RAML介绍

RAML介绍

> RAML是RESTFul API Modeling Language的缩写。
> ——最简单的设计API的设计方式

为讨厌文档，喜欢编码的程序员量身定做

——Somebody

Machine process-able
Human readable

Visual

The code generators

API Notebook

RAML介绍

包含以下几个大的方面

- 提供API基本信息—Basic Information
- 用户文档—User Documentation
- 资源类型跟特征—Resource Types and Traits
- 资源列表—Resources.

版本选择

V1.0

V0.8

RAML结构

For every API, start by defining which version of RAML you are using, and then document basic characteristics of your API - the title, baseURI, and version.

Create and pull in namespaced, reusable libraries containing data types, traits, resource types, schemas, examples, & more.

Annotations let you add vendor specifc functionality without compromising your spec

Traits and resourceTypes let you take advantage of code reuse and design patterns

Easily define resources and methods, then add as much detail as you want. Apply traits and other patterns, or add parameters and other details specific to each call.

Describe expected responses for multiple media types and specify data types or call in pre-defined schemas and examples. Schemas and examples can be defined via a data type, in-line, or externalized with !include.

Write human-readable, markdown-formatted descriptions throughout your RAML spec, or include entire markdown documentation sections at the root.

```yaml
#%RAML 1.0
title: World Music API
baseUri: http://example.api.com/{version}
version: v1

uses:
  Songs: !include libraries/songs.raml

annotationTypes:
  monitoringInterval:
    parameters:
      value: integer

traits:
  secured: !include secured/accessToken.raml

/songs:
  is: secured
  get:
    (monitoringInterval): 30
    queryParameters:
      genre:
        description: filter the songs by genre
  post:
  /{songId}:
    get:
      responses:
        200:
          body:
            application/json:
              type: Songs.Song
            application/xml:
              schema: !include schemas/songs.xml
              example: !include examples/songs.xml
```

Songs Library

```yaml
#%RAML 1.0 Library
types:
  Song:
    properties:
      title: string
      length: number
  Album:
    properties:
      title: string
      songs: Song[]
  Musician:
    properties:
      name: string
      discography: (Song | Album)[]
```

songs.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="song">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="title" type="xs:string">
          </xs:element>
          <xs:element name="artist" type="xs:string">
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
</xs:schema>
```

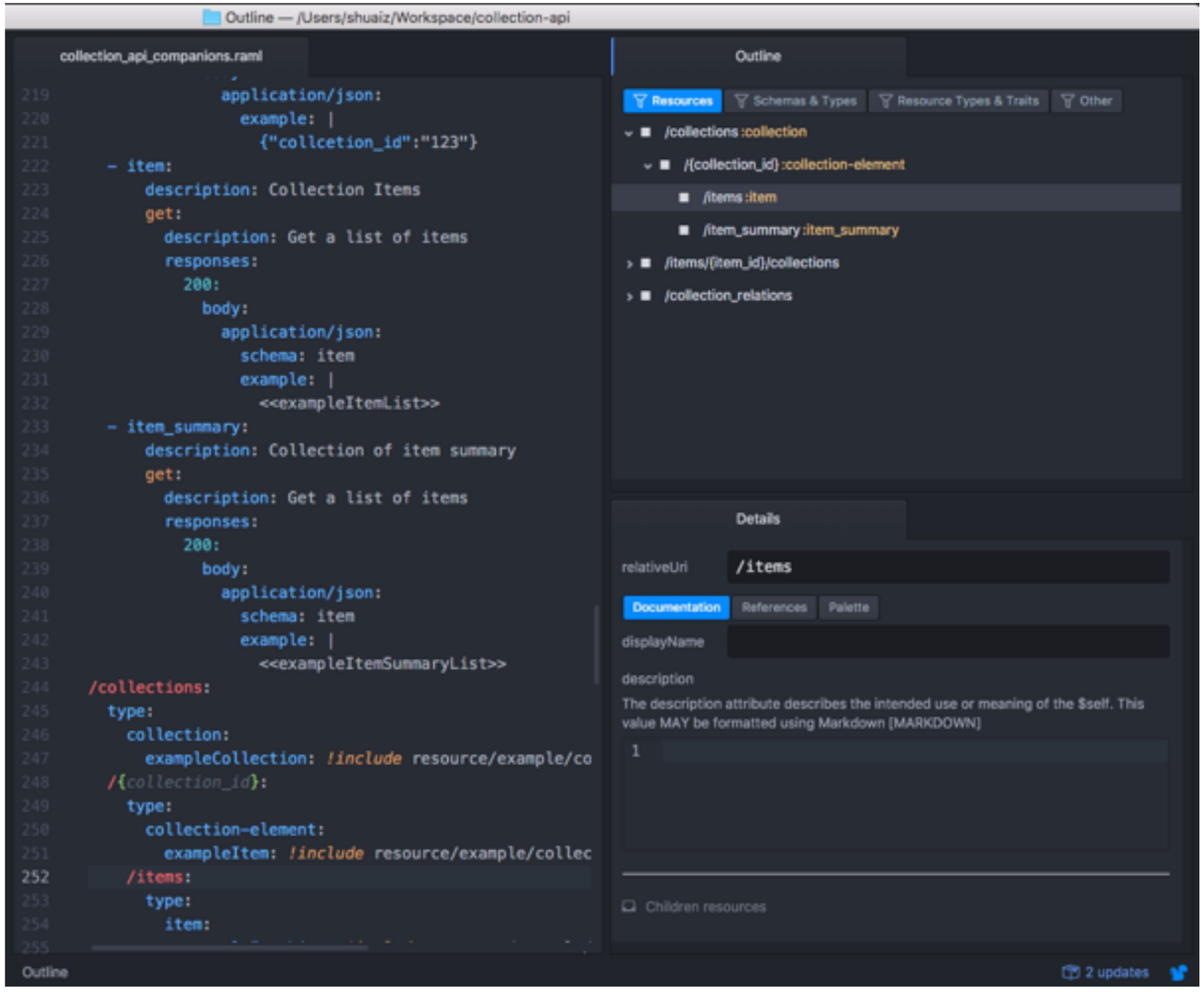
RAML可用的编译器

Atom

+

API Workbench

- 语法检查
- 内容识别
- 直观的输出



ThoughtWorks®

基本语法

关于RAML基本的用法

基本语法

基本的RAML第一行必须是以下面的内容开头：

```
#%RAML 0.8
```

必须包含的元素：

```
title: <API Title>
```

更多的信息：

点我

ThoughtWorks®

进阶语法

RAML的进阶用法，可以提高代码的复用性

进阶语法

Schema—JSON对象的约束

- 学习地址

Resource Type—定义一个符合CRUD操作的可重用资源

一般写法：

```
resourceTypes:
  - resource-name:
      CURD
```

Reserved Parameters: 保留的关键字

- <<resourcePath>> : /songs
- <<resourcePathName>> : songs

Parameters Transformers： 关键字转化（只支持英语）

- !singularize 单数
- !pluralize 复数

可以复写resourceType里面的定义的方法（类似OOP的继承概念）

Parameters： 自定义的替换关键字

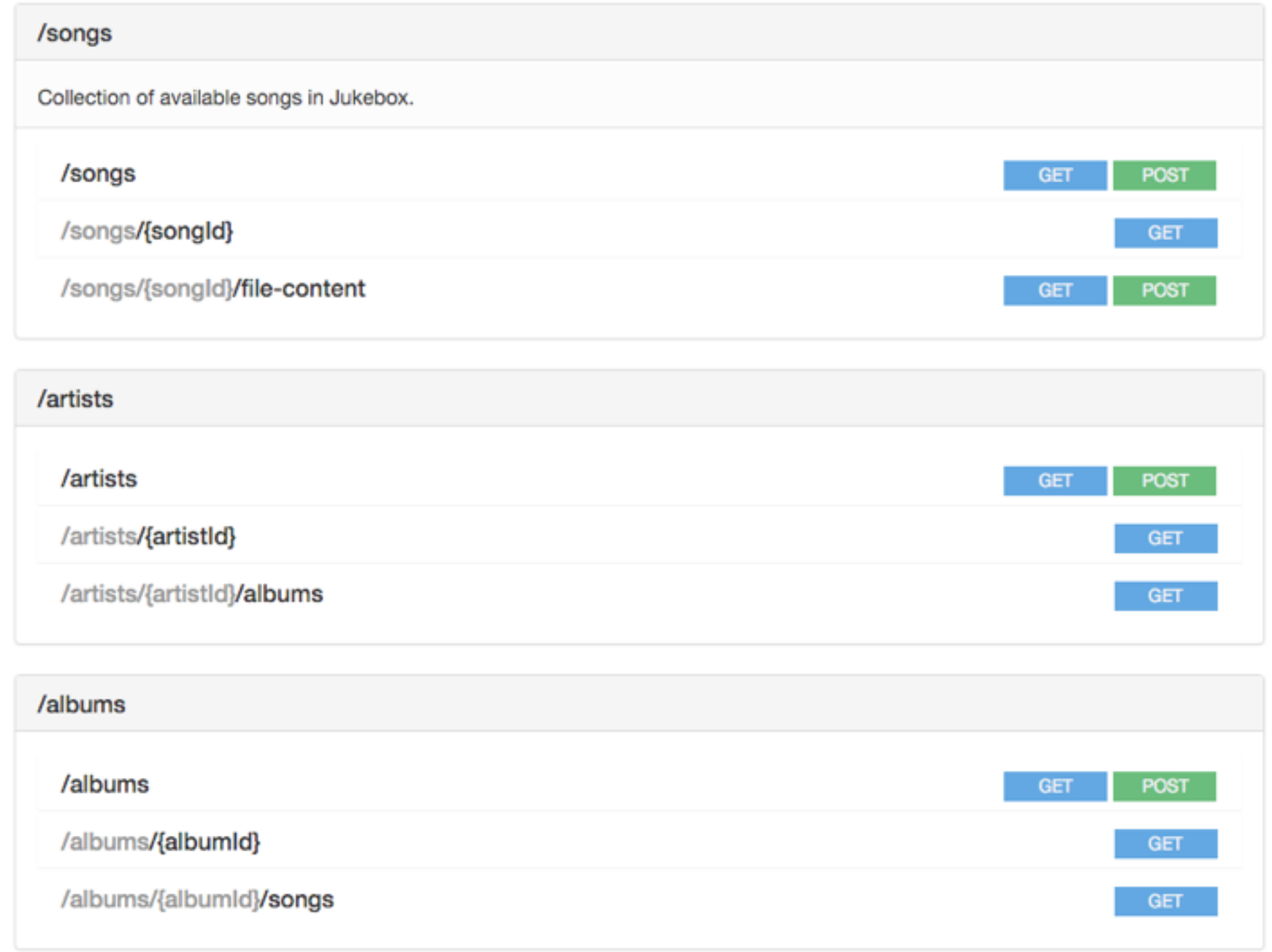
- <<custom name>> must be implement when using this resource type

Include—JSON对象的约束

- **!include 文件的分割**

CODE

RAML的可视化


/songs
Collection of available songs in Jukebox.
/songs
GET
POST
/songs/{songId}
GET
/songs/{songId}/file-content
GET
POST
/artists
/artists
GET
POST
/artists/{artistId}
GET
/artists/{artistId}/albums
GET
/albums
/albums
GET
POST
/albums/{albumId}
GET
/albums/{albumId}/songs
GET

Q&A

谢谢

有问题请联系

张帅 & 尚菲
shuaiz@thoughtworks.com
fshang@thoughtworks.com

ThoughtWorks®