

MyOhData

Overview

This is a simple OData server using CherryPy and Jinja2. It doesn't support much of the [OData](#) spec. It's basically the result of me trying to wrap my head around [OData](#) and how to provide it.

Installation

Run following commands at a terminal (I suggest doing this in a [virtualenv](#)):

```
$ tar xzvf myohdata-1.0.tar.gz
$ cd myohdata-1.0/
$ easy_install "CherryPy >= 3.2" Jinja2
$ python myohdata.py
```

That will launch an [OData](#) service running on *localhost:8011* with some sample data.

OData Features

This implementation is rather slim on features. You can host multiple catalogs, list the entries in catalogs and access individual entries. That's it. It supports a subset of the Atom version of OData. It is read-only.

The `/`, `/?$metadata`, `/<Catalog>` and `/<Catalog>(<id>)` endpoints have been validated against <http://services.odata.org/validation/> and conform to all of their rules as of 2011-04-22.

Advanced

Here are some details in case you want to play with more than just the one sample catalog. The data source that the app reads from is a Python dictionary. The dictionary represents *catalogs* of *entries*.

Catalogs

Catalogs should be named with strings. Here is an example:

```
all_data = {'Widgets':{ ... }}
```

One catalog named "Widgets" is defined there. What should a catalog contain, you ask? Metadata and entries is the answer!

Catalog Metadata

The catalog metadata describes the content of the catalog - namely, the structure of the entities that it contains. It is a sub-dictionary off of the catalog and must be keyed with `_meta`. Here are the required fields.

key

str - The field in each entity that represents the unique key that identifies the entity.

title

str - The field in each entity that represents the display title of the entity.

name

str - The name of the entity. Usually singular -- *Widget*, for example.

set

str - The name of a set of the entities. Usually plural -- *Widgets*. You get the idea.

properties

list of tuples - Each tuple in this list represents a property on the entity. The tuples should contain three values.

1. *str* The name of the property.
2. *str* The [EDM type](#) of the property's value.
3. *str* Whether the property can be null -- either "true" or "false".

Here is an example catalog with metadata (no entries):.

```
{ 'Widgets':  
  '_meta': {  
    'key': 'ID',  
    'name': 'Widget',  
    'set': 'Widgets',  
    'title': 'Name',  
    'properties': [  
      ('ID', 'Edm.Int32', 'false'),  
      ('Name', 'Edm.String', 'false'),  
      ('Function', 'Edm.String', 'true'),  
    ],  
  },  
}
```

Catalog Entries

Conceptually, an entry is an instance of an entity. Every other key/value pair in the catalog must represent an entry. The *key* must be equal to `value[meta['key']]`. The *value* should be a dictionary that maps to the properties defined for the entity. Here is an example of a catalog of entries that corresponds to the [metadata example](#) above:

```
{ 'Widgets':  
  ...  
  1: { 'ID': 1, 'Name': 'Foo', 'Function': 'Doing foo-like things.' },  
  2: { 'ID': 2, 'Name': 'Bar', 'Function': 'Processing bars.' },  
  3: { 'ID': 3, 'Name': 'Baz', 'Function': 'BAAAAZ!.' },  
  ...  
}
```

Complete Catalog Example

Putting it all together, we get:

```
{ 'Widgets':  
  '_meta': {  
    'key': 'ID',  
    'name': 'Widget',  
    'set': 'Widgets',  
    'title': 'Name',  
    'properties': [  
      ('ID', 'Edm.Int32', 'false'),  
      ('Name', 'Edm.String', 'false'),  
      ('Function', 'Edm.String', 'true'),  
    ],  
  },  
  1: { 'ID': 1, 'Name': 'Foo', 'Function': 'Doing foo-like things.' },  
  2: { 'ID': 2, 'Name': 'Bar', 'Function': 'Processing bars.' },  
  3: { 'ID': 3, 'Name': 'Baz', 'Function': 'BAAAAZ!.' },  
}
```

```
        ('ID', 'Edm.Int32', 'false'),
        ('Name', 'Edm.String', 'false'),
        ('Function', 'Edm.String', 'true'),
    ],
},

1: {'ID':1, 'Name':'Foo', 'Function':'Doing foo-like things.'},
2: {'ID':2, 'Name':'Bar', 'Function':'Processing bars.'},
3: {'ID':3, 'Name':'Baz', 'Function':'BAAAAZ!.'},
}
```

TODO

I doubt that I'll add these features, but the following would be nice:

- [Query options](#)
- [Associations](#)
- Full CRUD operations
- Support for paging large catalogs

Contact

I'm Christian Wyglendowski. I work for [YouGov](#). You can find me on the web on [my site](#), [Twitter](#) or shoot me an email (christian@dowski.com).