Ckan API 说明文档

通过 **Ckan API** 可以实现网页 UI 的全部功能。举个简单的例子，在网页中点击“增加数据集”可以新建数据集，使用“dataset_create()”这一 API 函数也可以完成同样功能。

简介

Ckan API 为 RPC 风格，主要分为 *create*、*delete*、*update* 和 *get* 四类，分别对应 SQL 中的“增删改查”。

使用 **Ckan API** 的途径有两种：

- 通过 web 使用；
- 通过 python 代码使用。

通过 web

在浏览器中输入以下链接将会以 web 方式调用 **Ckan API**：
http://202.121.178.242/api/3/action/action_name
请求结果将以 JSON 格式返回，以下给出几个例子。

- 列出所有数据集（仅返回公开数据集）：http://202.121.178.242/api/3/action/package_list
- 查看某一数据集的详细信息（在可以访问的前提下）：
  http://202.121.178.242/api/3/action/package_show?id=public-data
- 查询符合条件的数据集（仅返回公开数据集）：
  http://202.121.178.242/api/3/action/package_search?q=public

使用某些 **Ckan API** 的时候还需要提供 API Key（可以在个人主页中查询自己的 API Key），从而保证该操作是符合权限控制的。

考虑到受网页链接长度的限制，通过 web 一般只使用较简单的 **Ckan API**，对于需要设置多个参数的 **Ckan API**，更恰当的方式是通过 python 代码使用。

通过 python 代码

在 python 代码中，我们可以为要调用的 **Ckan API** 更完善地赋予需要的参数。话不多说，以之前提到的 *package_list* 为例，示范一下如何通过 python 代码调用 **Ckan API**，以下代码查询名字符合要求的数据集并列出查询结果。

```
#!/usr/bin/env python
import urllib2
import urllib
import json
import pprint
# Use the json module to dump a dictionary to a string for posting.
data_string = urllib.quote(json.dumps({'id': 'public-data'}))
```

```
# Make the HTTP request.
response = urllib2.urlopen('http://202.121.178.242/api/3/action/package_list', data_string)
assert response.code == 200

# Use the json module to load CKAN's response into a dictionary.
response_dict = json.loads(response.read())

# Check the contents of the response.
assert response_dict['success'] is True
result = response_dict['result']
pprint.pprint(result)
```

可以总结出 python 代码调用 **Ckan API** 的几个步骤：

- 定义 data_string 变量，该变量存储 API 函数需要的参数；
- 调用 urlopen()函数，该函数第一个参数和通过 web 使用 **Ckan API** 时的链接类似，第二个参数即为 data_dict；
- 调用 json.load()函数，该函数从 response 中读取内容，返回 dictionary 变量；
- 判断是否成功执行，如果成功则提取出结果并打印出来以供调试。

再给出一个涉及 API Key 的例子，以下代码使用 package_create 来新建数据集。由于 Ckan 规定未登陆用户不得新建数据集，因此新建数据集这一操作需要验证用户身份，即用户的 API Key。

```
#!/usr/bin/env python
import urllib2
import urllib
import json
import pprint

# Put the details of the dataset we're going to create into a dict.
dataset_dict = {
    'name': 'my_dataset_name',
    'notes': 'A long description of my dataset',
}

# Use the json module to dump the dictionary to a string for posting.
data_string = urllib.quote(json.dumps(dataset_dict))

# We'll use the package_create function to create a new dataset.
request = urllib2.Request('http:// 202.121.178.242/api/3/action/package_create')

# Creating a dataset requires an authorization header.
# Replace *** with your API key, from your user account on the CKAN site
# that you're creating the dataset on.
request.add_header('Authorization', '***')
```

```
# Make the HTTP request.
response = urllib2.urlopen(request, data_string)
assert response.code == 200

# Use the json module to load CKAN's response into a dictionary.
response_dict = json.loads(response.read())
assert response_dict['success'] is True

# package_create returns the created package as its result.
created_package = response_dict['result']
pprint.pprint(created_package)
```

步骤是类似的，但是此处由于操作为新建数据集，所以 data_string 涉及的参数相对多一些。同时在调用 urlopen()之前需要添加 API Key，使用 add_header()函数完成，之后请求数据、验证是否成功、返回数据即可。

至于其他 **Ckan API** 的使用方法，步骤和格式也都是完全相同，只需要修改 data_dict 内容即可，具体设置请参考 **Ckan API** 的 **API Reference**。

http://docs.ckan.org/en/ckan-2.2/api.html#action-api-reference

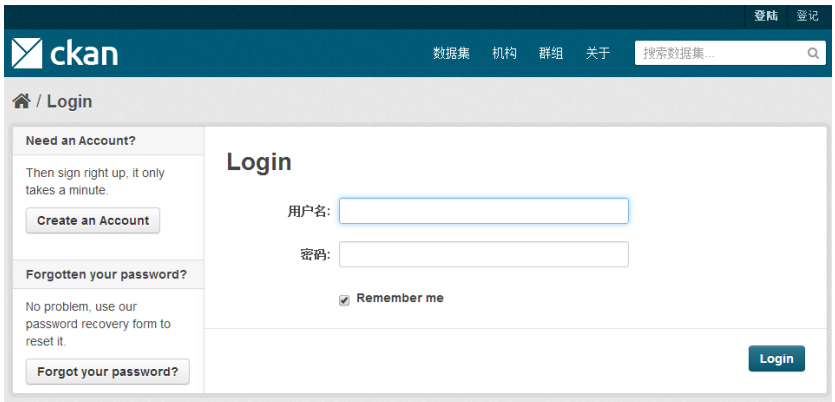
如何使用 Ckan API 上传数据

现在对给出的三个 python 文件进行说明。

- common.py 里面包含 fun_exe()、datastore_create()、url_update()和 datastore_upsert()四个函数，以及 resource_id 和 api_key 两个变量。
- setup.py 调用 datastore_create()和 url_update()，完成初始化工作。
- upload_data()调用 datastore_upsert()，完成数据上传工作。

具体来说，需要按照以下步骤上传数据：

1. 登陆 Ckan 网站并验证身份，http://202.121.178.242/；



2. 点击“增加数据集”以新建数据集；

3. 填写该数据集的基本信息(metadata)，包括标题、描述、标签和授权等基本内容。如果这个数据集不属于任何组织，在“组织”一栏选择“没有组织”并将可见性设为“公开”，否则应选择相应组织并确定数据集可见性。点击“下一步”；

4. 为数据集添加数据(resource)，网址暂时随便填一个非空值(由于我们的数据是由本地上传的，所以网上当然找不到一个指向该数据的链接)，在后面将会修改；填写名称和描述，格式填写为“json”。点击“下一步”，或者“保存并添加另一个”以添加另一项数据；

5. 添加一些额外信息，如数据来源、版本号、作者和作者邮箱等，但都是可选的，所以也可以直接点击“完成”。

6. 现在便可以看到刚才创建的数据集和数据，点击该数据并查看其 id，这个 id 唯一标识该数据；

其他信息

域	价值
最后更新	九月 6, 2014
创建的	九月 6, 2014
格式	JSON
授权	没指定授权类型
can be previewed	1
created	5 分钟前
format	JSON
id	bad086c5-23e7-4872-a8be-30ef40d1cbdd
resource group id	28019196-7422-428f-97a4-4f0ad79e5ddb
revision id	53332f4e-66ea-441d-bec7-3b71473bfbfe
state	active

隐藏

7. 接下来便可以向该数据上传数据项。修改 common.py 中的 resource_id 为以上的 id，修改 api_key 为你的用户 API Key(在个人中心的左下角可以查看)；

8. 在 common.py 中，修改 datastore_create()函数 resource_dict 变量“fields”值，使之符合你期望的 json 格式；

```
def datastore_create(resource_id,api_key):
    # create a datastore for a resource
    resource_dict = {
        "resource_id": resource_id,
        "force": True,
        "fields": [{"id": "author", "type": "text"}, {"id": "submit
        # 'resource': resource,
        # 'aliases': ['author', 'submitted_on', 'PM2_5', 'CO', 'PM1
        # 'records': records
        # 'primary_key': ['_id'],
        # 'indexes': ['id'],
    }
```

9. 运行 setup.py 文件，之后可以发现网站上的数据已经有了数据格式，并且 url 已经更新

并可下载；

10. 接下来即可按照 upload_data.py 给出的例子上传数据。在 upload_data.py 中只是简单地定义了一个符合 json 格式的 data 变量，并且调用 datastore_upsert()函数完成上传，实际应用中应当从真实数据源获得实时数据并上传。