《Web API 的设计与开发》阅读计划

——图灵架构与技术群阅读计划（第1期）

领读人：雨帆

本书特色

- 对现行的 Web API 的规范和问题进行了较为详细的分析和总结，部分内容甚至过于琐碎
- 提供 API 的后端开发人员，对接别的系统的 Web API 的开发人员，都能从此书中找到共鸣
- 对于 HTTP 一类的协议，常见的 API 安全问题也有详细的讲解，这也是本书的精华所在

适合读者：微服务架构开发人员和开放平台开发人员

答疑时间安排：因为此书阅读时长较短，暂定每周一次，周六晚20:00—22:00

总阅读时长：共计2周左右

每天阅读用时：平均2小时

图灵社区本书网址：http://www.ituring.com.cn/book/1583

图灵阅读计划网址：https://github.com/BetterTuring/turingWeChatGroups

阅读规划

个人建议

整本书的阅读顺序，推荐第 1 章 -> 第 4 章 -> 第 2 章 -> 第 3 章 -> 第 5 章 -> 第 6 章

第 1 章 什么是 Web API——1h

说明：章节后面是阅读时间，1d 为2h，下同

本章节属于总览介绍章节，对于 Web API 下了定义，介绍了了 Web API的需求场景、使用方式和为什么要设计“优美”的 Web API。

阅读建议

本章节属于介绍性章节，无论以前是否接触过 Web API 的开发，都不推荐在这个章节耗费时间。对于初次接触 Web API 的同学，建议有一个大概印象即可。

重点篇章：1.4、1.5 稍微细看一下即可。

1.1 Web API 的重要性——10m

简单说，就是 Web API 的历史演变过程说明。无重点内容，更多是对 Web API 的几个经典案例的介绍。

1.2 各种各样的 API 模式——10m

此处是对 Web API 的使用场景的列举。因为微服务的兴起，对于大多数开发者而言，可能最后的内部系统集成才是最主要的应用场景。无重点内容。

1.3 应该通过 API 公开什么——10m

定义 Web API 的开放边界，有点传教式的部分，个人建议 Pass。

1.4 设计优美的 Web API 的重要性——10m

分四点阐述了设计良好的 Web API 的好处，需要结合实践来感受。

1.5 如何美化 Web API——10m

对于两个设计基本原则的阐述:

1. 尊重设计规范
2. 尊重事实标准 后面的四章基本都是围绕着这个在进行说明

1.6 REST 与 Web API——5m

这里只是理清 REST 与 Web API 的关系。简单说：REST 属于 Web API 的子集，本书讨论的内容不仅仅是 REST。

1.7 作为目标对象的开发人员数量与 API 的设计思想——5m

没啥东西，就是说 Web API 是给人用的，设计时要多加考虑。

第2章 端点的设计与请求的形式——3d 1h

重点篇章：2.2、2.3、2.4、2.5

端点(Endpoint），在HTTP 协议里面，它相当于请求地址URI 。在业务系统里面，相当于某个具体的业务功能。 好的请求地址，是结合业务系统：易读、易改、有层次的。

对于请求形式，相当于不同的 HTTP Method，不同的 Method 有大家约定俗成的操作语意，如：GET 获取，PATCH 更新之类的。

本章节基本就是围绕着上述两大块，以设计一个 SNS 移动后端 Web API 的例子，结合各种设计时需要注意的细节来说明一个好的 Web API 该长什么样。

阅读建议

对于熟练的 Web API 开发或者对接人员，这一章节会感觉很亲切、阅读很快。对于从来没接触过此类开发的人员，优先推荐先去了解一些流行项目的 Web API。
如：https://mesosphere.github.io/ marathon/docs/generated/api.html

2.1 设计通过 API 公开的功能——15m

本小节主要是带领读者按照业务需求列举需要的 Web API。 核心思想是：不要按照系统数据库表结构的 CRUD 来设计 API，而要按照业务场景来列举功能点设计 API。

2.2 API 端点的设计思想——45m

本小节阐述了 API 端点(URI）设计的原则：容易记忆，URI 包含的功能一目了然。

并列举了这一原则的6大具体特点:

1. 短小便于输入的 URI
2. 人可以读懂的 URI
3. 没有大小写混用的 URI
4. 修改方便的 URI
5. 不会暴露服务器端架构的 URI
6. 规则统一的 URI

阅读建议

本小节内容需要结合一些常见 Web API 去类比，实际设计中，基于各种需求取舍，不会100%遵循这些设计原则。所以重点在于体会为什么这么设计好。

2.3 HTTP 方法和端点——1h

这部分内容基本和 RFC2616 第九章(https://tools.ietf.org/html/ rfc2616#section-9 一致，本小节主要说明，对于同一URI，可以按照 HTTP Method 的方法不同来设计不同的业务操作。

HTTP Method 简述:

1. GET 获取信息
2. POST 只创建信息(不同于表单，不用于更新)
3. PUT 更新或者创建信息
4. DELETE 删除信息
5. PATCH 只更新信息

阅读提示

熟悉 REST API 的同学会发现，这就是常见的 REST API 定义。而实际上，这些均属于 HTTP 协议的方法定义。主流的 API 均依照这种请求方式设计，实现一个 URI 地址，多种操作。

个人吐槽:遇到过部分国企一类的单位，因为防火墙和审计需求，只允许用 GET POST 的，此类单位的开发人员需要慎重。

2.4 API 端点的设计——1h

阅读建议

本小节属于实战章节，结合前面两小节的内容，对于 2.1中提及 SNS 系统后端 Web API 进行设计。大家在阅读数，一定要对照前面的内容，加深印象和体会。

2.5 搜索与查询参数的设计——1h

本小节其实是对2.2章节的补充，介绍了常见的 API 请求参数。请求时附加上这些请求参数，即可在前面设计的 API 基础上，实现分页、搜索等特定查询。最后在 2.5.5 章节，简单提及了请求参数和路径的设计取舍。

阅读提示

本章节的参数，常用于 GET 请求的查询，设计 API 的时候，要有一定的冗余，能兼容较多的查询方式为佳。对于本小节中的数据量和相对位置的查询，主要和实际的数据持久层有关，不能生搬硬套。

2.6 登录与 OAuth 2.0——1h

背景说明

API 因为其开放式设计和架构，一般是无 Session(无状态) 的，所以对于 API 如何设计身份验证就至关重要。本小节列举了比较常见的 OAuth2 认证方式。

阅读提示

1. rfc6749 对于 OAuth2 定义的认证流程，其实有好几种，本书只是介绍了其中一种。
2. 对于 API 认证，常见的除了 OAuth2 之外，还有 Basic Auth、 Key Auth、JWT 等，可以自行了解。如参考 Kong 的文档。

2.7 机名和端点的共有部分——10m

这一节就是我在2.2小节领读部分里面提到的取舍，实际 API 设计时会在地址里面加上一些额外的公共部分，如版本号之类的来描述 API。

2.8 SSKDs 与 API 的设计——10m

这里主要是简单介绍了 API 在 SSKDs 场景的不足，第 5 章才会讲的服务编排。

阅读建议

无重点，了解即可。

名词解释

- LSUDs:Lerge Set of Unknown Developers
- SSKDs:Small Set of Known Developers

本书在 5.1.1 和 5.1.2 详解了两种 API 的异同。

相关文献(日文)

http://qiita.com/peka2/items/273be01065a921833878

2.9 HATEOAS 和 REST LEVEL3 API——17m

REST API 被 Martin 分成了几个级别，HATEOAS属于级别3的综合体现。

阅读建议

偏理论，没有实质性的实用内容，了解即可，不要多花时间。

背景说明

Martin 还是现在时常被人讨论的微服务架构的布道者，微服务本身，和 Web API 是一脉相承的。

2.10 第2章自我总结——1h 23m

梳理前面阅读的内容，结合常见的 Web API 体会如何设计

第 3 章 响应数据的设计——2d

第2章说的是 API 的地址和请求 Method 的设计，但是和 API 相关的还有响应报文，本章说的就是这一块的内容。

传统的 Web Service主要使用 SOAP、WSDL 等，本质上都属于 XML- RPC。但是现行的服务化兴起，微服务架构的流行，多语言的系统架构，使得大家开始倾向于使用更轻量的 JSON-RPC。所以本章讲响应数据的设计，倒不如说是 JSON 报文的设计。

阅读建议

3.2 的跨域部分如非必要，建议直接 PASS。

重点篇章：3.3、3.6。

3.1 数据格式——10m

Web API 常见的数据交互格式有 XML、JSON。各有千秋，JSON 胜在于简单，很多语言原生支持。XML 在于有 Scheme 一类的规范定义，能对报文校验，但因为过于复杂，虽然在一些

企业级框架项目内还能看到，但实际上不是很常见了。

本节除了介绍格式之外，还介绍了如何在支持多格式的 API 内定义用户所需格式。

阅读建议

无重点内容，了解即可。

3.2 使用 JSONP——20m

浏览器对于 XHTTPRequest 有同源策略，这里介绍 JSONP 目的在于实现跨域。不属于需要重点学习的内容，对于后端 Web API 开发者而言，只需了解即可。

3.3 数据内部结构的思考方法——1h 30m

本小节为第3章较为重点的内容。API 报文如果过于简单，会增加调用次数，增加调用难度。但如果过于复杂，在性能等指标上又达不到业务需求。所以，需要结合不同场景设计。

本节基于上述观点，列举5个需要在响应报文内注意的方面:

1. 用户定制响应内容
2. 是否需要包装统一响应结构
3. 数据是否需要分层级
4. 序列数据是否需要包装
5. 序列个数与后续数据查询

阅读提示

本节报文设计内容非硬式要求，在实际设计 API 的时候，请依照业务需求，选择性使用。

3.4 各个数据的格式——20m

本小节主要是对常见的数据字段的命名和值的选取。包含用户 ID、性别、日期等数据。

阅读提示

重点在于大整数的数据，因为精度问题，有时候会有误差、科学计数等问题，需要考虑使用字符串数字表示。

3.5 响应数据的设——5m

无实际内容，过于啰嗦。

3.6 出错信息的表示——50m

对于 API 调用出现异常时，如何比较恰当地通知调用方也属于 API 设计是否规范的标准之一。本小节从 HTTP Status Code、HTTP Response Head、HTTP Response Body 三个方面列举了异常通知的方式。

阅读提示

需要自行了解常见的 HTTP 状态码对应的语义，本书在 P107 有涉及，可以提前看一下。包含业务语义的统一错误信息包装，需要在设计和开发 Web API 时就予以考虑，避免返回意义不明的报错信息。

3.7 第3章总结——45m

复习梳理第3章内容，自我总结 API 报文该如何设计。

第 4 章 最大程度地利用 HTTP 协议规范——2d

Web API 本身就是基于 HTTP 请求设计的，所以广义上的 Web API，就是依靠 HTTP 协议的综合交互定义。

本章主要为 Web API 中请求头的设计，分别为:状态码、缓存头、媒体类型、跨域资源共享、API 私有头这几大块。

阅读提示

就个人经验，4.3的缓存在 Web API 的设计中使用的不多， 可以作为了解章节。4.2的状态码必须掌握，4.4媒体类型中常遇到的问题是附加报文编码问题。4.5 CROS 需要结合 CSRF 学习。

重点篇章：4.2、4.4、4.5

4.1 使用 HTTP 协议规范的意义——10m

本小节主要是对 HTTP 的一些标准背景介绍，实际与之相关的 RFC 文档建议在闲的时候阅读。

阅读建议

无重点内容，了解即可。

4.2 正确使用 HTTP 状态码——55m

本小节主要介绍了常见的4类状态码，2xx、3xx、4xx、5xx。

阅读提示

在 Web API 设计的时候，多以 HTTP 状态码来描述请求结果，尤其是在出现异常状态的时候。请结合前面的 3.6 节加深理解。

4.3 缓存与 HTTP 协议规范——55m

缓存相关的定义，在 Web API 开发时多使用 Gateway 进行包装，但目前无看到具体的使用范例，建议了解即可。

4.4 媒体类型的指定——35m

本小节介绍了媒体类型的格式，并列举了常见的媒体类型。

阅读提示

本章的内容编排上稍有混乱，实际上需要学习的东西为以下三点:

1. 有哪些媒体类型
2. Content-Type 来约定消息接收方该以什么格式解析报文
3. Accept 来约定消息发送方能发送哪些格式的报文

4.5 同源策略和跨域资源共享——35m

阅读提示

同源策略主要用于跨域或提供给浏览器的 Web API 设计时要考虑的东西。如果用于内系统集成的，则无此考虑。

补充资料

https://developer.mozilla.org/zh-CN/docs/Web/HTTP/ Access_control_CORS

4.6 定义私有的HTTP头部——20m

私有 HTTP 头部，主要的使用场景是为请求增加额外的信息。比如：OAuth2 认证后给请求附加上对应的授权用户 ID 信息。 用它的最关键原因是，对于不同的 HTTP Method，请求头是所有Method 都能修改的地方。

4.7 第4章总结——30m

复习梳理第4章内容，自我总结 HTTP Head 在 API 设计中需要注意的点。

第 5 章 开发方便更改设计的 Web API——2d

Web API 的变更是每个开发人员都会遇到的问题，有时可能是因为现有API 不能满足需求，有时可能是对旧的 API 的废弃。

阅读提示

本章主要说的内容为 API 版本化，以及基于版本化之后的变更与服务编排。第五小节涉及比较多的外文文献，需要自行阅读。

重点篇章：5.2、5.5

5.1 方便更改设计的重要性——15m

本小节列举了三种不同的 Web API 使用场景，和在这些场景下如果需要变更 API 会导致的问题，侧面说明 API 设计初期就需要考虑易于变更的重要性。

阅读建议

无重点内容，了解大概了解 API 有三大类即可。

5.2 通过版本信息来管理API——45m

为了便于 API 变更，需要给 API 增加版本信息。本小节主要讲了三种方式: 1. 地址中增加版本号 2. 请求参数中增加版本号 3. 媒体类型指定版本号同时还简单说了一下版本号如何定义，相信在参考目前各类开放平 台的 API 大家会有自己加版本号的想法。

5.3 版本变更的方针——10m

本小节只是简述了一些版本变更中如何尽可能向下兼容的方式，说白了，就是冗余字段。如非必要，尽可能不要有大版本变动。

5.4 终止提供 API——50m

无重点内容，更多是实践性教条。

5.5 编排层——1h 30m

本章简单说了一下 API 编排，但是关键内容都以脚注形式提供，本章的阅读重点在于对于脚注的4篇文献，需要仔细阅读。

5.6 第5章总结——30m

第 6 章 开发牢固的 Web API——2d 1h 39m

本章节旨在说明如何让 API 更加安全、稳定。

阅读提示

本章算本书中最有内容的一章，列举了比较多的 API 安全性 问题和如何规避 API 设计风险。第四节的避免非法请求，第六节的流控，都属于 API 设计中比较容易忽视但又十分重要的内容，需要仔细阅读。

重点篇章：6.3、6.4、6.6

6.1 让 Web API 变得安全——10m

本小节主要是列举 API 的三大问题，无任何重点内容。

6.2 非法获取服务器端和客户端之间的信息——25m

6.3 使用浏览器访问 API 时的问题——1h 25m

6.4 思考防范恶访问的对策——39m

6.5 同安全相关的 HTTP 首部——40m

6.6 应对大规模访问的对策——1d 20m

阅读时间安排甘特图

[《Web API 的设计与开发》阅读时间安排甘特图]

甘特图使用 Omni Plan 绘制，macOS 用户可以下载对应的源文件于本地阅读，体验更佳。

本甘特图绘制的时候，设定每天时间为2个小时，读书时间为20:00—22:00。时间在规划的时候，基本是保证2个小时能读完某个具体章节。因为预估有所不准，所以可能时间会存在一定的误差，请按照个人速度自行调整我定的时间。

每章在甘特图上都包含阅读建议、重点章节，每小节都尽可能总结支持核心内容。对于比较忙的读者，可以直接跳过部分章节，直接阅读重点内容。

本书的编排上稍显杂乱，所以甘特图上第1章的描述给出了读完第1章先读第4章的建议，可以选择性采纳。

其他建议

请在阅读时不要局限于书本内容，Web API 更是一种实践式的总结。看书时参照一些常见的 API，如：GitHub、Facebook 等，能加快你对于此书的理解。

读书时一定要按照“为什么要这么设计？这么设计有什么好处？”的自我提问方式去思考阅读。