程序接口设计、错误处理与应用日志

2014.4.20 yanlong

摘要

- 让我们的程序模块更靠谱
  - 程序接口设计
    - 糟糕接口的特征
    - 函数（结构化、函数式）、类（OOP）、服务（http）
  - 错误处理
    - 何时、何地、如何
    - 返回值、异常、callback/handler
  - 应用日志
    - 日志的作用和分类
      - 错误日志（warning、fatal）
      - 业务日志（trace）

程序接口设计

- 接口是什么？
  - 程序世界中各个组件协作的通道
  - “通道”设计的好坏影响着协作效率
    - 协助效率影响项目的进度、迭代
- 有哪些类型接口？
  - function、class、module、service、UI、protocol、configuration
- 想想那些你用过的糟糕接口
  - 名字丑陋、参数繁多、调用复杂
  - 文档缺乏、示例没有
  - 不稳定
  - 500、false、null、undefined、Error、Exception，WITHOUT msg

程序接口设计

- 如何调用一个接口（函数）？
  - 函数名
    - 起个好名字是个难题
  - 参数名称、类型
    - 是否可选、注意数量、注释类型（动态类型语言）
  - 返回值、类型
    - 注释、示例
  - 错误信息（返回值、异常类型）
    - 无法处理就上报
    - 未知状态比错误更可怕
  - 注释：用途说明、前置条件、示例

面向过程接口

- 做一件事（任务）并做好
  - 避免重复，无论是代码还是数据
  - 合理的粒度（参数个数、代码行数）
  - 层次化与正交化
- 异常优于返回值
  - 核心理念：分离错误处理逻辑
  - 本质一样：错误的反馈与检测
  - 更丰富的错误信息：type、code、msg、stack trace、exception chain
  - 代码示例

面向对象接口

- 对象与类
  - 数据的聚合与函数的绑定
  - 状态的转移
- 设计原则
  - SOLID：单一职责、开放封闭、里氏替换、接口隔离、依赖倒置
  - 识别并封装变化
    - 添加间接性：abstract function、interface
    - 设计模式
      - 不要滥用
  - 不要过度设计
  - 特别注意构造函数
  - 注意线程安全

服务接口

- 跨越了网络，但是本质一样
- xml/json-rpc：面向过程接口网络版
- RESTful：面向对象接口网络版
  - 全名：资源表征状态转移
    - 资源表征：通过URI指向资源
      - GET /users/${userId}/apps/${appId}/configuration
    - 表征状态
      - 自描述的信息（JSON/XML），Accept/MIME
    - 状态转移
      - CRUD：PUT、DELETE、GET、POST
    - 使用http status code表示错误
      - 代码示例
  - 资源与资源的集合
    - mongodb、backbone.js

错误处理

- 承上启下，错误的检测与反馈
  - 顶层函数（main/entry）、中间函数（model/control）、底层函数(util)
- 检测与处理
  - 前置条件（参数、状态）
  - 子任务（函数）调用结果
    - 关键任务
      - 打印错误信息，中断（return/throw），返回错误信息
    - 可选任务
      - 打印错误信息但不中断
    - 永远不要捕获但是不处理错误
      - 不rethrow就打印日志，墨菲定律
    - 适当的包装错误信息，rethrow
      - 错误信息与函数语义相符

错误处理

- 异步错误处理
  - try/catch失效
    - Scope改变，try block
  - 异步函数
    - 使用callback反馈错误
      - return，避免callback重复调用等未知状态
    - 尽量不抛异常
    - 代码示例
  - Node domain
    - 错误逻辑分离
    - 异步try/catch
      - domain块
    - 看起来略复杂

应用日志

- 日志的作用
  - 优秀的程序通过日志能够获知程序运行的全部情况
  - 做了什么？
    - Access log、framework log
  - 做的怎么样？
    - notice、warning、fatal
  - 怎么样在做？
    - trace: 跟踪程序逻辑流程
- 报警+日志+统计
  - 一切尽在掌握的感觉

应用日志

- 理想的日志形态
  - 已树状结构展现各个重要步骤的执行状况
    - 日志示例
    - 代价比较大，需要传递日志上下文给函数
    - Rig示例
- 折中的解决方法
  - 规则一：在检测并处理错误的地方打印错误日志
  - 规则二：在函数正确返回时打印trace日志
  - 日志示例

- Thanks!