北京大学
PEKING UNIVERSITY

Python为核心的数据库管理和网页构建

汇报人：潘汪阅
组员：陆尧 邹瀚琳 夏天
2025/06/11

目录

第一部分

基于Python的Web框架介绍

第二部分

Flask介绍

第三部分

SQLite和Mysql的比较

PART 1

基于Python的Web框架介绍

1. Web框架简介
2. 几种Web框架对比

什么是Web框架？

Web框架是一套用于构建Web应用的工具和库，它提供了一系列预定义的功能和结构，帮助开发者更高效地开发Web应用，它本身不直接处理底层的服务器通信和资源管理等任务，而是依赖于底层的 Web 服务器和 WSGI（Web Server Gateway Interface）容器来运行。

Web框架的核心组件：

URL路由：将客户端的HTTP请求映射到相应的处理函数或类。

中间件：在请求处理过程中插入自定义逻辑，如身份验证、日志记录等。

ORM（对象关系映射）：将数据库表映射为Python类，简化数据库操作。

表单处理：自动处理用户提交的表单数据，包括验证和渲染。

模板引擎：将动态数据与静态模板结合，生成HTML页面。

内置安全机制：防止常见的安全威胁，如XSS、CSRF、SQL注入等。

插件和扩展：支持第三方插件和扩展，丰富框架的功能。

为什么需要Web框架？

1. 提高开发效率

预定义功能　　代码复用

2. 简化开发过程

结构清晰　　减少错误

3. 增强应用安全性

内置安全特性　　安全更新

4. 支持扩展和维护

插件和扩展　　社区支持

5. 提高应用性能

优化的底层实现　　异步处理

比较项目	Django	Flask	Tornado	FastAPI
类型	全栈框架	微框架	异步框架	现代异步框架
特点	内置功能丰富，适合快速开发大型应用	**轻量级，高度可定制，灵活性高**	异步处理，适合实时应用	基于Python 3.6+类型提示，自动生成API文档
开发效率	高，内置功能多，减少重复工作	中，灵活性高，适合快速原型开发	中，异步处理提升性能	高，类型提示和自动文档生成
学习难度	较难，需要学习Django的规范和架构	**较容易，易于上手，文档丰富**	中，需要了解异步编程模型	中，需要了解异步编程模型
安全性	内置多种安全机制，如CSRF保护等	安全机制较少，需手动实现	内置安全机制，支持HTTPS等	内置安全机制，支持HTTPS等
性能	中，适合中等负载	中，适合轻量级应用	高，异步处理适合高并发场景	高，异步处理提升性能
适用场景	**大型复杂企业级应用、CMS、电子商务等**	**小型应用、微服务、API服务**	实时应用、聊天室、数据推送	高性能API服务、微服务

PART 2

Flask介绍

1 Flask简介 2 Flask常用命令

Flask简介

什么是 **Flask**？

Flask是一个用Python编写的**微框架**，实现了**轻量级的、核心精简的、高度可扩展的**Web应用基础。“微框架”意味着它不捆绑数据库、表单验证等工具，这与其他全栈框架不同。

Flask本身不是一个全功能的套件，可以按开发者需求**自由选择和集成**各种扩展。Flask依赖于Werkzeug和Jinja2来处理底层请求和页面渲染。

为什么要用 **Flask**？

一个最简的Flask应用可以只是一个单一的Python脚本文件。

Flask是非常小的，是轻量级的，其核心代码简洁，不强制捆绑数据库、表单等重量级组件。

Flask遵循WSGI（Web服务器网关接口）标准，并提供了路由、请求处理等Web开发的核心功能。

Flask使用Python编写的，并提供了简洁、直观且易于使用的API（如装饰器路由）。

Flask可在任何支持Python的平台运行，包括Windows、macOS和各种Linux发行版。

Flask简介

常用命令：

```
#导入Flask及相关模块
from flask import Flask, request, render_template

#创建Web应用实例
app = Flask(__name__)

# 定义路由 (处理GET请求)和获取数据
@app.route('/login', methods=['GET’])
def login():
name = request.args.get('name’)
pwd = request.args.get('pwd’)
return '用户名：' + name + '，密码：' + pwd

#定义路由 (处理POST请求)
@app.route('/login', methods=[‘POST’])
def login():
name = request.form.get('name’)
pwd = request.form.get('pwd’)
return '用户名：' + name + '，密码：' + pwd

#渲染HTML模板
return render_template('profile.html', user=user)

# 启动Web服务器
app.run(host='127.0.0.1', port=5000)
```

PART 3

SQLite和Mysql的比较

1 SQLite简介 2 SQLite和Mysql的比较

SQLite简介

什么是 SQLite？

SQLite是一个进程内的库，实现了**自给自足的、无服务器的、零配置的、事务性的** SQL 数据库引擎。零配置意味着用户不需要在系统中配置，这与其他数据库不同。

SQLite引擎不是一个独立的进程，可以按应用程序需求进行静态或动态连接。SQLite直接访问其存储文件。

为什么要用 SQLite？

一个完整的 SQLite 数据库是存储在一个单一的跨平台的磁盘文件。

SQLite 是非常小的，是轻量级的，完全配置时小于 400KiB，省略可选功能配置时小于250KiB。

SQLite 支持 SQL92（SQL2）标准的大多数查询语言的功能。

SQLite 使用 ANSI-C 编写的，并提供了简单和易于使用的 API。

SQLite 可在 UNIX（Linux, Mac OS-X, Android, iOS）和 Windows（Win32, WinCE, WinRT）中运行。

Python中的sqlite3

SQLite 是一个C语言库，它可以提供一种轻量级的基于磁盘的数据库，这种数据库不需要独立的服务器进程，也允许需要使用一种非标准的 SQL 查询语言来访问它。一些应用程序可以使用 SQLite 作为内部数据存储，sqlite3 模块由 Gerhard Häring 编写。 它提供了 PEP 249 所描述的符合 DB-API 2.0 规范的 SQL 接口，并要求使用 SQLite 3.15.2 或更新的版本

常用命令：

```
# 连接数据库
conn = SQLite3.connect('mydb.db'）
cursor = conn.cursor()

# 查询数据
cursor.execute(“SELECT ... FROM users WHERE ...”)

# 插入/删除数据
cursor.execute(“INSERT INTO ... VALUES .../DELETE FROM ... WHERE ...”)

#获取所有结果
cursor.fetchall( )

# 提交未保存的更改（事务）
conn.commit()

# 关闭数据库连接
conn.close()
```

```
# 创建/修改/删除表
cursor.execute(“CREATE/ALTER/DROP TABLE ...”)

# 查询数据库中表的个数及其名称
cursor.execute("SELECT name FROM SQLite_master WHERE type='table';")
table_name = cursor.fetchall()
print("Table Name:", table_name)
```

SQLite和Mysql的比较

比较项目	SQLite	Mysql
架构模型	无服务器、嵌入式	客户端-服务器
数据库存储	单一磁盘文件（.SQLite, .db）	文件系统上的多个文件/目录
可移植性	极高	高（但需安装服务器）
设置与配置	零配置	需要配置
网络访问	仅限本地访问	支持远程网络访问
并发性与锁机制	数据库级锁	表级锁或行级锁
可扩展性	低	高（适合高负载、大型数据集）
数据类型	动态类型	严格静态类型
用户管理与权限	无内置用户概念， 依赖文件系统权限	完善的内置用户和权限系统
适用场景	本地应用、移动应用、嵌入式 设备、小型网站、测试/原型、 数据分析工具、只读或低并发应用	Web应用、在线事务处理、 高并发读写、大型应用、 需要远程访问、多用户协作

SQLite和Mysql的比较（续）

比较项目	SQLite	Mysql
优势	轻量级、零管理、跨平台文件、简单易用、部署成本低	高性能、高并发、高可靠性、功能丰富、可扩展性强、安全控制完善
劣势	并发写入差、无网络访问、无内置用户权限、不适合大数据高负载	需要安装配置服务器、管理相对复杂、资源消耗更大
SQL 兼容性	良好 (支持大部分标准SQL)	优秀 (广泛支持标准SQL和扩展)
编程语言接口	广泛支持 (C/C++, Python, Java, .NET, PHP)	广泛支持 (几乎覆盖所有主流语言)
存储引擎	单一引擎 (自研)	可插拔引擎 (InnoDB, MyISAM, Memory)
事务支持	支持 ACID 事务	支持 ACID 事务 (主要使用 InnoDB)
数据量限制	理论上很大 (TB级), 但实践中建议较小 (GB级)	支持超大规模数据库 (TB/PB级)

北京大学
PEKING UNIVERSITY

谢谢您的聆听