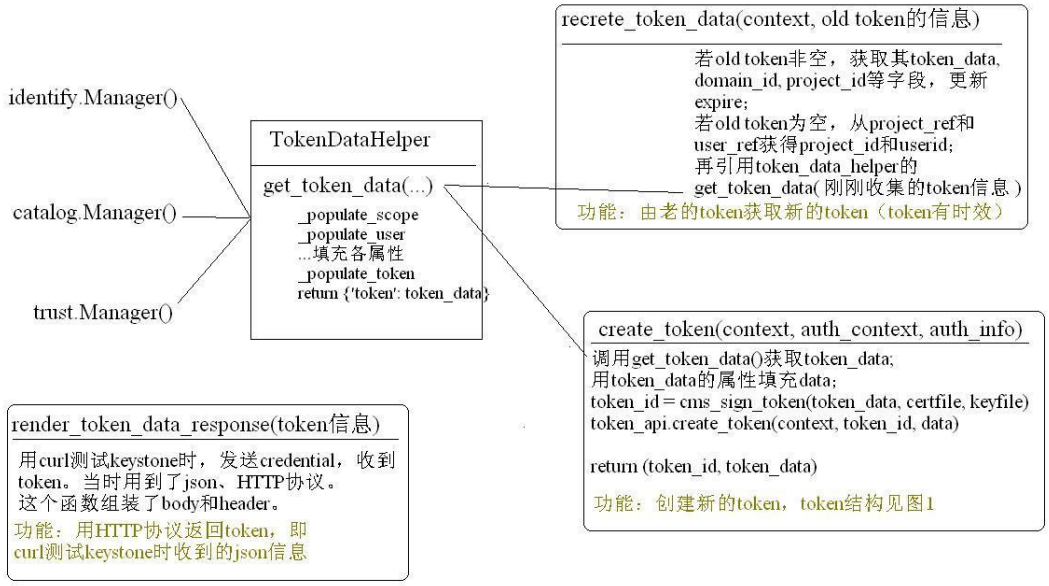
1 源码位置：keystone/auth/token_factory.py

2 token 包含的属性

```
#   id = some_id
#   user_id = some_user_id
#   expires = some_expiration
#   extras = {'user': {'id': some_used_id},
#             'tenant': {'id': some_tenant_id},
#             'token_data': 'token': {'domain': {'id': some_domain_id},
#                                     'project': {'id': some_project_id},
#                                     'domain': {'id': some_domain_id},
#                                     'user': {'id': some_user_id},
#                                     'roles': [{'id': some_role_id}, ...],
#                                     'catalog': ...,
#                                     'expires_at': some_expiry_time,
#                                     'issued_at': now(),
#                                     'methods': ['password', 'token'],
#                                     'extras': { ... empty? ...}
```

3 文件包含的函数以及功能，结构图。



1 源码位置：keystone/auth/routers.py

2 源码截图和说明：

```
def append_v3_routers(mapper, routers):
    auth_controller = controllers.Auth()

    mapper.connect('/auth/tokens',
                   controller=auth_controller,
                   action='authenticate_for_token',
                   conditions=dict(method=['POST']))
    mapper.connect('/auth/tokens',
                   controller=auth_controller,
                   action='check_token',
                   conditions=dict(method=['HEAD']))
    mapper.connect('/auth/tokens',
                   controller=auth_controller,
                   action='revoke_token',
                   conditions=dict(method=['DELETE']))
    mapper.connect('/auth/tokens',
                   controller=auth_controller,
                   action='validate_token',
                   conditions=dict(method=['GET']))
    mapper.connect('/auth/tokens/OS-PKI/revoked',
                   controller=auth_controller,
                   action='revocation_list',
                   conditions=dict(method=['GET']))
```

个人理解：用 curl 测试 keystone 时要写信息发送的 URI，这里可能是把服务地址、提供服务的函数、action 和使用的 HTTP 方法（例如 GET 和 POST）绑定一起。参考 HTTP 工作原理。

1 源码位置：keystone/auth/core.py
2 功能
纯注释文档，说明 authenticate 的格式、过程和返回结果。
3 原文翻译
def authenticate(self, context, auth_payload, auth_context):
认证用户，返回认证上下文
:param context: keystone 的请求的内容
:auth_payload: 既定函数的认证内容（应该是指 authenticate 函数将要验证的内容）
:auth_context: 用户认证上下文，所有 plugin（稍后会提，这个 plugin 提供具体的认证操作）共享的字典类。这个 auth_context 默认包含 method_names 和 extras 变量，其中 method_name 是一个链表，extras 是一个字典类。

如果认证成功，plugin 必须必须在 auth_context 中设置 user_id 属性。如果认证是为了 re-scoping（俺不懂嘛叫 re-scoping）的话，method_name 链表就用来装载附加的认证函数。

举例来说，如果认证是为了 re-scoping, plugin 必须在 mehod_name 链表的默认加上之前的函数名。同时，plugin 也可能加附加信息到 extras 字典中。任何 extras 中的内容都会放在 token 的 extras 区域中。以下是一个 auth_context 认证成功的例子。

```
{"user_id": "abc123",
 "methods": ["password", "token"],
```

```
"extras": {}}
```

Plugin 按照 identity 对象中的 methods 属性中的顺序依次调用。
例如，对于以下认证请求：

```
{"auth": {
    "identity": {
        "methods": ["custom-plugin", "password", "token"],
        "token": {
            "id": "sdfafasdfsfasfasdfds"
        },
        "custom-plugin": {
            "custom-data": "sdfdfsfsfsdfsf"
        },
        "password": {
            "user": {
                "id": "s23sfad1",
                "password": "secrete"
            }
        }
    }
}}
```

调用顺序为：
Custom-plugin, password, token（两个 plugin 在 plugins 文件夹里面有，提供了具体的认证操作，稍后分析）
: returns：如果认证成功，返回 None。如果用了多步认证，authentication payload（即将要认证的内容）以字典（dictionary 是 python 语言的内建对象）的格式储存，交给下一步认证。
: raises：如果认证失败，抛出 exception.Unauthorized。