大家下午好。那也非常感谢大家可以抽出时间来参加我们这次的api相关知识 的交流分享。

API 作为目前 相对主流的 程序之间进行沟通的桥梁，与我们的工作息息相关，也希望这次可以通过这次分享 能够让大家更好地认识API，从而助力我们的工作。

Page1 开始

那在今天的sharing中

考虑到team中的小伙伴在工作中也并不是都能接触到api，从而对api一些基础的认知和了解

那么

我会先分别以用户的角度 来向大家科普什么是api ，api如何使用 以及使用中需要注意的点

然后会以开发者的角度，来分享一下我对 如何设计一个良好的api的一些见解

接着会结合实际情况来谈一下 我们在使用常规方式去开发API是怎么样的流程

和用配置的方式去开发API的一个优缺点（这里计划进行提问）

当然 我也针对这种特殊的方式 开发了一个小demo，来向大家演示一下 如何去快速开发一个或一类 逻辑相差不大的api接口

Page 1结束

Page2开始

那我们回归正题，

什么是API。 API 是 Application programming interface 的缩写，意为应用程序编程接口

像微信，支付宝，都提供了各种各样的API 给大家使用,比如转账支付，查询天气啊 等等，

但我们只需要给对应的功能提供需要的资料或者参数即可。但至于 微信怎么帮我们进行转账或者它内部进行一系列很专业的操作，我们就不需要关心了，只要等着拿到处理好的结果就可以了。这个就是现实生活中的api ， 也就是说 通过API 接口，我们可以在约定好的规则下，方便快捷地获得某个内容提供方提供的服务。但是我们又不需要搞懂内部具体的工作机制，只需要关注输入和输出即可, 其实这也是我们常说的黑箱理论.

Page2结束

PAGE3 开始

那，在了解了API的基本概念之后，那我们为什么需要API呢.

关于这点 我也向小伙伴了解了下 我们目前的业务场景和相关的痛点

首先 据我了解，我们目前向下游提供数据的方式是会我们主动去推送，如果我们需要将同一类数据向多个源进行推送。那不管以什么形势来说，对我们来说，作为我们数据提供方，多多少少都会有一些effort来做推送这个工作。那如果推送的结果正常还好说，如果数据或者推送的结果中间出了问题，那我们也是需要对所有下游重新再全部推送一遍的。

那如果使用了API ，我们就不需要再主动推送数据。我们只要设置好API的规则，剩下的就交给下游自己调用就可以了。

第二点呢 ，我们通过 api 可以更容易地去控制我们拥有的数据资源。只要对API加入相应的鉴权和认证机制，就可以保证正确的人可以拿到正确的数据.

第三点，算是一个标准化， 因为现在大多服务之间的交互都是通过API来完成的，那如果我们也使用通用的方式来当做交互的桥梁，也会更便于后续我们与其他更多的服务进行对接

第四点 版本控制。在使用api之后，我们可以很容易的对 一个api或者一组api 组 进行版本控制 。不同的版本的api 去操作相同数据源，可以拿到不一样的数据.比较常见的就是在api的地址部分 通过 /v来区分版本

PAGE3结束

PAGE4 开始

那，我们该如何去设计一个api呢

首先

首先我们需要根据需求制定好对应api的API的规范。同时制作出相关的文档。一般来说，api提供方都会为消费者提供相应的API文档。

文档的内容包括 。像api接口地址、它的请求方式、请求Header、请求参数，返回值以及相应的说明。有的相对规范的文档中也会包含请求代码示例、返回值得示例代码等。

第二点

根据API的使用要求选择合适的认证方式，来确定调用者的身份是否拥有使用该API的资格。因为时间原因，我们这里就先不展开讲了。我挑了几种常见的认证方式，和大家简单介绍一下:

第一种 token-based 的认证方式

这种认证方式就像是一张“临时通行证”，只在短时间内有效。这样，即使别人拿到了你的“通行证”，也不能长时间地冒充你。

通常，在你首次登录,或者调用相关认证接口之后，服务方会生成一个“令牌”（Token）给你。在token超时时间之内，每次服务方提供的服务时，你请求头中带上这个token就可以了。

第二种是 OAuth 的认证方式

他本质上一种客户端与认证服务端直接的协议。具体的表现就是可以在不提供用户名和密码的前提下，实现完成从一个服务到另一个服务的授权。

其实这种方式我们日常中经常有见到， 比如我们可以用微信 去授权给京东 然后直接登录京东app而不需要在京东重新注册用户

最后一种是 叫JWT 的授权方式。

它和第一种token-based 授权方式类似。简单来说，在第一种的基础上额外在token中增加了相关的用户数据,并且在服务端不会保存密钥。

为了保证API的稳定性，我们也需要对API进行限流.一般来说我们会通过缓存来记录一个用户在一段时间内API的调用频次，并且对他调用次数做一定的限制.

以上这些都是 在设计一个API时 需要考虑的最基本的一些要素

Page4 结束

PAGE 5开始

除此之外呢。

对于API还有个重要的指标就是它的性能 。 评判API性能好坏 一般来说会参考3个因素

* 并发能力 指的是 一个ApI同一时间处理多个请求的能力。
* QPS：服务 每秒能够完成相应的查询次数。一般来说会和并发数一起来说，比如多少并发量下的QPS
* 每秒事务数（TPS）：服务 每秒能够某个事务次数. 这里的某个事务 中可以包含多个查询. 基本可以 TPS = 查询数xQPS

导致API 性能表现差的 因素有很多。

有时候 单纯是代码问题

那我们 需要对代码进行分析。

有时候，也可能因为业务逻辑的问题 存在多次递归复杂查询

需要增加缓存,减少直接查询的次数。

有时候，因为API涉及的业务链太长。那需要引入一些中间件，比如消息队列。

再比如，因为单表或者单库数据量过大。那我们需要做分库分表。

或者 确实并发量大 导致web容器，没有办法即时处理请求，  
那这时候需要横向对服务进行扩容。

那关于优化api的性能。其实归根结底，还是要具体问题具体分析的。找到瓶颈所在。才能得出一个有效的优化方法。

Page 5 结束

今天关于API 要分享的知识部分 就这么多了。接下来呢 我也开发了一个小demo。可以让不专精于API开发的小伙伴 可以通过一些的配置来快速开发一些简单的API。 当然也是时间原因 demo里目前支持的功能 也只是一个API的雏形，并没有相关的保护 验证之类的 机制.

当然完整版的功能在部门里已经有做了。大家也可以先借这个demo 大概了解下后续我们怎么通过配置来 开发API。

Demo section