resty Documentation

Release 0.1.1

Core Services Team

February 08, 2017

Contents

1 Installation **3**
- 1.1 Install `resty` . . . 3

2 Getting started tutorial **5**
- 2.1 Step 1: Configure the client . . . 5
- 2.2 Step 2: Load the entrypoint and select a service . . . 5
- 2.3 Step 3: Use filters . . . 6
- 2.4 Step 4: Iterating through items . . . 6
- 2.5 Step 5: Accessing metadata and usefull content . . . 6
- 2.6 Step 6: Using links to interact with available relationships . . . 6

3 What's new in resty 0.1? **9**
- 3.1 Features . . . 9

4 Indices and tables **11**

`resty` library provides user-friendly abstractions that allow you to easily interact with RESTful API. This documentation contains guidance on how to install and use the library.

`resty` is:

- simple high-level API interaction
- easy to use
- smart

Contents:

CHAPTER 1

Installation

This tutorial will walk you through the process of installing resty

1.1 Install `resty`

The installation can be easy done by using the *easy_install*:

```
easy_install resty
```

You can upgrade your older version too:

```
easy_install --upgrade resty
```

Or using *pip*:

```
pip install resty
```

CHAPTER 2

Getting started tutorial

This tutorial aims to get you started with `resty` as quickly as possible.

It is made up of a number of examples of increasing complexity, each of which shows one or more useful `resty` features.

If you have any comments about these tutorials, or suggestions for things we should cover in them, then please contact us via Github

2.1 Step 1: Configure the client

`resty` provides two types of clients:

- the default `client` - uses summaries if available therefore the interactions with the api are optimized
- `dumb_client` - does not use summaries and relies strictly on links

Note: The summary is composed from all additional attributes (not required) that a minimum document representation has. The minimum documentation representation is used whenever a document has a relationship to another document.

2.2 Step 2: Load the entrypoint and select a service

The URL where the Service document is located is named the API Entrypoint. In the provided example we used `http://services.pbs.org/` url as an entrypoint and selected the zipcodes service from the available services:

```
>>> from resty import client
>>> c = client.load("http://services.pbs.org/")
>>> c
<resty.types.Service object at 0x26abd10>

>>> zipcode_collection = c.service("zipcodes")
<resty.types.Collection object at 0x2597e90>
```

Note: The call to `c.service("zicpodes")` will select the top level collection named `zipcodes`.

2.3 Step 3: Use filters

The `filters` are used to describe complex interactions that usually require some sort of human input. One particularly common situation is searching trough the elements of a collection. Templates are available only in collections. Since `zipcode_collection` returns a collection we can filter it based on zip.

```
>>> filtered_zipcodes = zipcode_collection.filter('zip', zipcode='22202')
```

Note: The `filter` method takes one positional argument that represents the filter name and a number of keyword arguments where the keys represent the placeholder names and the desired values.

2.4 Step 4: Iterating through items

Items represents the list of objects available in that collection. In the above example the `filtered_zipcodes` returns a collection with a single object. Let's select the first object from the list:

```
>>> zipcode_resource = filtered_zipcodes.items()[0]
>>> zipcode_resource
<resty.types.Resource object at 0x259fc50>
```

2.5 Step 5: Accessing metadata and usefull content

At this point we have a `zipcode_resource` and we can extract informations like metadata and content specific informations

```
>>> print zipcode_resource.content.zipcode
u'22202'
>>> print zipcode_resource.class_
u'Zipcode'
```

Note: For example when representing a document in json, properties which are prefixed with $ are considered metadata.

2.6 Step 6: Using links to interact with available relationships

Using the `related` method one can get from a document to a related document by specifying the relationship name.

Let's see all the callsigns that are available for zipcode 22202 with their corresponding confidence:

```
>>> callsign_collection = zipcode_resource.related('search')
>>> for c in callsign_collection.items():
>>>     print c.related('related').content.callsign, c.content.confidence
WETA 100
WMPB 100
WWPB 100
WHUT 100
WFPT 100
```

```
WVPY 100
WMPT 100
WGTV 80
KRMA 80
WTTW 80
WTVS 0
KCTS 0
KSPS 0
WGBH 0
WNED 0
```

Note: The `related` method takes one argument that represents the relationship name. In case there are multiple relationship with the same name one can pass an additional argument representing the related resource class or in case of a colletion the type of the elements. In the above example the call to `c.related('related')` is equivalent to `c.related('related', 'Callsign')`

CHAPTER 3

What's new in resty 0.1?

We are pleased to announce the initial release of the `resty` library.

As this is the first version of `resty` library, these are the main features.

3.1 Features

- simple high-level API interaction
- easy to use
- smart

CHAPTER 4

Indices and tables

- genindex
- modindex
- search