**Swordfish API – Developer Documentation**

Table of Contents

[1) Requirements: 2](#_Toc516233891)

[2) Overview: 2](#_Toc516233892)

[2.1 Static Resources: 2](#_Toc516233893)

[2.2 Dynamic Resources: 2](#_Toc516233894)

[2.3 Static VS Dynamic Resources: 3](#_Toc516233895)

[2.4 Mockups and templates: 3](#_Toc516233896)

[2.5 Resource\_manager.py file: 3](#_Toc516233897)

[2.6 Utils.py file: 5](#_Toc516233898)

[3) Implementing Dynamic structure 5](#_Toc516233899)

[3.1 Writing Template Files: 5](#_Toc516233900)

[3.2 Writing API Files: 7](#_Toc516233901)

[4) Redfish Emulator :- 9](#_Toc516233902)

[5) Swordfish Functionality: 12](#_Toc516233903)

[5.1 StorageServicesAPI file 13](#_Toc516233904)

# Requirements:

**Python version:** 2.7, 3.1, 3.6

**Ubuntu:** 16.4v

**Windows:** 7,8 and 10

# Overview:

The Swordfish API Emulator can emulate a Swordfish-based system that responds to create, read, update, and delete RESTful API operations to allow developers to model new Swordfish functionality, test clients, demonstrate Swordfish, and do other similar functions. The Swordfish API Emulator extends the DMTF Redfish Interface Emulator by adding code to an installation of the Redfish Interface Emulator code. Review “**Redfish Interface Emulator**” Documentation for more reference.

The Redfish Interface Emulator contains Static and Dynamic resources. Static resources are copy of redfish mockups in ./static directory. Dynamic Resources are accomplished by creating python files. Redfish creates API files and Template files Using Code-generators exists in ./codegen directory .

Swordfish Interface Emulator resources are dynamic. It contains Template and API files for Singleton and collection resources . Swordfish files are stored in ./Resource Directory where CRUD operations can be done on dynamic Resources using REST operations.

## 2.1 Static Resources:

Static resources are in api\_emulator/redfish/static directory. Redfish Emulator have instructions for static mockups that can be found in Redfish Interface Emulator doc’s folder or Readme file.

## 2.2 Dynamic Resources:

Dynamic resources can have following properties:

* It can create resources and sub-resources using template structure via POST operation.
* Can change the properties of a specific resource.
* GET, POST, PUT & DELETE operations are used.

## 2.3 Static VS Dynamic Resources:

Static resources are populated by placing appropriate JSON mockup folders into the ./static directory.

Dynamic Resources in emulator can be populated through APIs using CRUD operations. In Swordfish Emulator New resources are created using template structure.

## 2.4 Mockups and templates:

Mockups are just an example view or static view of possible configuration. Mockups will simply provide the overview of all swordfish models.

Template is a base structure for creating new element in a collection / Sub Collection. We can create members of collection using template structure. Template structure is designed based on latest SNIA Swordfish latest Schema V1.0.5

Template file divides in to two types of sections.one is “template declaration” and other is “function”

## 2.5 Resource\_manager.py file:

* The redfish root structure is defined in Resourcemanager.py file. So now developer has to make some suggested changes to run swordfish emulator.
* Redfish emulator already contains static and dynamic resources in resourcemanager.py file. It already contains static resources from ./redfish/static directory which is similar/copy to mockup data.
* ResourceManager class can load static resources and dynamic resources.
* Swordfish developerhas to attach API for dynamic resources.
* All the API files are stored in ./api\_emulator/swordfish directory.

Fig A: Imports all the api files

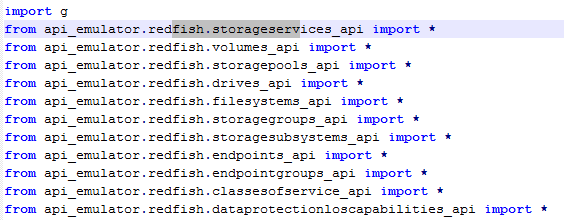
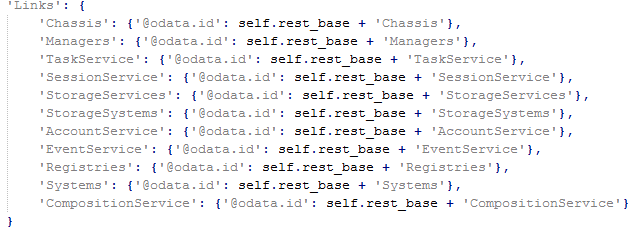


Fig B: Attaching API for dynamic resources

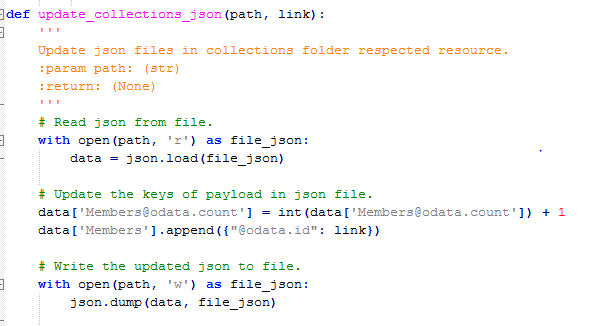


Fig C: Adding Collections to



## 2.6 Utils.py file:

* Added new service for members count.



# Implementing Dynamic structure

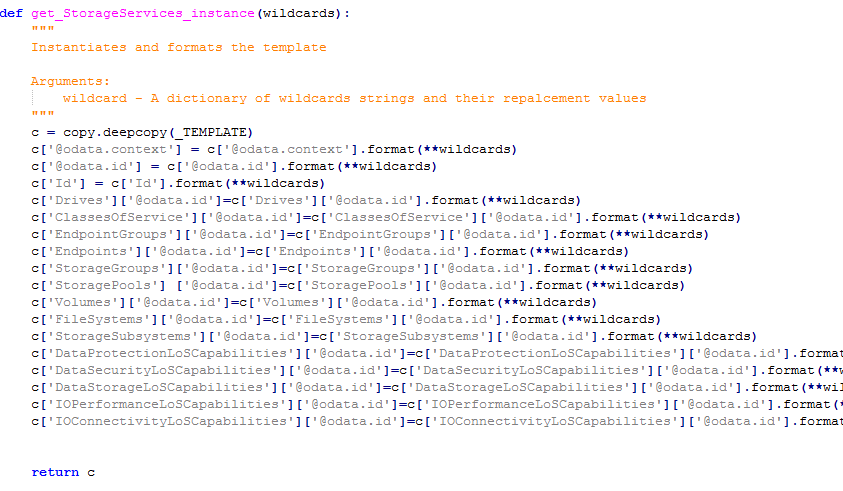
* The static resources are in /api\_emulator/redfish/static/.. Folder, these are just a simple copy.
* The dynamic resource is in /Resource/...Folder. We can specify any path to declare the dynamic resources.
* Developer has to create a template file and api files for dynamic structure.
* The file in template directory and api directory are to be paired appropriately.

## 3.1 Writing Template Files:

* Template file directory is /api\_emulator/swordfish/template
* Template declaration is as similar to json file (shown in below figure). All the properties for a particular resource are to be added by referring the updated schemas released by SNIA-Swordfish.
* Template declaration also contains substitution fields
* Each template contains similar fields {rb}&{id} which are declared in below function are replaced by root\_base(/redfish/v1) and ID .

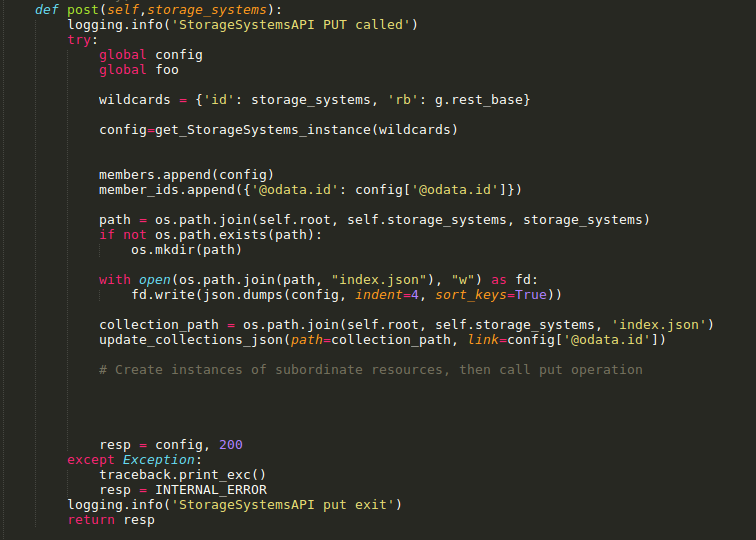


* Below figure contains function, which is used to call the instance of resource.
* We are using wildcards for replacing the values.
* Using deep-copy is to make copy of template declaration.
* {rb} –{rest-base} is default set to {redfish/v1}

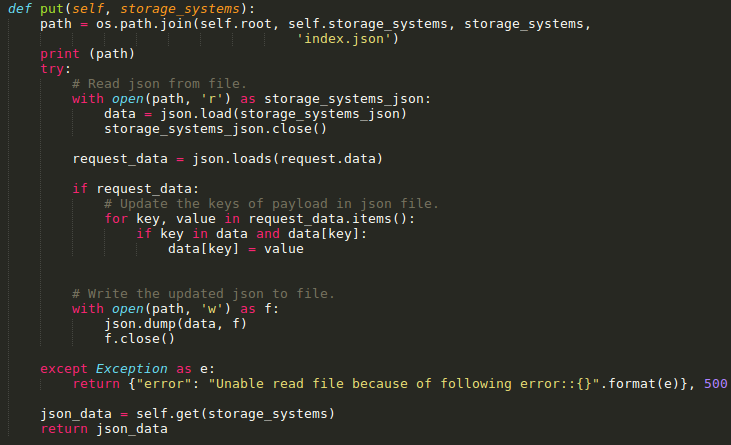


## 3.2 Writing API Files:

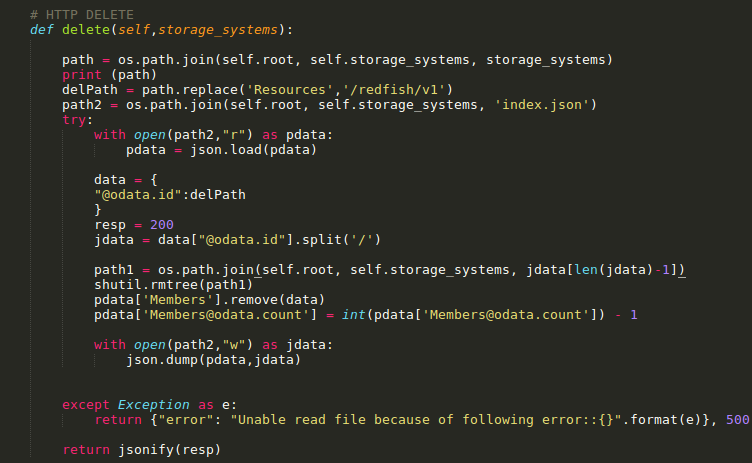
* We have to import the above template function to add a new element in to collection.
* The API file commonly used to retrieve data form remote systems. Using API , we can make request to web server and retrieve data that is needed. In emulator APIcode will interacting with all collections and member resources.
* For all Collection-API’s , GET and POST methods are defined, post command will create a sub resource API to the interface.
* For all API’S there are API classes and collection classes.
* The http functions GET, POST, PUT and DELETE are declared based on requirements.
* The Post function can create resources ,update members and members.id
* Post function can also attach API’s of sub resources and creates instance of sub-resources.



* PUT function is used to edit specific properties in a collection.



* Delete Function will try to delete a item from members list . after deleting resource , it will update members and members count in collection.



# Redfish Emulator :-

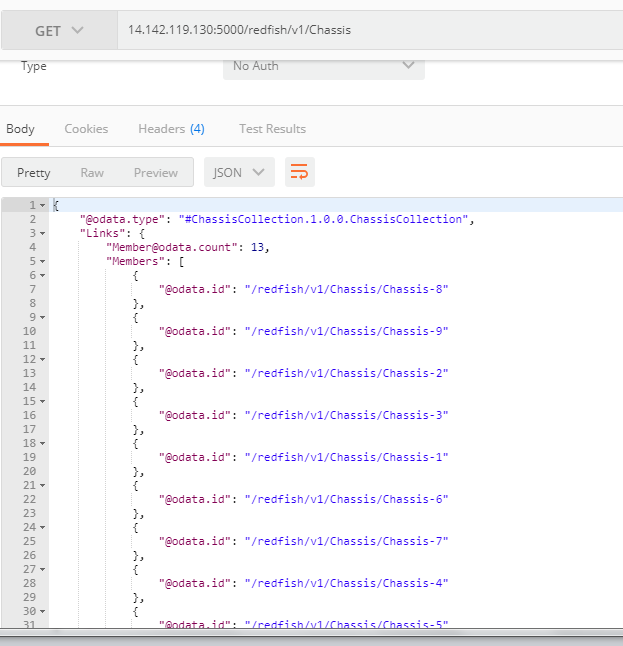
In Redfish Emulator dynamic files are stored in memory. Redfish emulator loads all the static and dynamic files. Emulator used to update and delete an item from collection by passing “id”. After making all changes if redfish used to restart its emulator, the data in memory will be lost and it comes with a new version of redfish.

Redfish automatically creates Template files and API files using “codegen”

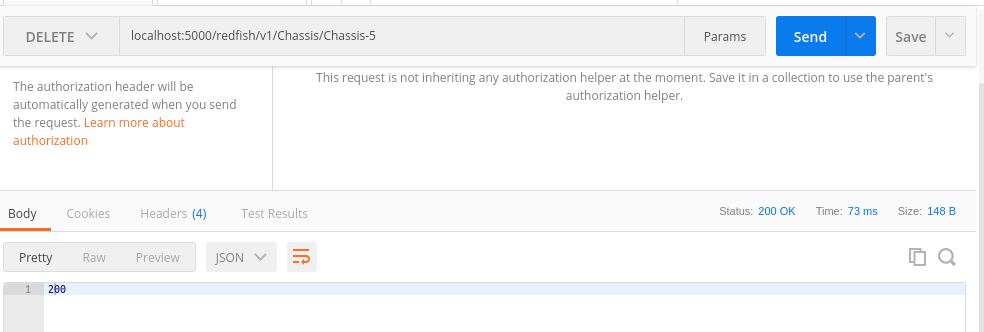
Whenever we start emulator, Redfish used to automate template and api files

Example:

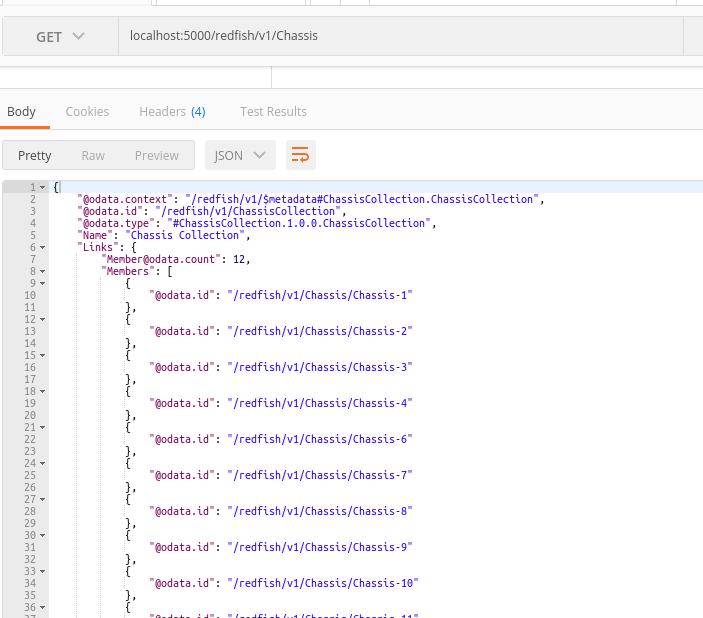
Below figure shows the main collection of chassis, which contains “13” items in a collection. We can go through each member of collection individually. We can update the collection and delete a collection.



The below figure shows deleting “Chassis-5” from collection



From below figure members and member count are updated . Chassis-5 is not in members list.

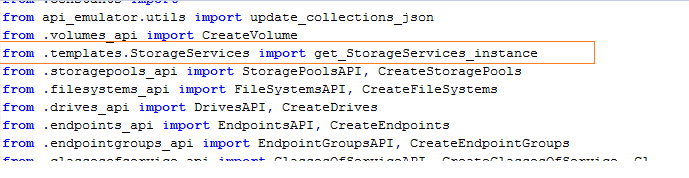


But if we try to restart the emulator, It automatically creates chassis member “chasiss-5 “

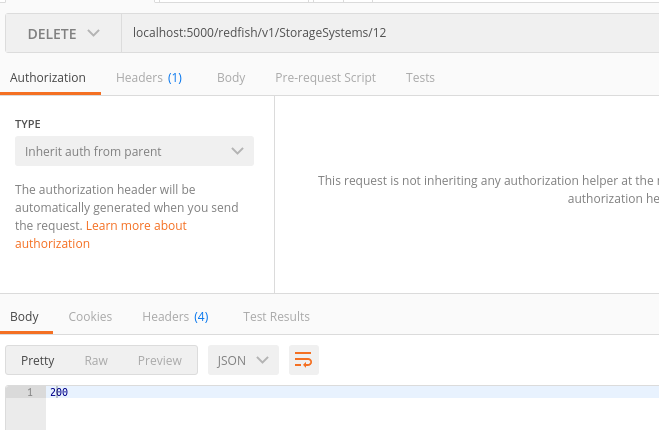
As its storing in memory it won’t save the previous updated data. It stores in cache and it automatically re-collects everything form latest changes.

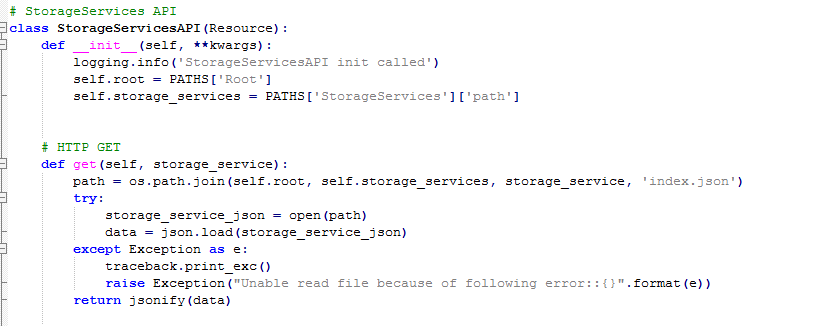
# Swordfish Functionality:

* For Example, If we want to create StorageServices as a dynamic resource using template and API files.
* StorageServices is a member of collection. We have to create template and API files with similar name.
  + Template file path is /api\_emulator/swordfish/template/StorageServices.py
  + API file path /api\_emulator/swordfish/storageservices\_api.py
* **Template** file contains two sections as shown in figure.
  + Template declaration
  + Function ("get\_StorageServices\_instance")



* Storageservices\_api.py file contains two different classes. Api class and collection class.
* StorageServices collectionAPI file
* By using url (/redfish/v1/StorageServices) we can “GET” members and collection data
* We can delete an item from collection using “Delete”.



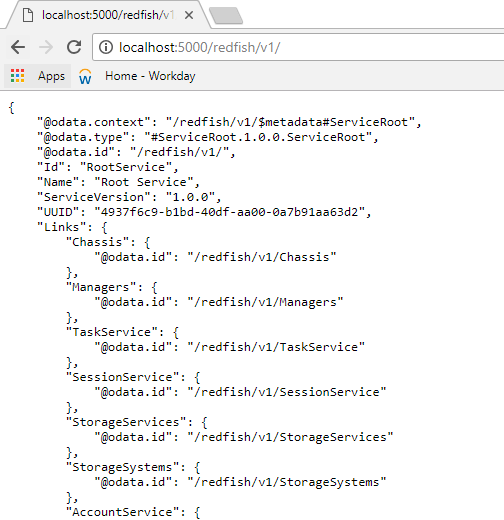


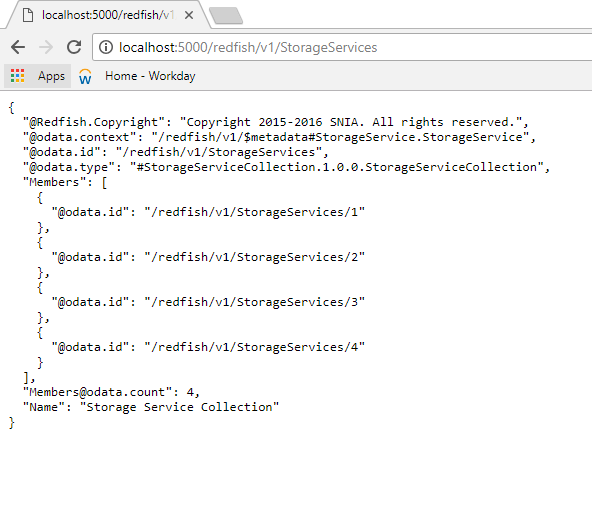
## 5.1 StorageServicesAPI file

* + Add new elements of a collection using “post”. By calling template file we can create a base structure for new element .

**Example**:

Go through Storageservices resource by following steps.



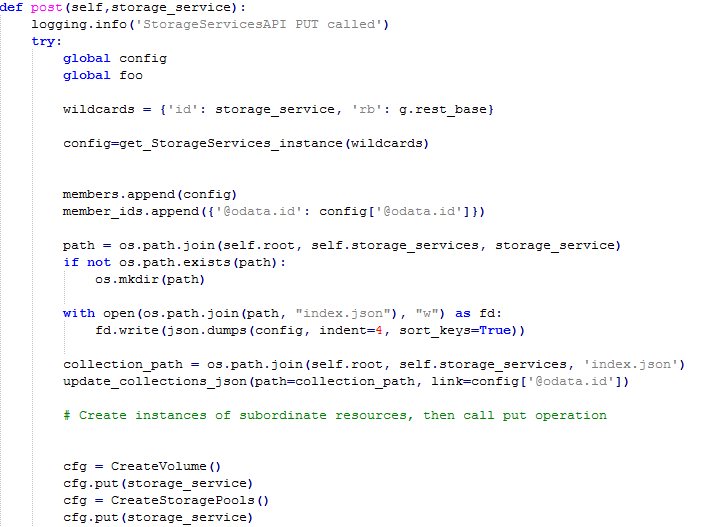


Now we are in **/redfish/v1/storageservices**, here the above figure shows that there are 4 members created in storageservices. If we ant to create more members we have to use “POST” function.

The subresources are created at the time of member creation by writing logic in “post”Function AS shown in the figure below.



The logic of the above output is show in below figure.



We are creating by passing “post” method. We are creating instance of subresources using post operation.

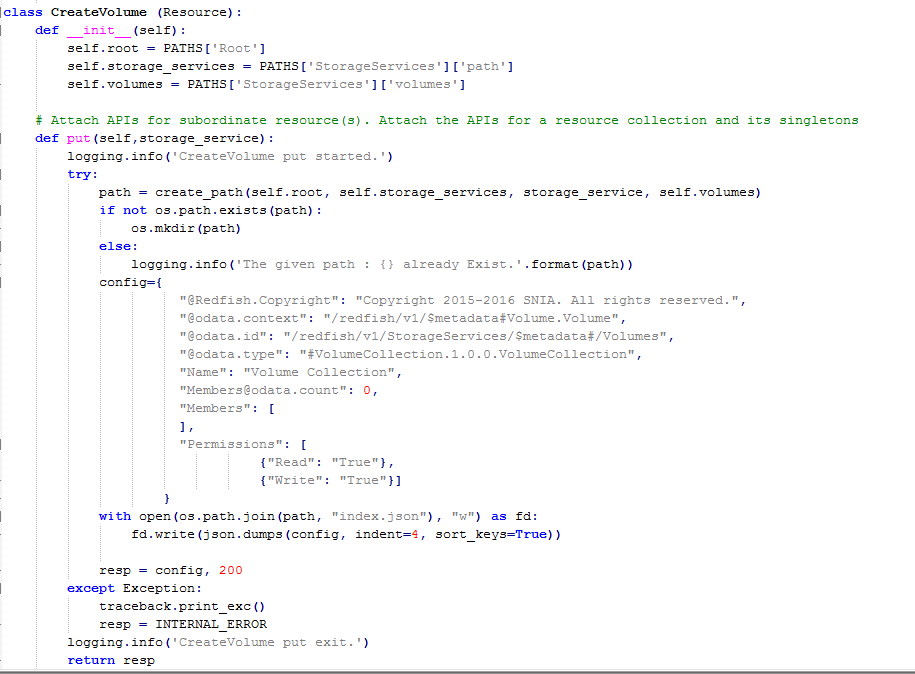
The above process is to create a new resource StorageServices. It stored in root directory.

Whenever we create a new resource , it will automatically creates a folder and subfolder.

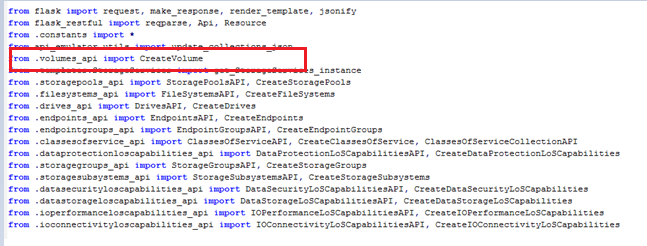
We are creating theses with the help of template files.

Example : we are creating volume as a sub-resource in storageServices.

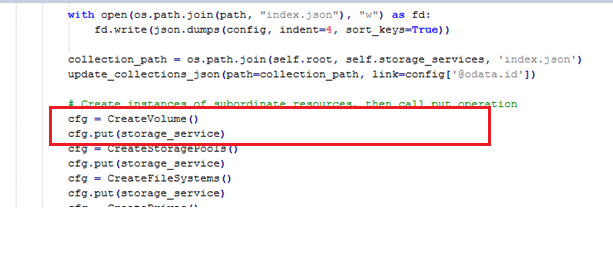
**Step1**:create a class” createvolume ()” in volume\_api.py file



**Step2**:- Importing volumes\_api file to storageservices\_api file



**Step3**:-create instance of volume and then call put operation.



So based on the above code StorageServices, a developer should create resources and sub-resources using template structure.