Aptche

HTTP Server & Web Framework

答辩人：脱敏处理

题目：Project-6 HTTP 服务器与客户端的自定义实现

项目介绍

项目名称：Aptche

项目模块：

- Aptche HTTP Server: 基于 asyncio 实现的 WSGI 服务器
- Aptche Web Framework: 基于 Aptche HTTP Server 实现的 Web 框架
- Aptche TODO List: 基于 Aptche Web Framework 实现的 TODO List

开发思路 - Aptche HTTP Server

参考 WSGI 规范，实现一个 WSGI 服务

WSGI(Web Server Gateway Interface)，
服务器和 Web 应用程序或框架之间的
目的是为了提供一种简单的方式，使
通信。在本项目中，Aptche HTTP Serv
例对象（app），将请求信息传递给 a
数据发送给客户端。

Aptche HTTP Server 基于 asyncio 实现，
应构造、支持基于 SSL/TLS 的 HTTPS 加
HTTP Server 能够支持高并发的 HTTP

```
def get_response(self, environ):
        headers_set = []
        headers_sent = []


        def write(data):
            if not headers_sent:
                # 发送响应头
                status, response_headers = headers_set
                response = f"{environ.get('SERVER_PROTOCOL', 'HTTP/1.1')} {status}\r\n"
                for header in response_headers:
                    response += f"{header[0]}: {header[1]}\r\n"
                response += "\r\n"
                writer.write(response.encode("latin1"))
                headers_sent[:] = [status, response_headers]
            writer.write(data)


        def start_response(status, response_headers):
            headers_set[:] = [status, response_headers]
            return write


        result = self.app(environ, start_response)
        response_body = b"".join(result)
        writer = BytesIO()
        # 将响应头和响应体写入缓冲区
        write(response_body)
        return writer.getvalue()
```

基于 SSL/TLS 的 HTTPS 加密连接

使用 mkcert(https://github.com/FiloSottile/mkcert) 工具生成自签名证书，并在本机安装证书。

然后在 Aptche HTTP Server 中加载密钥和证书证书文件，通过 ssl 模块的 create_default_context 方法创建 SSL 上下文，传入 asyncio 的 start_server 方法，即可实现 HTTPS 加密连接。

```
if settings.get("HTTPS", "False"):
                if not self.certfile or not self.keyfile:
                    raise ValueError("SSL 证书文件和密钥文件必须提供")
                ssl_context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
                ssl_context.load_cert_chain(
                    certfile=self.certfile, keyfile=self.keyfile
                )
                https_server = await asyncio.start_server(
                    self.handle_client, self.host, self.https_port, ssl=ssl_context
                )
                self.servers.append(https_server)
```

开发思路 - Aptche Web Framework

```python
class Request:
    def __init__(self, environ):
        self.environ = environ
        self.method = environ["REQUEST_ME
        self.path = environ["PATH_INFO"]
        self.query_string = environ["QUER
        self.headers = self._parse_header
        self.body = environ["wsgi.input"]
        self.params = self._parse_query_s
        self.data = self._parse_form()


    def _parse_headers(self, environ):
        headers = {}
        for key, value in environ.items():
            if key.startswith("HTTP_"):
                headers[key[5:].replace("_", "-").title()] = value
        return headers


    def _parse_query_string(self):
        return parse_qs(self.query_string)


    def _parse_form(self):
        if self.method == "POST":
            return parse_qs(self.body)
        return {}
```

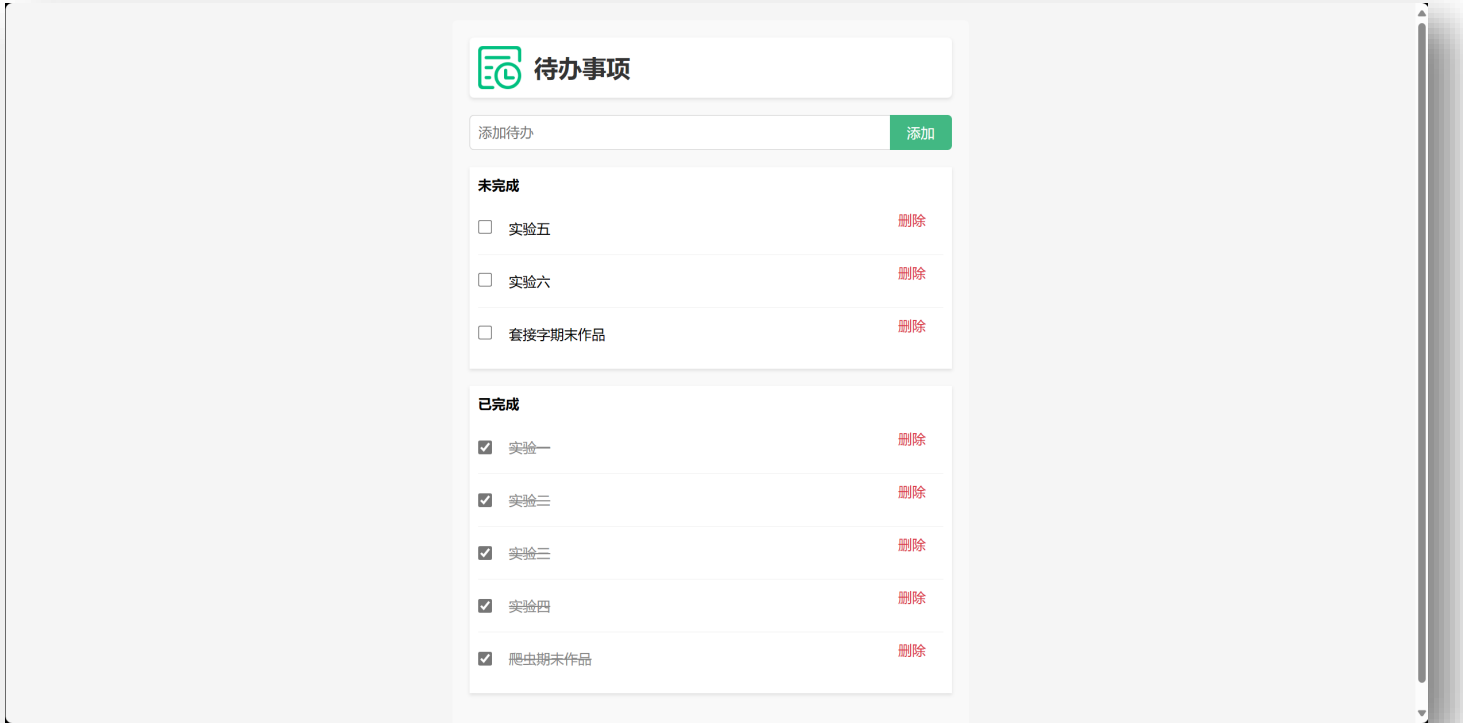
```python
def route(self, path, methods=None):
        if methods is None:
            methods = ["GET"]


        def decorator(func):
            self.router.add_route(path, func, methods)
            return func


        return decorator
```

```python
_by_content, get_gmt_date


s="200 OK", headers=None):


]


r):
ontent-Type", "text/html; charset=utf-8"))
    elif isinstance(self.body, bytes):
        self.headers.append(
            ("Content-Type", get_content_type_by_content(self.body))
        )
    self.headers.append(("Date", get_gmt_date()))


def __call__(self, environ, start_response):
    start_response(self.status, self.headers)
    if isinstance(self.body, str):
        return [self.body.encode("utf-8")]
    elif isinstance(self.body, bytes):
        return [self.body]
    return [self.body]
```

开发思路 - Aptche TODO List

再基于 Aptche Web 框架，实现一个简单的 TODO List。用于测试框架和服务器是否正常工作。

基本同 Flask 的语法，通过 Aptche Web Framework 提供的路由功能，实现了一个简单的 TODO List，支持添加、删除、完成/未完成等基本功能。



参考文献

1. Python软件基金会. PEP 3333 - Python Web Server Gateway Interface v1.0.1 [EB/OL]. (2010-09-26)[2024-12-03]. https://peps.python.org/pep-3333/.
2. 廖雪峰. WSGI接口 - Python教程 - 廖雪峰的官方网站[EB/OL]. [2024-12-03]. https://liaoxuefeng.com/books/python/web/wsgi/index.html.
3. Python软件基金会. asyncio — Asynchronous I/O[EB/OL]. (2019)[2024-12-03]. https://docs.python.org/3/library/asyncio.html.
4. Flask团队. Welcome to Flask — Flask Documentation (3.1.x)[EB/OL]. [2024-12-03]. https://flask.palletsprojects.com/en/stable/.

谢谢~