核心模型设计

创建分支

在猛犸平台集成CMDB子系统

```
$ git checkout -b cmdb
$ python manage.py startapp cmdb
```

创建应用cmdb，为cmdb包新建urls.py

路由配置

主路由mammoth/urls.py中增加到cmdb的路由项

```
urlpatterns = [
    # path('admin/', admin.site.urls),
    path('login/', tobview, name='login'),
    path('token/', tobview, name='token_obtain_pair'),  # 获取
    path('token/refresh/', TokenRefreshView.as_view(), name='token_refresh'),
# 刷新
    path('users/', include('user.urls')), # 二级路由到user/
    path('cmdb/', include('cmdb.urls')), # 二级路由到cmdb/
]
```

新建cmdb/urls.py

```
from django.urls import path

urlpatterns = []
```

MongoDB支持

Mongoengine实现ODM。

DRFM(Django Rest Framework Mongoengine)，与DRF集成，提供

- 序列化器
- 视图、视图集
- 路由器

https://github.com/umutbozkurt/django-rest-framework-mongoengine/blob/master/README.md

依赖`Django 2.x或3.x，DRF 3.x，mongoengine0.18*|0.19*`

```
$ pip install mongoengine
$ pip install django-rest-framework-mongoengine
```

注意：Django-MongoEngine不要使用，官方称是不稳定版本。

注册应用

```
INSTALLED_APPS = (
    ...
    'rest_framework',
    'rest_framework_mongoengine',
    'cmdb',
)
```

全局MongoDB配置：

项目需要使用MongoDB，这是自定义的配置项。主要提供给连接函数使用。

参考 mongoengine.connection.connect，其核心调用的 mongoengine.connection.register_connection

```
MONGODB_DATABASES = {
    'name': 'cmdb', # db也行
    "host": '192.168.142.130',
    "port": 27017,
    # "password": 'wayne',
    # "username": 'wayne',
    'tz_aware': True, # 如果Django中USE_TZ = True
}
```

启动加载

应用被加载后，就被执行？比较好的方法，把代码放到应用目录/apps.py的AppConfig子类中，定义ready方法

参考 https://docs.djangoproject.com/en/3.2/ref/applications/#django.apps.AppConfig.ready

cmdb/apps.py如下

```
from django.apps import AppConfig
from django.conf import settings
from mongoengine import connect

class CmdbConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'cmdb'

    def ready(self):
        print('~~cmdb项目加载，建立MongoDB连接~~')
        # 建立别名default的MongoDB连接
        connect(**settings.MONGODB_DATABASES)
```

CIType设计

CMDB**最难的设计**就是如何将千万种资产数据统一存储管理，还可以扩展类型和类型属性。

每一种类型一个集合，这样和关系数据库表设计一样，类型多则对应的表多，一种类型一张表，100种类型100个集合或表，显然不合适，这就要充分的利用MongoDB的优势。

把所有类型放在一个集合中，可以吗？可以，MongoDB允许集合中字段不一样。那怎么知道，类型A有多少个字段呢？类型B有多少个呢？字段类型是什么呢？动态的字段最终还要动态的显示在浏览器上。

在设计时，还要考虑资产之间的所属关系。例如服务器有硬盘、网络接口。硬盘、网络接口都有独立的资产属性字段。硬盘可以是独立的资产项，但是网络接口不是独立的资产项，必须包含在其它资产中。

Server服务器常见字段，如下

```
Name资产名
Organization所属组织
Location位置
Asset number资产编号

设备管理信息
Brand品牌、Model型号，复杂点可以变成级联选择
OS Family操作系统、OS version版本，也可以是级联
Management IP 手动填写
CPU  多个
RAM  内存，多个
Rack 机架
Serial number序列号

Production Date上线时间、Purchase date购买日期、End of warranty保修期结束
description多行的

Applications 部署的应用，应用是Application类型的实例
network interfaces 网络接口，网络接口时Network Interface类型的实例
...... 太多了
```

上面简单列出了服务器的一些字段，如有需要，自行扩展。

Network Interface 必须从属于某个表，比如说，它是服务器的一部分，一台服务器有N个网络接口，这是一对多关系。它有如下一些字段

```
Name设备名称，IP Address、Mac Address、GateWay、Mask
```

类型数据必须有类型id和Version。有了新版老版本直接失效，只用来显示旧数据。

要设计一个前端可视化界面完成类型构建，由于时间关系不完成了。手动完成类型元数据设计

```
{
    "id": "xxxxxxxxxxxxxxxx",
    "name": "Network Interface",
    "label": "网络接口",
    "version": 1,
    "fields": [
        {"name": "name", "label": "名称", "type": "str", "required": true},
        {"name": "IP Address", "label": "IP", "type": "str", "required": true},
        {"name": "Mac Address", "label": "MAC", "type": "str", "required": true},
        {"name": "GateWay", "label": "网关", "type": "str", "required": false},
        {"name": "Mask", "label": "掩码", "type": "str", "required": false}
    ]
}
```

从上例的设计可以看出，分为类型信息和字段信息

- 类型属性：id、name、label、version、fields。字段们是一个列表，可以有多个字段定义
- 字段属性：id、name、label、type、required
    - type，指定字段类型，字符串str、整数int、浮点数float等

name是内部使用，label用于显示前端网页上。

参照上面的设计，使用ODM，完成模型类

cmdb/models.py

```
from mongoengine import (
    Document, EmbeddedDocument,
    StringField, IntField, BooleanField,
    ListField, EmbeddedDocumentField
)

class CiTypeField(EmbeddedDocument):
    meta = {'collection': 'citypes'}
    name = StringField(required=True, max_length=24)
    label = StringField(max_length=24)
    type = StringField(max_length=24)
    required = BooleanField(default=False)

    def __str__(self):
        return "<F {},{}>".format(self.name, self.type)

class CiType(Document):
    meta = {'collection': 'citypes'}
    name = StringField(required=True, unique_with='version', max_length=24)
    label = StringField(max_length=24)
    version = IntField(required=True, default=1)
    fields = ListField(EmbeddedDocumentField(CiTypeField))

    def __str__(self):
        return "<CiType {}:{}, {}>".format(
            self.name, self.version, self.fields)
```

网络接口类

```
from mongoengine import connect

MONGODB_DATABASES = {
    'name': 'cmdbtest', # db也行
    'host': '127.0.0.1',
    'port': 27017,
    # 'username': 'wayne',
    # 'password': 'magedu',
    'tz_aware': True, # 如果Django中USE_TZ = True
}
connect(**MONGODB_DATABASES)

#### 以上代码测试用 ####

from cmdb.models import CiType, CiTypeField

ct = CiType()
ct.name = 'Network Interface'
ct.label = '网络接口'
ct.fields = [
    CiTypeField(name='name', label='名称', type='str', required=True),
    CiTypeField(name='IP Address', label='IP', type='str', required=True),
    CiTypeField(name='Mac Address', label='MAC', type='str', required=True),
    CiTypeField(name='GateWay', label='网关', type='str'),
    CiTypeField(name='Mask', label='掩码', type='str'),
]

ct.save()
print(ct)
```

在测试过程中，改变index后，有可能出现`pymongo.errors.DuplicateKeyError: E11000 duplicate key error collection`的异常，就是collection中已经有了之前定义的index影响当下，删除index或者collection就行了，重建即可。

```
from cmdb.models import CiType, CiTypeField

for x in CiType.objects:
    print(type(x), x)
    print(x.to_json())
```

```
<class 'cmdb.models.CiType'> <CiType Network Interface:1, [<CiTypeField: <F name,str>>, <CiTypeField: <F IP Address,str>>, <CiTypeField: <F Mac Address,str>>, <CiTypeField: <F GateWay,str>>, <CiTypeField: <F Mask,str>>]>

{"_id": {"$oid": "61813b5bc162177d1110949f"}, "name": "Network Interface", "label": "\u7f51\u7edc\u63a5\u53e3", "version": 1, "fields": [{"name": "name", "label": "\u540d\u79f0", "type": "str", "required": true}, {"name": "IP Address", "label": "IP", "type": "str", "required": true}, {"name": "Mac Address", "label": "MAC", "type": "str", "required": true}, {"name": "GateWay", "label": "\u7f51\u5173", "type": "str", "required": false}, {"name": "Mask", "label": "\u63a9\u7801", "type": "str", "required": false}]}
```

服务器类

```
from mongoengine import connect

MONGODB_DATABASES = {
    'name': 'cmdbtest', # db也行
    'host': '127.0.0.1',
    'port': 27017,
    # 'username': 'wayne',
    # 'password': 'magedu',
    'tz_aware': True, # 如果Django中USE_TZ = True
}
connect(**MONGODB_DATABASES)

#### 以上代码测试用 ####

from cmdb.models import CiType, CiTypeField


ct = CiType()
ct.name = 'Server'
ct.label = '服务器'
ct.version = 1
# 服务器属性字段太多，省略一些，完成核心功能
ct.fields = [
    CiTypeField(name='name', label='资产名称', type='str', required=True),
    CiTypeField(name='Asset number', label='资产编号', type='str'),
    CiTypeField(name='Brand', label='品牌', type='str'),
    CiTypeField(name='Model', label='型号', type='str'),
    CiTypeField(name='OS Family', label='操作系统', type='str'),
    CiTypeField(name='OS Version', label='OS版本', type='str'),
    CiTypeField(name='Management IP', label='管理IP', type='str'),
    CiTypeField(name='CPU', label='CPU', type='str'),
    CiTypeField(name='RAM', label='内存', type='str'),
    CiTypeField(name='Rack', label='机架', type='str'),
    CiTypeField(name='Production Date', label='上线时间', type='date'),
    CiTypeField(name='Purchase date', label='购买日期', type='date'),
    CiTypeField(name='End of warranty', label='保修期结束', type='date'),
    # 可以有N个网络接口，一对多
    CiTypeField(name='Network Interface', label='网络接口',
type='list:Network Interface'),
]
ct.save()
print(ct.to_json())
```