Web 服务基础与构建

Web Service Basics and Building

中山大学
数据科学与计算学院

潘茂林
panml@mail.sysu.edu.cn

大纲

- REST概念与特征
- 基于HTTP的RESTful服务约定
- Local 键值数据库 boltDB
- Go REST 客户端与服务器的设计与编程
  - Github v3 研究
- Web 服务 API 及其设计与实现
  - RPC style
  - REST style
  - GraphQL
- RPC vs REST vs GraphQL

REST概念

- REST = Representational State Transfer
  - RESTful 是一种用于分布式超媒体系统的软件架构模式。
  - 它对分布在www上资源的访问与处理给定一组约束，以构建高可用、高弹性的云应用
- 基本概念
  - 资源（resources）：部署在互联网服务器中的任意的超媒体、实体或对象。如：文档、图片、用户、订单等
    - 资源不一定部署在一台服务器上，而是分布式的
  - 关系（relationships）：实体之间的关联，通常用链接表示两个资源之间的导航关系。下个实体、包含的实体等
  - 领域模型（domain model）：从客户端角度，为了解决特定领域问题而观察到的资源与资源之间关系构成的网络

Roy Fielding 博士论文，2000，https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

REST架构的约束与特征

- 无状态（Stateless）
- 可缓存（Cache）
- 统一接口
  - 用于大粒度的超媒体数据传输
  - 四个接口：
    - 资源识别：URL & URI
    - 资源表示：Media Types
    - 自描述消息：xml, json
    - HATEOAS（Hypermedia as the engine of application state）：用超媒体链接组织资源，如 html
- 层次化系统

无状态服务

- 访问与处理资源的服务必须是无状态的

  $s_{i+1}=f(s_i,r_i)$

  f 是一个服务
  s是该服务使用的网络资源的状态
  r是服务请求
  - 即对于一次资源访问请求，资源状态变化必须唯一，即服务不能内部持有状态
  - 服务处理资源是原子性的。服务必须仔细处理资源访问冲突
- 无状态服务的优势
  - 可视性（visibility）
  - 可高兴（reliability）
  - 弹性/伸缩性（scalability）

基于HTTP的RESTful服务约定

- 注意：是约定，不是标准
- 资源命名约定

```
curl -i https://api.github.com/users/octocat/orgs
curl -i https://api.github.com/repos/golang/go
```

- 主资源
  - 第一个资源，/实体复数/:id
- 子资源
  - 主资源后面的资源
  - Github api 与 网上一般性建议相比

```
GET /repos/golang/go
GET /users/golang/repos/go
```

  - 你喜欢哪个？

约定规则建议参考 Github API v3：https://developer.github.com/v3/ 不太负责任的中文翻译

基于HTTP的RESTful服务约定

- 授权
  - 注意：http协议是无状态的， 客户端对 服务器的每个请求都要求认证。
  - 基本认证

```
curl -v -u"yourid"  https://api.github.com/
```

  - OAuth2 token

```
curl -H "Authorization: token OAUTH-TOKEN" https://api.github.com
curl https://api.github.com/?access_token=OAUTH-TOKEN
```

  - 服务之间的认证

```
curl 'https://api.github.com/users/whatever?client_id=xxxx&client_secret=yyyy'
```

  - 一个应用多个服务之间每次请求都需要认证，因此需要一些基础设施来简化服务编程

基于HTTP的RESTful服务约定

- 参数
  - 利用查询字符串（HTTP query string）

```
curl -i "https://api.github.com/repos/vmg/redcarpet/issues?state=closed"
```

  - 例如请求体（HTTP body），常用媒体类型包括 application/json 或 application/x-www-form-urlencoded

```
curl -i -u username -d '{"scopes":["public_repo"]}' https://api.github.com/authorizations
```

- 端点（endpoint）
  - 根端点应返回端点目录

```
curl https://api.github.com
```

- 错误.....

基于HTTP的RESTful服务约定

- 资源操作（CRUD）与HTTP verbs

Verb	Description
`HEAD`	Can be issued against any resource to get just the HTTP header info.
`GET`	Used for retrieving resources.
`POST`	Used for creating resources.
`PATCH`	Used for updating resources with partial JSON data. For instance, an Issue resource has `title` and `body` attributes. A PATCH request may accept one or more of the attributes to update the resource. PATCH is a relatively new and uncommon HTTP verb, so resource endpoints also accept `POST` requests.
`PUT`	Used for replacing resources or collections. For `PUT` requests with no `body` attribute, be sure to set the `Content-Length` header to zero.
`DELETE`	Used for deleting resources.

基于HTTP的RESTful服务约定

- Cross origin resource sharing
- JSON-P callbacks
- 。 。 。 。

Golang 本地数据库 - BoltDB

- 为什么需要 local KV Database
  - Web 服务需要高性能缓存
  - 作为简单应用程序的数据库
    - 不能作为无状态服务的数据库
  - 数据库代码入门
- BoltDB核心概念
  - 事务
    - 全局互斥的读写事务，即任何时刻至多有一个写事务执行
    - 全局共享读事务，即有多个读事务并行
  - 桶（buckets）
    - K-V对（[]bytes – []bytes）
    - 子桶
    - 键值管理

官网说明 https://github.com/boltdb/bolt 不负责任的中文翻译

Golang RESTful 服务实现

- 服务端开发
  - 任意 web 框架都可以，是一个简单的事情
  - 问题：
    - 自动由接口生成 web 服务
    - 自描述机制
    - 自动协商机制
- 客户端开发
  - Go http 库

Web 服务 API 及其设计与实现

- RPC 风格
  - Apache/thrift
  - Go rpc
  - Go json-rpc
  - gRPC
- REST 风格
  - Swagger
  - Postman
  - Blueprint(Markdown)
  - RAML
  - Apizaa
- QraphQL

RPC风格服务

- RPC：远程过程调用(Remote Procedure Call)。它是一个客户端-服务器通讯的协议，允许客户端通过网络调用服务端的接口函数。
  - 模型：把服务器看成一个黑盒组件，通过request-response 完成函数调用

  - 机制：

  - 规范
    - ?? IDL（接口定义语言）

维基百科：https://en.wikipedia.org/wiki/Remote_procedure_call

Apache thrift

- 一个软件框架支持跨语言服务开发
  - 官方网站：http://thrift.apache.org/
- 使用步骤
  - 下载工具
  - 编写 thrift IDL 描述文件，例如：test.**thrift**
  - 产生指定语言的服务器或客户端框架代码，如go、c++、java、python 等
  - 填写程序业务逻辑
- Thrift 介绍
  - 2012，Apache Thrift - 可伸缩的跨语言服务开发框架，https://www.ibm.com/developerworks/cn/java/j-lo-apachethrift/
- Go 语言开发的案例
  - Golang开发Thrift接口，http://blog.cyeam.com/golang/2014/07/22/go_thrift

Go语言RPC

- 传统 RPC 的问题：机制复杂，承担了
  - 跨语言交互操作
  - 函数参数的编码格式与效率
  - 支持多种通讯链路
  - 同步与异步支持
  - 并发与安全管理
- Go 语言的 rpc（轻量，实用）
  - 支持一种函数形式（一元 RPC）
  - 支持有限编码格式 gob， json
  - 支持链路 TCP，HTTP
  - 支持同步与异步操作
- 官方支持
  - net/rpc
  - net/jsonrpc
    - 标准 jsonrpc-v1
    - https://www.jsonrpc.org/specification_v1

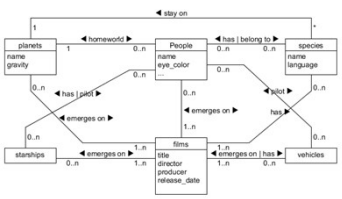
gRPC

- 基于 ProtoBuf 编码，支持 http2 的 thrift
- What is gRPC?
  - https://grpc.io/docs/guides/
  - 数据格式
    - Protocol Buffers - Google's data interchange format
    - Google Protocol Buffer 的使用和原理，https://www.ibm.com/developerworks/cn/linux/l-cn-gpb/index.html
  - IDL （Protocolbuf + ServiceInterface）
    - 一元（Unary）RPC，即 response operation(request)
  - HTTP2
    - steaming 支持（双向）
    - 认证
- Go Quick Start
  - https://grpc.io/docs/quickstart/go.html

gRPC 概念，中文翻译：https://blog.csdn.net/u010918487/article/details/79958704

REST 风格服务

- Demo 网站
  - SWAPI（The Star Wars API）https://swapi.co/
  - API root
- Resource
  - Data model
  - Domain model
  - Conceptual mod



- Encoding
  - Json

Swagger 描述语言与工具

- 什么是 Swagger?
  REST API 一种标准与一组工具集
  - OpenApi Specification (OAS)
  - 设计工具
    - 支持 API 优先设计的理念
  - 构建工具
    - 生成多种语言服务器、客户机程序原型
    - 支持 API 代码规范性与一致性检查
  - 文档工具（服务器代码与文档双向生成）
  - 测试工具（支持基于规范接口的mock测试）
    - 支持客户端与服务器开发分离
    - 实现在线mock测试服务器，独立测试客户端
    - 实现在线mock客户端，独立测试服务器

官网 https://swagger.io/

REST API 设计工具

- Swagger editor
  - Yaml OAS 可视化工具
  - 版本
    - Swagger editor 单机版
    - SwaggerHub 在线团队版
  - 版本特性差异
    - https://swagger.io/tools/swagger-editor/download/
  - 在线演示
    - https://swagger.io/tools/swagger-editor/
    - 演示案例： **Swagger Petstore**：https://petstore.swagger.io/
- 主要功能
  - 编写 API 规范描述文件 yaml
  - 可视化，可执行的的客户端
  - 生成服务器、客户端基本代码

OpenAPI 规格说明主要对象

- swagger:#version
- host:#domain-name
- basepath:#api-base
- schemes:
- paths
  - /pet/{id}
    - Get
    - Post
- securityDefinitions
- definitions
  - Pet

测试工具

- Swagger Inspector
  - https://swagger.io/tools/swagger-inspector/

Go swagger

- golang implementation of Swagger 2.0 (aka OpenAPI 2.0)
- Version
  - 0.17.x
- **Features**
  - Generates a server from a swagger specification
  - Generates a client from a swagger specification
  - Supports most features offered by jsonschema and swagger, including polymorphism
  - Generates a swagger specification from annotated go code
  - Additional tools to work with a swagger spec
  - Great customization features, with vendor extensions and customizable templates
- CLI tools
  - ...
- Documents
  - https://goswagger.io/

Blueprint - apiary

- How apiary works
  - https://apiary.io/how-apiary-works
- TDD Workflow
  1. 用 github 账号在 apiary.io 登陆
  2. 用 markdown 编写 api
  3. 生成 mock server 在线
  4. 制作客户端原型，使用 api 演示应用
  5. 返回 2 修改，直到原型成功
- 微服务TDD教程
  - 在Go中构建微服务
    https://blog.csdn.net/karamos/article/details/80130988

GraphQL

- REST 本质
  - 对资源的CRUD
  - 资源由网状数据模型定义
- GraphQL
  - 面向资源的查询语言
  - 基本语法

Operation :: 动词（参数）{资源，资源，...}
资源 :: 资源名称（过滤器）{资源，资源，...}

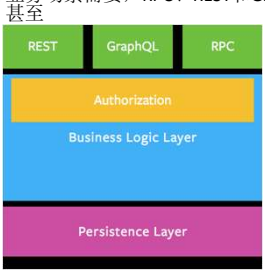
  - 动词
    - Query（查询）, mutation（修改），subscribe（订阅）
  - 面向数据的协议
    - 通讯协议无关，语言无关
  - 核心术语
    - Schemas and Types
    - Operations and Selections

GraphQL 资源与工具

- 英中文一致，翻译准确、且强大的官网
  - 英文 https://graphql.org/
  - 中文 https://graphql.cn/
- 阅读指南
  - 首页 demo 与 特征
  - 学习 GraphQL 入门
    - 最佳实践！
  - 代码
    - Go 语言服务端
    - JavaScript 客户端
    - 工具
    - BaaS 服务
  - 规范 – “GraphQL是一种在还在演进的未完成的新语言”

RPC pk REST pk GraphQL API

- 处于种种业务场景需要，RPC、REST和GraphQL 将长期共存，甚至



图片来源：http://graphql.cn/learn/thinking-in-graphs/

RPC pk REST pk GraphQL API

描述	RPC	REST	GraphQL
基本理念（远端服务的内容）	模块（函数）	资源	有关系的数据
定义语言	Interface DL	Resource CURD	Data Query
传输协议	TCP 等	HTTP	---
数据编码	Thrif，gob，json，proto	Json,xml html...	GraphQL Json
与 javascript 互操作	差	优	较优
性能	优	一般	较好
多语言互操作	少	任意	---
服务伸缩性	难确定	确定	确定
标准化	厂商主导	社区主导	社区主导
易开发性（目前）	易	一般	难
合适传输数据类型	函数参数	超媒体	结构化数据
成熟度	高	较高	一般