```
# Copyright 2016 IBM All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
# See the License for the specific language governing permissions and
# limitations under the License.
"""
The v1 Discovery Service
(http://www.ibm.com/watson/developercloud/doc/discovery/)
"""
import json
import mimetypes

from .watson_developer_cloud_service import
WatsonDeveloperCloudService

default_url = 'https://gateway.watsonplatform.net/discovery/api'
latest_version = '2016-11-07'


[docs]class DiscoveryV1(WatsonDeveloperCloudService):
    """Client for Discovery service"""

    def __init__(self, version, url=default_url, username=None,
password=None,
                 use_vcap_services=True):
        """
        Construct an instance. Fetches service parameters from
VCAP_SERVICES
        runtime variable for Bluemix, or it defaults to local URLs.
        :param version: specifies the specific version-date of the
service to
        use
        """

        WatsonDeveloperCloudService.__init__(
            self, 'discovery', url, username, password,
use_vcap_services)
        self.version = version

[docs]    def get_environments(self):
        """
        Retrieves information about the environments associated with
the user
        """
        return self.request(method='GET', url='/v1/environments',
                            params={"version": self.version},
accept_json=True)


[docs]    def get_environment(self, environment_id):
        """
        Retrieves information about the specific
```

```
        :param environment_id:
        :return:
        """

        return self.request(method='GET',
url='/v1/environments/{0}'.format(environment_id),
                            params={"version": self.version},
accept_json=True)


    def _valid_name_and_description(self, name, description):
        if len(name) not in range(0, 255):
            raise ValueError(
                "name must be a string having length between 0 and 255
"
                "characters")
        if len(description) not in range(0, 255):
            raise ValueError(
                "description must be a string having length between 0
and 255 "
                "characters")

[docs]    def create_environment(self, name="", description="",
size=1):
        """

        :param name: name of the environment (max 255 chars) can be
empty
        :param description: description of the environment (max 255
chars)
        can be empty
        :param size: size of the environment (1,2, or 3)
        :return:
        """
        self._valid_name_and_description(name=name,
description=description)
        if size not in range(1, 4):
            raise ValueError("Size can be 1, 2, or 3")

        body = json.dumps({"name": name,
                           "description": description,
                           "size": size})
        return self.request(method='POST',
                            url='/v1/environments',
                            params={"version": self.version},
                            data=body,
                            headers={'content-type':
'application/json'},
                            accept_json=True)


[docs]    def update_environment(self, environment_id, name="",
description=""):
        """

        :param environment_id: guid of the environment to modify
        :param name: update the name of the environment
        :param description: update the description of the environment
        :return:
        """
```

```
        self._valid_name_and_description(name=name,
description=description)
        body = json.dumps({"name": name, "description": description})
        return self.request(method='PUT',

url='/v1/environments/{0}'.format(environment_id),
                            params={"version": self.version},
                            data=body,
                            headers={'content-type':
'application/json'},
                            accept_json=True)


[docs]    def delete_environment(self, environment_id):
        """
        Deletes the specified environment.
        :param environment_id: guid of environment to delete
        :return:
        """
        url_string = '/v1/environments/{0}'.format(environment_id)
        return self.request(method='DELETE', url=url_string,
                            params={"version": self.version},
                            accept_json=True)


[docs]    def list_configurations(self, environment_id):

        format_string = '/v1/environments/{0}/configurations'
        url_string = format_string.format(environment_id)
        return self.request(method='GET', url=url_string,
                            params={'version': self.version},
                            accept_json=True)


[docs]    def get_default_configuration_id(self, environment_id):
        all_configs =
self.list_configurations(environment_id=environment_id)
        try:
            configs = [x['configuration_id']
                       for x in all_configs['configurations']
                       if x['name'] == 'Default Configuration']
            if len(configs) > 0:
                return configs[0]
            else:
                return None
        except KeyError:
            pass # this isn't a problem and supress isn't in 2.7
        return None


[docs]    def get_configuration(self, environment_id,
configuration_id):

        format_string = '/v1/environments/{0}/configurations/{1}'
        url_string = format_string.format(environment_id,
configuration_id)
        return self.request(method='GET', url=url_string,
                            params={'version': self.version},
                            accept_json=True)
```

```
[docs]    def list_collections(self, environment_id):
        """
        Retrieves information about the collections within a given
environment
        :param environment_id: this is the guid of a valid environment
        :return: json results of the collections in an environment
        """
        url_string = '/v1/environments/{0}/collections'.format(
            environment_id)
        return self.request(method='GET', url=url_string,
                            params={"version": self.version},
                            accept_json=True)
```