41 前后端新增规约

更新时间：2020-10-22 18:09:29

> 构成我们学习最大障碍的是已知的东西，而不是未知的东西。 —— 贝尔纳

1. 前言

《手册》在不断迭代，现在已经迭代到嵩山版，嵩山版重点新增了前后端规约。

前后端规约主要讲前后端交互时的一些约定，遵循这些规约可以尽可能降低出错的可能性。

本文将重点对嵩山版的前后端规约部分进行解读，希望大家能够通过本文的学习，能够掌握规约学习的重点，掌握举一反三的能力。

2. 典型前后端规约解读

2.1 解读之前

虽然遵循《手册》中的规约很重要，可以帮助我们降低犯错的几率。但是规约的学习绝对不是记忆这么简单，其背后的依据更重要，才是我们学习的重点，如果不能够深刻理解很多规定的依据，只是停留在记忆层面，很容易遗忘，更难在未来的工作中做到举一反三灵活运用。

在正式解读前介绍几种方法和思想：

1、**“猜想和验证”**的方法，可以猜想这么设计的原因再和解释进行对比，理解才能更加深入。

2、“**以终为始**”的思想，多想一想规约的目的是啥，只有以终为始才能透过现象看本质，学到更多。

3、“**面向失败编程**”的思想，很多规约符合面向失败编程的思想，通过面向失败编程提高代码的健壮性。

4、大家在看到规约时一定要多一些思考，可以思考下为什么这么规定 / 设计？如果你来设计会怎么设计？如果不这么规定 / 设计会有啥坏处？

2.2 解读

解读 1：【强制】前后端数据列表相关接口返回，如果为空，则返回空数组 [] 或空集合 {}。

在说明中讲到此规约的目的是为了让数据层面上的协作更高效，减少前端很多琐碎的 null 判断。

那我们采用逆向思维，前端又能为后端数据层面的协作更高效，做点啥呢？

【补充】前端调用后端搜索接口时，如果用户没有输入任何搜索关键字，则不传或传 null 而不是空字符串。

通常对于后端而言，null 和空字符串是完全不同的概念，有些场景下空字符串还可能有着特殊的业务含义。

解读 2：【强制】服务端发生错误时，返回给前端的响应信息必须包含 HTTP 状态码，errorCode、errorMessage、用户提示信息四个部分。

浏览器、前端开发人员、错误排查人员、用户是请求的四个涉众对象，给出上述四点信息足以让涉众清楚请求的结果。

为了实现上述效果，应该添加全局错误处理器或采用其他方式在最外层将错误进行封装成上述格式。

其实联系实际项目中一些不优雅的设计就可以想到这么规定的原因。

很多项目不重视异常的分类和处理，在发生错误时后端抛异常前端直接将后端抛的 SQL 语法错误等具体信息展示给用户、给用户的提示信息过于技术化的现象，对用户非常不友好。

通常开发中会定义两种异常方式，一种是业务异常，一种是非业务异常。业务异常的错误信息通常会给前端直接展示给用户，用户可以清楚地知道发生了什么，自己该怎么处理；非业务异常（系统异常）通常会通过 “系统异常，请稍后重试”等描述反馈给用户。

解读 3：【强制】在后端交互的 JSON 格式数据中，所有的 key 必须为小写字母开始的 lowCamelCase 风格，符合英文表达习惯，且语义完整。

很多开发人员会根据自己的喜好遵守这样的那样的编程风格，但是很多人却从来没有思考为什么这么做。

我们要谨记“**以终为始**”的思想，前后端交互中 JSON 格式中的 key 是为了让前端理解数据的含义并准确展示的，换句话说是给人看的或者需要人中转，因此采用下换线、中划线、没有分割符、全部大写等方式显然更不可读，增加理解成本和出错的概率。

解读 4：【强制】errorMessage 是前后端错误追踪机制的体现，可以在前端输出到 type = “hidden” 文字类控件中，或者用户端日志中，帮助我们快速定位问题。

我个人不建议直接输出到 type = “hidden” 文字类控件中，因为查找起来有些麻烦，而且有些信息不应该暴露给用户（虽然普通用户不会直接看到）。我更倾向于建议放到用户端日志中，后端返回前也要进行日志记录。

此外日志中如果涉及到敏感信息如身份证号、商品价格等，建议进行脱敏处理。

其实这还不够，很多团队服务之间通常没有传递调用链路 ID （traceId），无法将前后端以及后端的不同服务之间的日志“串起来”，排查问题仅通过关键字，效率略低。

推荐引入日志链路追踪技术如 CAT，可以通过 traceId 追踪整个链路日志，提高排查问题的效率。

通常 traceid 由入口服务的名称和 uuid 组合构成，通过 traceId 可以看到事件来源，可以抓到上下游的日志，方便跨服务甚至跨中间件排查问题。

在使用链路追踪框架时要注意解决因使用线程池导致调用链丢失问题。

解析 5：HTTP 请求通过 URL 传递参数时，不能超过 2048 字节

手册中给出了详尽的解释，不同浏览器对 URL 的最大长度限制不同，超出长度的处理逻辑也不同， 2048 字节是取最小值的结果。

首先不推荐使用 URL 传递过长参数，此外除了长度问题外还要注意分隔符问题。

比如在实际工作中就遇到有人通过 URL 传递名称列表并通过逗号分隔，将名称传递到另外一个页面的设计，此时如果名称中包含逗号，那么格式将发生错乱，就需要对内容中的分隔符进行转义或者其他方式来实现。

【补充】调用二方或者三方批量接口时一定要注意列表数量限制。

除了 URL 请求的参数长度限制外，我们可能还会遇到调用二方或者三方接口批量接口（如批量发送公众号消息，批量查询店铺信息等接口），这些批量接口通常会有数量限制（如最多一次查询 50 个，最多一次发送给 50 个用户），一定要和下游确认清楚，即使没有限制，我们也不可能传入超大长度的集合去调用，避免测试时没有覆盖到，发布上线后出现 BUG 或故障。

不知道你是否想过，下游为什么会有这样的限制呢？或者换个角度，如果不限制会有什么坏处呢？

原因可能有很多，其中一个原因是如果批量查询过多，后端很容易超时，如果并发量超大下游也很容易 OOM。

解读 6：在翻页场景中，用户输入参数小于 1，则前端返回第一页参数给后端；后端发现用户输入的参数大于总页数，直接返回最后一页。

前面解读中讲到“**以终为始**”，我们要了解用户意图，当页码大于总页数时最有可能试图翻到最后一页，因此可以直接返回最后一页数据。

这里还要提到“**面向失败编程**”，尤其是后端要校验参数格式的合法性，参数业务的合法性，还要考虑下游接口异常情况等。

比如前端传入订单 ID，我们要检查订单是否存在，订单的状态是否符合要求等，再进行后续业务操作。

比如异步执行任务时，下游接口可能会因请求超时而偶发报错，可以通过重试来解决。

尤其在和外部系统对接时，特别要考虑各种失败的情况，评估是否需要做降级限流，是否需要做幂等处理等。

3. 总结

本文选取《手册》嵩山版新增的几条典型的规约进行解读，并结合自己的工作经验对前后端规约进行部分补充，希望能帮助大家更好地理解新增规约，希望大家不要去记忆规约，而是通过多一些思考深入理解规约，掌握规约背后的思想，未来的工作中更好地举一反三。

}

40 错误码相关新增规约的启示

42 专栏总结