王志军, 姚文达. Web API 后端接口管理与应用[J]. 智能计算机与应用,2024,14(5):247-251. DOI:10.20169/j.issn.2095-2163.240535

Web API 后端接口管理与应用

王志军, 姚文达
(中冶南方工程技术有限公司技术研究院, 武汉 430223)

摘 要: 本文介绍了 ASP.NET Web API 的主要特点及使用方法。在此基础上提出了自承载、跨域资源共享、Swagger 文档管理、缓存、限流等增强和改进技术。接下来,又提出了针对大量 API 的接口管理需求及其技术实现方案。进一步地,提出这些技术的集成方案,并应用于真实工业环境中,现场运行稳定,接口查看和调用方便,为业主创造了巨大经济效益。

关键词: Web API; 接口管理; Swagger; 技术集成
中图分类号: TP315 **文献标志码**: A **文章编号**: 2095-2163(2024)05-0247-05

Web API back-end interface management and application

WANG Zhijun, YAO Wenda
(R& D Institute, WISDRI Engineering & Research Incorporation Limited, Wuhan 430223, China)

Abstract: This paper introduces the main features and usage of ASP.NET Web API. Based on the above, the paper proposes the enhancement and improvement techniques, such as self-hosting, Cross-Origin Resource Sharing, Swagger document management, cache, and rate limit. Afterwards, the paper presents the demand for interface management for a large number of APIs and its technical implementation scheme. Furtherly, the paper puts forward the integration scheme of these technologies that are applied in the real industrial environment. The results of application show that the site's operation is stable, the interface viewing and calling are convenient. Besides, the research application creates a huge economic benefit for the owners.

Key words: Web API; interface management; Swagger; technology integration

0 引 言

随着工业信息化水平的提高,现代企业的业务范围在不断扩大,企业的业务数据也呈现出大幅增长的趋势,并且包含不同的业务维度。由此,各个应用或服务之间的数据也变得较为耦合。不同业务数据的开发商不同,导致这些数据分布于不同应用服务器或数据库中。数据的高效共享和处理难度逐渐增大,因而形成了烟囱式系统和数据孤岛,浪费了大量的数据资源,也不利于数据的分析以及挖掘[1-2]。

Web API 是一个可以对接各种客户端的技术方案,实现各种数据资源的整合和对外服务的统一[3-6],从而极大地方便了数据的使用和管理。随着Web API 数量的不断增长,Web API 接口的监控、运行状态的跟踪和历史记录查看变得愈发重要。

1 ASP.NET Web API

1.1 Web API 主要特点

ASP.NET MVC 4 是构建网络应用程序的一种优秀的设计模式,主要概念包括模型(Model)、视图(View)、控制器(Controller)[5-9]。Web Service、WCF、和 Web API 是基于 MVC 的不同技术路径。其中,ASP.NET Web API 是最适合构建 RESTful 服务[10-13]的平台,可以轻松创建一个连接移动物联设备、浏览器等不同客户端的 http 服务框架[14]。Web API 的主要特点如下:

(1)是一个简单构建 http 服务的新框架,并因其是无状态的,因此比较轻量;

作者简介: 姚文达(1995-),男,硕士,工程师,主要研究方向:冶金全流程智能制造。
通讯作者: 王志军(1985-),男,硕士,高级工程师,主要研究方向:冶金全流程智能制造。Email:zhijunwang@163.com
收稿日期: 2023-04-18

(2)是.NET 平台中最适合构建 RESTful 服务的技术;

(3)可以使用 http 的全部特点,包括 URIs、request/response 头、缓存、等;

(4)支持 MVC 的特征,比如路由、控制器、action、filter 等;

(5)由于 Web API 是强类型,开发过程不用考虑服务器和客户端之间传输数据的序列化与反序列化问题;

(6)可以部署在 IIS 或者嵌入到普通的应用程序中。

1.2 使用方法

(1)创建 Web API 项目。在 Visual Studio 2022 中新建项目,搜索并选择 ASP.NET Web 应用程序,点击创建项目,选择 Web API。将会自动生成一套 Web 项目模板;

(2)添加 model。model 就是应用程序中的一个具体的数据对象,根据配置(Json/XML),Web API 框架会序列化为字符串,在前端可以通过反序列化得到具体的数据对象;

(3)添加控制器。右击 Controllers 文件夹,右键点击添加控制器,根据需求选择 Web API 标签下的任一模板。需要注意控制器的名称要以 Controller 结尾。新建的控制器会自动继承 ApiController 类;

(4)创建方法。在新建的控制器中,建立一个新的方法,包括如下 4 个部分:

① 路由。用于确定最终访问的 url;

② http 谓词。常用的包括 HttpGet、HttpPost,对应 GET 请求和 POST 请求;

③ 方法。返回值是任意类型,将会序列化为 JSON 或者 XML 返回给前端;

④ 输入参数。如果不加标签,将会从路由中读取。加上 Frombody 标签,将会从 body 中获取并转成对应数据格式。

2 Web API 增强及改进

2.1 自承载

Web API 项目的输出是程序集,不能直接作为独立进程运行,需要承载到相应的宿主中。通常情况下,Web API 项目最常见的承载方案是部署在 IIS 上。但是,IIS 的部署和设置需要对操作系统进行一定的配置,使用起来不太方便。

为了解决这个问题,本文研究采用基于自承载(Self-host)方案,即在独立进程中通过监听指定端口来实现 http 服务,并将 Web API 托管到 Windows 应用程序、控制台应用程序或者 windows 服务上。

这种方案具有以下优点:

(1)部署简单。只需拷贝即可使用;

(2)调试方便。支持断点调试,便于开发人员进行调试和排查问题。

2.2 跨域资源共享

出于安全考虑,浏览器只允许 JavaScript 或 Cookie 访问同站(协议、域名或 IP、端口完全相同)下的内容,对于脚本中发起的跨站请求进行限制。这样,不同项目(或者不同后台)之间的调用会默认被浏览器拦截。

为了解决跨域问题,使用跨域资源共享(Cross-Origin Resource Sharing,CORS)技术。CORS 的原理是在请求和响应报文中添加相应的标识,通知浏览器可以访问的范围。其中,访问控制来源(Access-Control-Allow-Origin)是重要的一个参数,通过这个参数指定可以接受的请求资源。这样浏览器就可以根据该参数得知该请求是否被允许,从而实现跨域请求的安全控制。

2.3 Swagger 文档管理

团队进行项目合作开发时,前端与后端之间的交互文档尤为重要。然后随着产品功能的迭代优化,后端服务的相关代码修改,导致相关接口规约或者功能的变化,却并不能体现到接口文档上。频繁人工修改接口文档,除了增加工作量以外,还容易出错,增加了前端开发难度。Swagger 是一个规范且完整的框架,提供描述、生产、消费和可视化 RESTful Web Service,用于解决这一系列问题。

其主要特点如下:

(1)代码改变,运行后文档自动改变;

(2)支持主流的多种编程语言;

(3)swagger-ui 呈现出来的是一份可交互式的网页,可以供前端进行接口测试调用;

(4)可以将文档规范导入相关工具(postman),随后可以创建自动化测试。

Swagger 主要包含了 3 个模块,具体如下:

(1)Swagger editor。是一个开源的编辑器,用于编写 yaml 或者 json 配置的 OpenAPI 文档;

(2)Swagger UI:用于呈现 OpenAPI 文档,在浏览器中打开可以查看接口定义,并测试相关接口;

(3)Swagger Codegen:是一个代码生成器,可以基于根据 Swagger 定义的 RESTful API 自动生成服

务端和客户端代码。

2.4　缓存

Web API 接口缓存是指在客户端和服务器之间缓存数据，从而避免重复请求和响应数据的过程。通过缓存，可以减少网络流量，加快响应速度，提高系统的可扩展性和可靠性。缓存的主要方法包括：前端 HTTP 缓存、CDN 缓存和后端缓存。

数据库缓存和应用程序缓存是 2 种常见的后端缓存技术。其中，数据库缓存是通过将数据缓存到内存中来提高读取速度的一种技术。当需要查询数据时，首先会从缓存中读取数据，如果缓存中没有数据，则会从数据库中读取数据。可以使用第三方缓存库，如 Redis、Memcached 等来实现数据库缓存技术，也可以使用内置缓存机制，如 ASP.NET 中的 Output Cache 等。应用程序缓存是通过将数据缓存到应用程序的内存中来减少对数据库的访问的一种技术。当需要查询数据时，首先会从应用程序的缓存中读取数据，如果缓存中没有数据，则会从数据库中读取数据。可以使用内置缓存机制，如 ASP.NET 中的 Cache 等来实现应用程序缓存技术，也可以使用第三方缓存库。这 2 种缓存技术都可以加快数据读取速度，减轻数据库的负担，从而提高系统的性能和可扩展性。

2.5　限流

Web API 接口限流是一种控制并发请求的技术，用于控制 API 的访问速率和请求的数量，防止过多的请求导致系统崩溃或无法响应。通过限流，可以保证系统的稳定性和可靠性，提高系统的可用性。进行限流的方法具体如下。

(1)令牌桶算法：是一种基于令牌的限流算法，通过一个固定容量的令牌桶，按照固定的速率往桶里添加令牌，每次请求从桶里获取一个令牌，如果桶里没有令牌则拒绝请求；

(2)漏桶算法：是一种基于漏桶的限流算法，通过一个固定容量的漏桶，按照固定的速率漏水，每次请求需要等待漏桶腾出足够的空间才能处理；

(3)计数器算法：是一种基于计数器的限流算法，通过统计单位时间内的请求次数，当请求次数达到一定阈值时拒绝请求；

(4)基于时间窗口的限流算法：是一种基于时间的限流算法，将请求按时间窗口进行划分，统计每个时间窗口内的请求次数，当请求次数达到一定阈值时拒绝请求。

3　Web API 接口管理

3.1　概述

Web API 在应用中的价值越来越得到广泛认可，企业需要对其进行科学的管理和优化。当 Web API 数量众多时，监控和历史记录查看等功能是极其重要的，同时也是比较复杂的工作，需要采用专业的技术解决方案。

Web API 的监控是指对 API 接口的性能、可用性、健康状态等进行实时、自动化的跟踪和管理，从而帮助企业实现 API 接口的优化和最大化运行效率。Web API 的监控可以针对多个方面进行，例如对接口的请求次数、响应时间等性能参数进行监控，对接口的访问来源、地理位置等进行统计和记录，并且对接口的健康状态进行监测和维护。通过这些监控和跟踪手段，企业可以及时发现和解决 API 接口的性能和安全问题，确保 API 接口的稳定运行和数据安全。

除了实时监控外，Web API 历史记录查看也是一个非常重要的功能。历史记录查看可以帮助企业了解 API 接口的历史使用情况、数据传输情况等，从而掌握 API 接口的工作效果。此外，历史记录查看还可以提供 API 接口的异常情况记录、异常处理的方式等，帮助企业更好地了解 API 接口的使用情况，为进一步的优化和管理工作提供合理指导。

3.2　实现方案

过滤器(Filter)是 MVC 的一个特性，其中方法过滤器(ActionFilterAttribute)是 MVC 专门处理 action 过滤的类。同时实现了 IActionFilter 接口，也实现 IResultFilter 接口。包括 4 个方法，分别在 Action 执行前、Action 执行后、Result 前、Result 后这 4 个时机执行。

在 Action 执行前，记录调用时刻，新建一个 Stopwatch，开始运行。并将调用时刻和 Stopwatch 存入 http 请求的属性中。

在 Action 执行后，获取请求和响应，并在请求的属性中获取运行时刻和 Stopwatch 统计的耗时。可获取的接口的关键信息见表 1，上述信息将存入数据库。

为了进一步提高程序运行效率，可以先将上述信息推送到消息服务器，随后再将新的线程存入数据库。

表 1 接口关键信息

Table 1 Interface key information

字段	中文名称	备注
Uid	唯一 ID	使用 snowflake 生产唯一升序 ID
RequestTime	请求时刻	
Ip	请求端 IP	
URI	URI 地址	
Method	HTTP 方法	GET、POST 等
MicroServiceName	微服务名	采用 Self-Host,使用计算机名和进程名替代
ThreadId	线程号	
ElapsedMilliseconds	耗时	
RequestBody	请求 Body	
ResponseContent	响应内容	
ActionName	方法名	
ControllerName	控制器名	

4 技术集成方案及应用

基于本文所述的增强及改进方案,选用的主要库和组件见表 2。

表 2 技术集成方案

Table 2 Technology integration scheme

编号	主要功能	核心库/组件
1	Web API	Microsoft.AspNet.WebApi.Core
2	自承载	Microsoft.AspNet.WebApi.SelfHost
3	跨域资源共享	Microsoft.AspNet.WebApi.Cors
4	接口文档管理	Swashbuckle.Core
5	缓存	Microsoft.Extensions.Caching.Memory
6	限流	AspNetCoreRateLimit

在钢铁行业的某管理系统中,应用上述技术管理后台接口近 100 个,后台服务采用多个进程分布式部署,并解决了浏览器跨域调用问题。接口输入输出信息可以在网页上随时查看,并进行测试查看结果。接口调用记录存入数据库,可以供相关人员进行接口耗时统计,调用情况分析,结果重现。针对关键接口(实时显示最新数据)进行了缓存,减轻了业务数据库的压力。对异常调用进行了限流,保证后台稳定运行。接口文档及测试工具示例如图 1 所示。

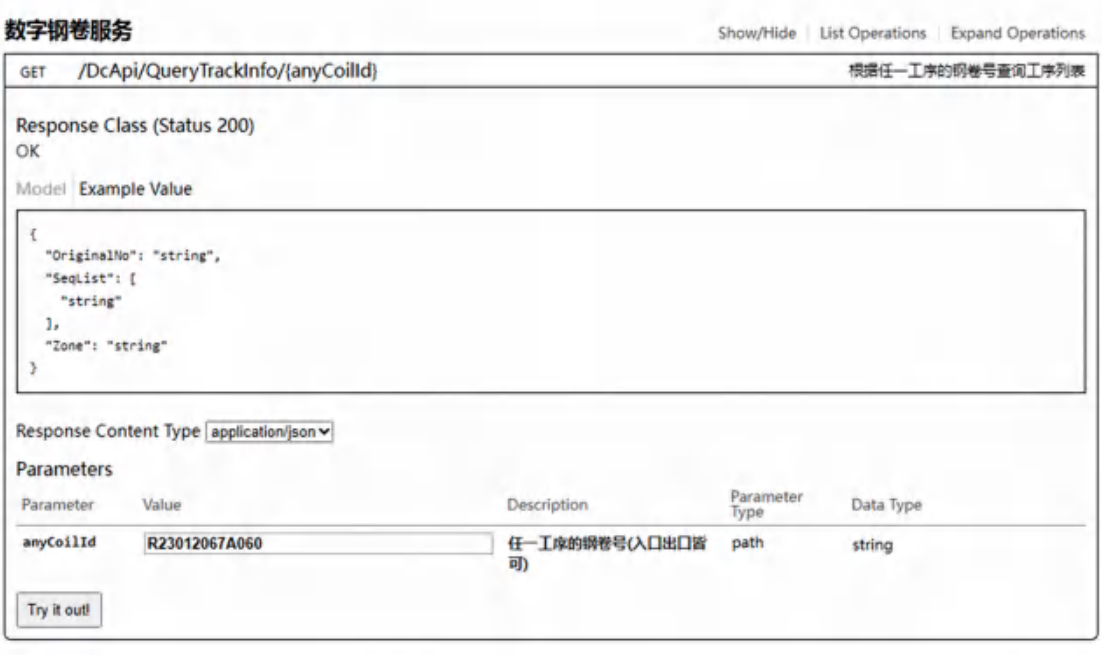

图 1 某管理系统接口文档及测试工具

Fig. 1 Interface document and test tool for management system

5 结束语

Web API 能轻松建立基于 http 协议的 RESTful 风格的服务,用于与前端或其它应用程序交互。本文在常规的 Web API 服务的基础上,实现了自承载,解决了跨域资源共享,并采用 Swagger 自动生成交互文档和测试工具,对海量接口使用过程存储和管理,针对关键接口进行缓存与限流,减少业务数据

库压力，保障后端服务稳定运行。在钢铁行业某大型管理系统中稳定运行一年多，运行经验表明本文方案易于开发维护，可扩展性强，后台占用资源可控，接口调用记录可供查询和分析优化，为企业带来可观的经济效益。

参考文献

[1] 彭烨，李晓如，韩勇华. 水务企业数据整合与共享关键技术研究及应用[J]. 城镇供水，2022（1）：66-68，60.

[2] 胡亨汶，孟祥印，李丹，等. 基于 RESTful Web Services 的云边数据交换设计与实现[J]. 现代制造工程，2022(8)：25-32.

[3] LIU Jun，ZHANG Zhinan. Web services-based knowledge sharing，reuse and integration in the design evaluation of mechanical systems[J]. Robotics and Computer-Integrated Manufacturing，2019，57：271-281.

[4] LYU S. Chatting in real-time with WebSocket [M]// /Practical Rust Web Projects. Berkeley，USA：Apress，2021：103-141.

[5] 吴江，戴雄奇，孙锋. 城市供水系统监管平台企业数据服务接口的设计与实现[J]. 给水排水，2020，56(8)：144-147.

[6] 卢瑜，翟明，赵华涛，等. 基于 ASP.NET Web API 平台的高炉自动化报表系统[J]. 现代冶金，2018，46(2)：43-47.

[7] 张亮，李正卫，蒋烨. 基于 AOP 实现冲突动态检测的实验室预约系统设计[J]. 计算机测量与控制，2020，28(4)：185-190.

[8] 祝杨军. 高校创业实验室的建设与探索 [J]. 实验技术与管理，2016，33 (11)：259-262.

[9] 欧阳宏基，宋笑雪，李红. 整合 ESMSH 框架的 JavaEE 应用架构[J]. 计算机测量与控制，2018，26 (10)：230-234.

[10] MALESHKOVA M，PEDRINACI C，DOMINGUE J. Investigating Web APIs on the world wide Web[C]// Proceedings of the 8th IEEE European Conference on Web Services. Ayia Napa，Cyprus：IEEE，2010：107-114.

[11] GODEFROID P，LEHMANN D，POLISHCHUK M. Differential regression testing for REST APIs[C]// Proceedings of the 29th ACM SIGSOFT Int′l Symposium on Software Testing and Analysis. ACM，2020：312-323.

[12] RENZEL D，SCHLEBUSCH P，KLAMMA R. Today’s top “RESTful” services and why they are not RESTful [C]// Proceedings of the 13th Int′l Conference on Web Information Systems Engineering. Paphos，Cyprus：Springer，2012：354-367.

[13] 周芯宇，陈伟，吴国全，等. REST API 设计分析及实证研究[J]. 软件学报，2022，33(9)：3271-3296.

[14] 栾鹏，刘协伟，王华春. 地调优选项目投标谈判预算信息采集系统构建[J]. 现代信息科技，2022，6(20)：22-25.