴巴首席技术官程立联袂推荐
从设计原则、模式/反模式、技术选项、设计方法、行业案例等多个维度
全面总结阿里云云原生架构的方法论和实践经验
云计算与虚拟化技术丛书
阿里云
云原生架构实践
阿里集团 阿里云智能事业群 云原生应用平台 著
Alibaba Cloud Native Architecture Practice
机械工业出版社
China Machine Press

THANK YOU