**SNIA CODE WALKTHROUGH**

We developed storage services module in SNIA SwordFish API Emulator

**Resource\_manager.py:**

**Step1:**

In resource manager class we have loaded dynamic resources from Storage\_Sevices.

imported classes which are written in storage services.py

**Step2:**

Loads Dynamic Resources (All api's and collections which are created in Storage\_Services.py)

Attach URI's for Subordinate Resources.

**g.api.add\_resource(StorageServicesCollectionAPI, '/redfish/v1/StorageServices/')**

**g.api.add\_resource(StorageServicesAPI, '/redfish/v1/StorageServices/<string:storage\_service>')**

**Step3:**

Declare storage\_services in configuration properties.

**'StorageServices': {'@odata.id': self.rest\_base + 'StorageServices'}**

**Storage\_Services.py:**

* 1. Storage services API we have developed API'S and collections for classes based on the requirement.
  2. Declares the variables and set Odata.id Properties.
  3. HTTP GET method for all classes in Storage Services.
  4. Attach the API’s for Subordinate resources.
  5. Create an instance of the Subordinate resources

Steps for classes that are declared in storage\_services.py are described below…(both API’s & Collections)

**Steps**

**1.-->** **\_\_init\_\_ (Self):** Declare root path

In Python self represents instance of object . we are creating instance of class and calls the methods in the class.

**2.--> get ()** method:

* We declare path dynamically (for all routes and sub routes)
* Declares list of keys and key values which are specified in mocks.
* In those key values we declare type definitions for objects.
* Calls JSON files through Get method and read the files which contains Odata.id properties.
* We have to open the json file as

Storage\_services\_json=open(path)

* After opening specific path we have to read path as

data= json.load(Storage\_services\_json)

* now check the type definitions declared in key\_ value.

Example for the following steps are given below.



**System \_details.py:**

we have developed 2 dynamic classes just for reviewing the json .

**SystemDetailsAPI:**

* We retrieve details from importing psutil
* We retrieve 'total disk space', 'available space’, ‘used space ‘and 'percentage used' data for detailed view of Systems API.

**System MemoryDetailsAPI**:

* In this class we have taken a json file and retrieve specific data by reading json file.
* We are navigating the information from system variables