核心功能开发

资产是资产类型的实例，资产类型中定义了该类型的应该具有哪些字段。资产类型和资产之间形成了一对多关系。

CIType功能

可以对资产类型进行增删改查，但是要注意，删除和修改对该类型已有资产实例的影响。

后台代码

user/views.py 菜单项

```python
from django.contrib.auth.models import AbstractUser

@api_view(['GET'])
@permission_classes([IsAuthenticated]) # 覆盖全局配置单独设置权限
def menulist_view(request:Request):
    user:AbstractUser = request.user
    auth = request.auth
    print(user, auth)
    # print(dir(user)) # .is_superuser
    menulist = {
        'data': []
    }
    if user.is_superuser: # 用户管理必须管理员
        p1 = PermissionItem(101, '用户管理')
        p1.append(PermissionItem(102, '用户列表', '/users/'))
        p1.append(PermissionItem(103, '角色列表', '/users/roles/'))
        p1.append(PermissionItem(104, '权限列表', '/users/perms/'))
        menulist['data'].append(p1)
    # if user.has_perm('user.view_userprofile'):
    #     pass

    # cmdb
    p5 = PermissionItem(501, '资产管理')
    p5.append(PermissionItem(502, 'CI类型列表', '/cmdb/citypes/'))
    p5.append(PermissionItem(503, 'CI列表', '/cmdb/cis/'))
    menulist['data'].append(p5)

    return Response(menulist)
```

马哥教育 IT人的高薪职业学院

cmdb/models.py

```python
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)
```

创建cmdb/serializers.py

```python
from rest_framework_mongoengine import serializers
from .models import CiType

# 使用DRFMongoEngine的序列化器类
class CiTypeSerializer(serializers.DocumentSerializer):
    class Meta:
        model = CiType
        #fields = ('id', 'name', 'label','version')
        exclude = ['fields']
```

马哥教育 IT人的高薪职业学院

cmdb/views.py

```python
from rest_framework_mongoengine.viewsets import ModelViewSet
from .models import CiType
from .serializers import CiTypeSerializer

class CiTypeViewSet(ModelViewSet):
    # CiType是ODM，因此使用DRFM的ModelViewSet
    queryset = CiType.objects.all()
    serializer_class = CiTypeSerializer
    permission_classes = [] # 移除所有权限要求
```

cmdb/urls.py

```
from django.urls import path
from .views import CiTypeViewSet
from rest_framework_mongoengine.routers import SimpleRouter

router = SimpleRouter()
router.register('citypes', CiTypeViewSet)

urlpatterns = [] + router.urls

print('~' * 30)
print(urlpatterns)
print('~' * 30)
```

前台代码

在组件目录下新建cmdb目录创建CiTypes.vue。

src/router/index.js，省略无关代码

```
import CiType from '../components/cmdb/CiTypes.vue'

const routes = [
  { path: '/', redirect: '/login' },
  { path: '/login', component: Login },
  {
    path: '/home',
    component: Home,
    redirect: '/welcome',
    children: [
      { path: '/welcome', component: Welcome },
      { path: '/users', component: User },
      { path: '/users/perms', component: Perm },
      { path: '/users/roles', component: Role },
      { path: '/cmdb/citypes', component: CiType }
    ]
  }
]
```

马哥教育 IT人的高薪职业学院

新建src/components/cmdb/CiTypes.vue

```
<template>
  <div>
    <el-breadcrumb separator-class="el-icon-arrow-right">
      <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
      <el-breadcrumb-item>资产管理</el-breadcrumb-item>
      <el-breadcrumb-item>资产类型</el-breadcrumb-item>
    </el-breadcrumb>
    <el-card class="box-card">
      <el-table :data="dataList" border style="width: 100%">
        <el-table-column type="index" label="序号"> </el-table-column>
        <el-table-column prop="label" label="名称"> </el-table-column>
        <el-table-column label="操作">
          <el-tooltip content="分配权限" effect="light">
            <el-button type="info" icon="el-icon-plus" size="mini"></el-button>
          </el-tooltip>
          <el-tooltip content="修改" effect="light">
            <el-button type="success" icon="el-icon-edit" size="mini"></el-button>
          </el-tooltip>
          <el-tooltip content="删除" effect="light">
            <el-button type="danger" icon="el-icon-delete" size="mini"></el-button>
          </el-tooltip>
        </el-table-column>
      </el-table>
      <el-pagination
        @current-change="handleCurrentChange"
        :current-page="pagination.page"
        :page-size="pagination.size"
        layout="total, prev, pager, next, jumper"
        :total="pagination.total"
      >
      </el-pagination>
    </el-card>
  </div>
</template>

<script>
export default {
  created() {
    this.getList()
  },
  data() {
    return {
      dataList: [],
      pagination: { page: 1, size: 1, total: 0 }
    }
  },
  methods: {
    handleCurrentChange(val) {
      console.log(`当前页: ${val}`)
      this.getList(val)
    },
    async getList(page = 1) {
      if (!page) {
        page = 1
      }
      const { data: response } = await this.$http.get('cmdb/citypes/', {
        params: {
          page
        }
      })
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.dataList = response.results
      this.pagination = response.pagination
    }
```

马哥教育 IT人的高薪职业学院

```
  }
}
</script>

<style>
</style>
```

增删改功能，请参照之前代码，自行补全。

CI功能

对资产的增删改查，资产属于某一个类型。

后端代码

由于资产类别有很多种，每一种的字段也都不一样，产生的资产实例字段完全不一样。

需要为每一种类别建一个Collection吗？

不需要。将所有资产都放到一个Collection中，命名为cis。cis的字段完全不固定，怎么设计模型类呢？

http://docs.mongoengine.org/guide/defining-documents.html#dynamic-document-schemas

cmdb/models.py

```python
from mongoengine import (
    Document, EmbeddedDocument, DynamicDocument,
    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)

class Ci(DynamicDocument):
    # 字段取决于CiType中某一种类型在fields中定义的字段，可以说几乎每种类型的字段都不一样
    meta = {'collection': 'cis'}
```

马哥教育 IT人的高薪职业学院

cmdb/serializers.py

```python
from rest_framework_mongoengine import serializers
from .models import CiType, Ci

# 使用DRFMongoEngine的序列化器类
class CiTypeSerializer(serializers.DocumentSerializer):
    class Meta:
        model = CiType
        #fields = ('id', 'name', 'label', 'version')
        exclude = ['fields']

class CiSerializer(serializers.DynamicDocumentSerializer):
    class Meta:
        model = Ci
        fields = '__all__'
```

cmdb/views.py

增加一个不分页的返回全部CI分类的功能

```python
from rest_framework.response import Response
from rest_framework_mongoengine.viewsets import ModelViewSet
from rest_framework.decorators import action
from .models import CiType, Ci
from .serializers import CiTypeSerializer, CiSerializer

class CiTypeViewSet(ModelViewSet):
    # CiType是ODM，因此使用DRFM的ModelViewSet
    queryset = CiType.objects.all()
    serializer_class = CiTypeSerializer
    permission_classes = [] # 移除所有权限要求

    @action(detail=False) # 默认GET
    def all(self, request):
        """返回所有分类。相当于重写list，去掉分页"""
        queryset = self.filter_queryset(self.get_queryset())
        serializer = self.get_serializer(queryset, many=True)
        data = serializer.data
        return Response(data)

class CiViewSet(ModelViewSet):
    queryset = Ci.objects
    serializer_class = CiSerializer
    permission_classes = [] # 移除所有权限要求
```

cmdb/urls.py

```python
from .views import CiTypeViewSet, CiViewSet
from rest_framework_mongoengine.routers import SimpleRouter

router = SimpleRouter()
router.register('citypes', CiTypeViewSet)
router.register('cis', CiViewSet)

urlpatterns = [] + router.urls
```

动态改变字段

需求：列表页和详情页看到的字段不一样。

解决方案：

1. 不用视图集，定义列表页、详情页2个视图类，各自指定不同的序列化器
2. 使用视图集，内部区分列表页、详情页，来使用不同的序列化器

cmdb/serializers.py

```python
from rest_framework_mongoengine import serializers
from .models import CiType, Ci

# 使用DRFMongoEngine的序列化器类
class CiTypeSerializer(serializers.DocumentSerializer):
    class Meta:
        model = CiType
        #fields = ('id', 'name', 'label', 'version')
        exclude = ['fields']

class CiTypeFieldsSerializer(serializers.DocumentSerializer):
    class Meta:
        model = CiType
        fields = ('id', 'fields')

class CiSerializer(serializers.DynamicDocumentSerializer):
    class Meta:
        model = Ci
        fields = '__all__'
```

马哥教育 IT人的高薪职业学院

访问某一个CiType的时候，也就是详情页，需要返回所有字段定义，这和列表页不一样。

如何区分列表页、详情页？用是否有pk或id参数。

如何更换序列化器？这就要覆盖get_serializer_class了。

```python
from rest_framework.response import Response
from rest_framework_mongoengine.viewsets import ModelViewSet
from rest_framework.decorators import action
from .models import CiType, Ci
from .serializers import CiTypeSerializer, CiSerializer,
CiTypeFieldsSerializer

class CiTypeViewSet(ModelViewSet):
    # CiType是ODM，因此使用DRFM的ModelViewSet
    queryset = CiType.objects.all()
    serializer_class = CiTypeSerializer
    permission_classes = [] # 移除所有权限要求

    @action(detail=False) # 默认GET
    def all(self, request):
        """返回所有分类。相当于重写list，去掉分页"""
        queryset = self.filter_queryset(self.get_queryset())
        serializer = self.get_serializer(queryset, many=True)
        data = serializer.data
        return Response(data)

    def get_serializer_class(self):
        if 'id' in self.kwargs: # 有id就是详情页替换序列化器
            return CiTypeFieldsSerializer
        return super().get_serializer_class()


class CiViewSet(ModelViewSet):
    queryset = Ci.objects
    serializer_class = CiSerializer
    permission_classes = [] # 移除所有权限要求
```

前端代码

新建Cis.vue组件文件。

src/router/index.js 路由配置

```javascript
import CiType from '../components/cmdb/CiTypes.vue'
import Ci from '../components/cmdb/Cis.vue'

const routes = [
  { path: '/', redirect: '/login' },
  { path: '/login', component: Login },
  {
    path: '/home',
    component: Home,
    redirect: '/welcome',
    children: [
      { path: '/welcome', component: Welcome },
      { path: '/users', component: User },
      { path: '/users/perms', component: Perm },
      { path: '/users/roles', component: Role },
      { path: '/cmdb/citypes', component: CiType },
      { path: '/cmdb/cis', component: Ci }
    ]
  }
]
```

马哥教育 IT人的高薪职业学院

src/plugins/element.js

```javascript
import Vue from 'vue'
import {
  // 省略重复代码
  Select,
  Option,
  DatePicker
} from 'element-ui'

Vue.use(Select)
Vue.use(Option)
Vue.use(DatePicker)
```

Select选择框、DatePicker日期

src/components/cmdb/Cis.vue

```
<template>
  <div>
    <el-breadcrumb separator-class="el-icon-arrow-right">
      <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
      <el-breadcrumb-item>资产管理</el-breadcrumb-item>
      <el-breadcrumb-item>资产列表</el-breadcrumb-item>
    </el-breadcrumb>
    <el-card class="box-card">
      <el-row :gutter="20">
        <el-col :span="12">
          <el-button type="primary" @click="addDialogVisible = true">增加资产</el-button>
        </el-col>
      </el-row>
      <el-table :data="dataList" border style="width: 100%">
        <el-table-column type="index" label="序号"> </el-table-column>
        <el-table-column prop="label" label="名称"> </el-table-column>
        <el-table-column label="操作">
          <el-tooltip content="分配权限" effect="light">
            <el-button type="info" icon="el-icon-plus" size="mini"></el-button>
          </el-tooltip>
          <el-tooltip content="修改" effect="light">
            <el-button type="success" icon="el-icon-edit" size="mini"></el-button>
          </el-tooltip>
          <el-tooltip content="删除" effect="light">
            <el-button type="danger" icon="el-icon-delete" size="mini"></el-button>
          </el-tooltip>
        </el-table-column>
      </el-table>
      <el-pagination
        @current-change="handleCurrentChange"
        :current-page="pagination.page"
        :page-size="pagination.size"
        layout="total, prev, pager, next, jumper"
        :total="pagination.total"
      >
      </el-pagination>
    </el-card>
    <!-- 添加对话框 -->
    <el-dialog title="增加" :visible.sync="addDialogVisible" @close="resetForm('add')">
      <el-form :model="addForm" :rules="addRules" ref="add" label-width="100px">
        <el-form-item label="名称">
          <el-select v-model="addForm.citype" placeholder="请选择" @change="handleCiTypeChange">
            <el-option v-for="item in citypes" :key="item.id" :label="item.label" :value="item.id"> </el-option>
          </el-select>
        </el-form-item>
      </el-form>
      <div slot="footer" class="dialog-footer">
```

```
        <el-button @click="addDialogVisible = false">取 消</el-button>
        <el-button type="primary" @click="add">确 定</el-button>
      </div>
    </el-dialog>
  </div>
</template>

<script>
export default {
  created() {
    this.getList()
    this.getAllTypeList()
  },
  data() {
    return {
      dataList: [],
      pagination: { page: 1, size: 1, total: 0 },
      citypes: [],
      //
      addDialogVisible: false,
      addForm: {
        citype: null
        // name: ''
      },
      addRules: {
        // name: [
        //   { required: true, message: '请输入名称', trigger: 'blur' },
        //   { min: 1, max: 16, message: '长度在 1 到 16 个字符', trigger: 'blur' }
        // ]
      }
    }
  },
  methods: {
    handleCurrentChange(val) {
      console.log(`当前页: ${val}`)
      this.getList(val)
    },
    resetForm(name) {
      this.$refs[name].resetFields()
    },
    async getList(page = 1) {
      if (!page) {
        page = 1
      }
      const { data: response } = await this.$http.get('cmdb/cis/', {
        params: {
          page
        }
      })
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.dataList = response.results
      this.pagination = response.pagination
    },
    async getAllTypeList(page = 1) {
      if (!page) {
```

马哥教育
IT人的高薪职业学院

```
    page = 1
   }
   const { data: response } = await this.$http.get('cmdb/citypes/all/')
   if (response.code) {
    return this.$message.error(response.message)
   }
   this.citypes = response
  },
  async handleCiTypeChange() {
   console.log(this.addForm)
   // 去请求类型的字段
   const { citype: id } = this.addForm
   const { data: response } = await this.$http.get(`cmdb/citypes/${id}/`)
   if (response.code) {
    return this.$message.error(response.message)
   }
   console.log(response) // 返回所有定义的字段
  },
  add() {}
 }
}
</script>

<style>
</style>
```

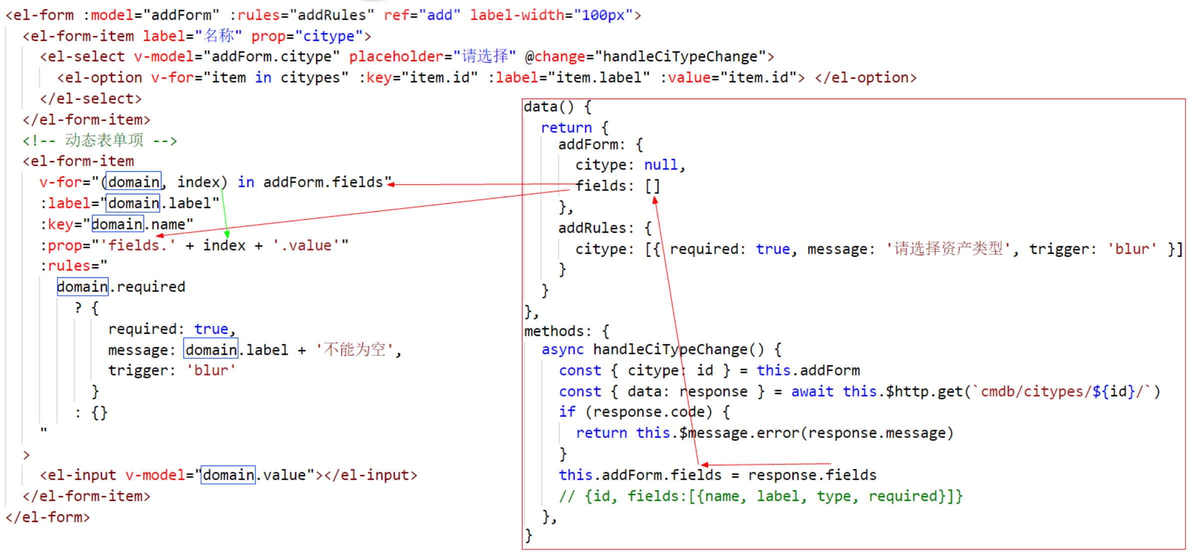
马哥教育 高薪职业学院

动态增加表单项**

参考 https://element.eleme.cn/#/zh-CN/component/form#dong-tai-zeng-jian-biao-dan-xiang

由于Ci Type定义不同的类型，每个类型对应的字段不一样，所以，表单项也要变化，不能提前构建。

返回类型的fields数组后，遍历它。el-form-item组件中使用v-for指令遍历创建表单项。



```
<el-form :model="addForm" :rules="addRules" ref="add" label-width="100px">
  <el-form-item label="名称" prop="citype">
    <el-select v-model="addForm.citype" placeholder="请选择" @change="handleCiTypeChange">
      <el-option v-for="item in citypes" :key="item.id" :label="item.label" :value="item.id"> </el-option>
    </el-select>
  </el-form-item>
  <!-- 动态表单项 -->
  <el-form-item
    v-for="(domain, index) in addForm.fields"
    :label="domain.label"
    :key="domain.name"
    :prop="'fields.' + index + '.value'"
    :rules="
      domain.required
        ? {
            required: true,
            message: domain.label + '不能为空',
            trigger: 'blur'
          }
        : {}
    "
  >
    <el-input v-model="domain.value"></el-input>
  </el-form-item>
</el-form>
```

```
data() {
  return {
    addForm: {
      citype: null,
      fields: []
    },
    addRules: {
      citype: [{ required: true, message: '请选择资产类型', trigger: 'blur' }]
    }
  }
},
methods: {
  async handleCiTypeChange() {
    const { citype: id } = this.addForm
    const { data: response } = await this.$http.get(`cmdb/citypes/${id}/`)
    if (response.code) {
      return this.$message.error(response.message)
    }
    this.addForm.fields = response.fields
    // {id, fields:[{name, label, type, required}]}
  },
}
```

`:prop="'fields.' + index + '.value'"`，这里是v-bind，说明写的是表达式。表达式里面的值**不可以**随便给，就是给下面的:rules规则安个名字，但要和addForm中字段值fields对应。之所以用index，是因为数组用`.index`访问第几个。

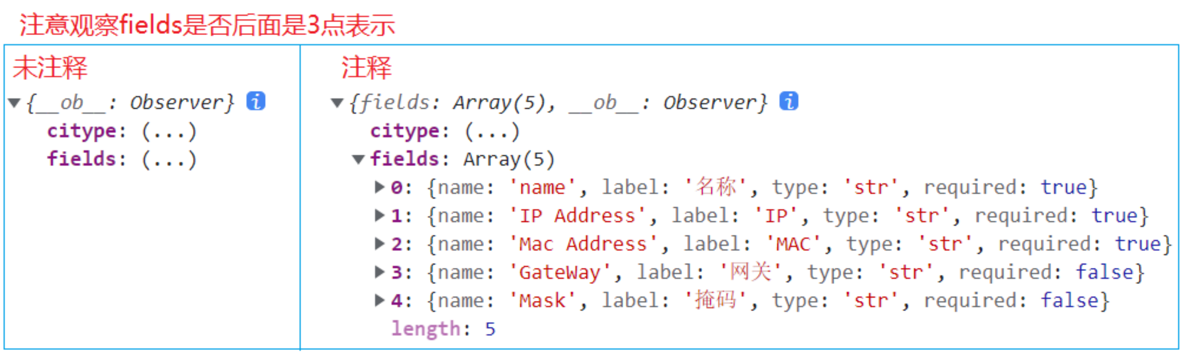
反应式对象

确切地讲，在Vue中，被观察的Reactive（响应式，反应式）对象，发生改变会引起视图更新。

```
<script>
export default {
  created() {
    this.getList()
    this.getAllTypeList()
  },
  data() {
    return {
      addForm: {
        citype: null
        // fields: [] // 注释这一句看看
      },
      addRules: {
        citype: [{ required: true, message: '请选择资产类型', trigger: 'blur' }]
      }
    }
  },
  methods: {
    async handleCiTypeChange() {
      const { citype: id } = this.addForm
      const { data: response } = await this.$http.get(`cmdb/citypes/${id}/`)
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.addForm.fields = response.fields
      console.log(this.addForm) // 注释fields，观察注释前后并对比
      // {id, fields:[{name, label, type, required}]}
    }

  }
}
</script>
```

马哥教育 高薪职业学院



注意观察fields是否后面是3点表示

未注释

```
▼{__ob__: Observer} ℹ
   citype: (...)
   fields: (...)
```

注释

```
▼{fields: Array(5), __ob__: Observer} ℹ
   citype: (...)
  ▼fields: Array(5)
    ▶0: {name: 'name', label: '名称', type: 'str', required: true}
    ▶1: {name: 'IP Address', label: 'IP', type: 'str', required: true}
    ▶2: {name: 'Mac Address', label: 'MAC', type: 'str', required: true}
    ▶3: {name: 'GateWay', label: '网关', type: 'str', required: false}
    ▶4: {name: 'Mask', label: '掩码', type: 'str', required: false}
     length: 5
```

如果是3点表示，它就是被观察的Reactive对象，它的改变会引起Vue的视图更新。

把addForm中的fields注释掉，动态为这对象增加fields属性，这个属性就不是Reactive对象。这需要Vue.set全局函数，或者使用其在实例上的别名this.$set。

```
// this.addForm.fields = response.fields
this.$set(this.addForm, 'fields', response.fields) // 增加Reactive对象
```

列表类型字段处理***

网络接口的type为`list:Network Interface`，怎么处理?

冒号后面写的是类型的名称， 名称是唯一键，查询也可以，未来还要生成多版本。类型新版信息名称不变但是会生成同名新版本记录，新的id。

修改类型测试代码，重新生成一次数据

```
CiTypeField(name='Network Interfaces', label='网络接口',
            type='list:{}:{}'.format(ctni.name, ctni.version))
```

遇到这种问题的时候，必须知道`Network Interface:1`，对应的字段是谁？什么时候去查询字段？.

采用懒查询策略，也就是用户不`增加`，这里不去查询`Network Interface:1`对应的字段。

通过`名称`和`版本`请求后台，如下

```
this.$http.get(`cmdb/citypes/${name}/${version}/`)
```

后台Django视图类中要增加对应的action装饰的handler。

马哥教育 IT人的高薪职业学院

cmdb/view.py

增加action get_by_name_and_version

```
from rest_framework.response import Response
from rest_framework_mongoengine.viewsets import ModelViewSet
from rest_framework.decorators import action
from .models import CiType, Ci
from .serializers import CiTypeSerializer, CiSerializer,
CiTypeFieldsSerializer

class CiTypeViewSet(ModelViewSet):
    # CiType是ODM，因此使用DRFM的ModelViewSet
    queryset = CiType.objects.all()
    serializer_class = CiTypeSerializer
    permission_classes = [] # 移除所有权限要求

    @action(detail=False) # 默认GET
    def all(self, request):
        """返回所有分类。相当于重写list，去掉分页"""
        queryset = self.filter_queryset(self.get_queryset())
        serializer = self.get_serializer(queryset, many=True)
        data = serializer.data
        return Response(data)

    @action(detail=False, url_path='(?P<name>[^/.]+)/(?P<version>\d+)')
    # /cmdb/citypes/Network Interface/1/
    #url_path不支持最新3.x中path的<int:version>语法，只能使用正则。但是version拿到的是str类型
    def get_by_name_and_version(self, request, name, version):
```

```python
        print(name, version, "+++")
        obj = self.get_queryset().get(name=name, version=version) # 名字加版本必须唯一
        serializer = CiTypeFieldsSerializer(obj)
        data = serializer.data
        return Response(data)

    def get_serializer_class(self):
        if 'id' in self.kwargs: # 有id就是详情页替换序列化器
            return CiTypeFieldsSerializer
        return super().get_serializer_class()


class CiViewSet(ModelViewSet):
    queryset = Ci.objects
    serializer_class = CiSerializer
    permission_classes = [] # 移除所有权限要求
```

src/components/cmdb/Cis.vue

遇到list类型，那么还可以动态增加表单项

```html
<template>
  <div>
    <el-breadcrumb separator-class="el-icon-arrow-right">
      <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
      <el-breadcrumb-item>资产管理</el-breadcrumb-item>
      <el-breadcrumb-item>资产列表</el-breadcrumb-item>
    </el-breadcrumb>
    <el-card class="box-card">
      <el-row :gutter="20">
        <el-col :span="12">
          <el-button type="primary" @click="addDialogVisible = true">增加资产</el-button>
        </el-col>
      </el-row>
      <el-table :data="dataList" border style="width: 100%">
        <el-table-column type="index" label="序号"> </el-table-column>
        <el-table-column prop="label" label="名称"> </el-table-column>
        <el-table-column label="操作">
          <el-tooltip content="分配权限" effect="light">
            <el-button type="info" icon="el-icon-plus" size="mini"></el-button>
          </el-tooltip>
          <el-tooltip content="修改" effect="light">
            <el-button type="success" icon="el-icon-edit" size="mini"></el-button>
          </el-tooltip>
          <el-tooltip content="删除" effect="light">
            <el-button type="danger" icon="el-icon-delete" size="mini"></el-button>
          </el-tooltip>
        </el-table-column>
      </el-table>
      <el-pagination
        @current-change="handleCurrentChange"
```

马哥教育 IT人的高薪职业学院

```
        :current-page="pagination.page"
        :page-size="pagination.size"
        layout="total, prev, pager, next, jumper"
        :total="pagination.total"
      >
      </el-pagination>
    </el-card>
    <!-- 添加对话框 -->
    <el-dialog title="增加" :visible.sync="addDialogVisible" @close="resetForm('add')" width="90%">
      <el-form :model="addForm" :rules="addRules" ref="add" label-width="100px">
        <el-form-item label="名称" prop="citype">
          <el-select v-model="addForm.citype" placeholder="请选择" @change="handleCiTypeChange">
            <el-option v-for="item in citypes" :key="item.id" :label="item.label" :value="item.id"> </el-option>
          </el-select>
        </el-form-item>
        <!-- 动态表单项 -->
        <el-form-item
          v-for="(domain, index) in addForm.fields"
          :label="domain.label"
          :key="domain.name"
          :prop="'fields.' + index + '.value'"
          :rules="
            domain.required
              ? {
                  required: true,
                  message: domain.label + '不能为空',
                  trigger: 'blur'
                }
              : {}
          "
        >
          <div v-if="!domain.type.startsWith('list:')"><el-input v-model="domain.value"></el-input></div>
          <div v-else>
            <el-button icon="el-icon-plus" size="mini" plain type="success" @click="handleAddChild(domain)"></el-button>
            <el-card v-for="(def, i) in domain.fieldsDef" :key="`${domain.name}.def.${i}`">
              <!-- <div>{{ def }}</div> -->
              <!-- 动态子表单项 -->
              <el-form-item
                v-for="(subDomain, j) in def.fields"
                :label="subDomain.label"
                :key="subDomain.name"
                :prop="'fields.' + index + '.fieldsDef.' + i + '.fields.' + j + '.value'"
                :rules="
                  subDomain.required
                    ? {
                        required: true,
                        message: subDomain.label + '不能为空',
                        trigger: 'blur'
                      }
                    : {}
```
马哥教育 IT人的高薪职业学院

```
        "
       >
        <el-input v-model="subDomain.value"></el-input>
       </el-form-item>
      </el-card>
     </div>
    </el-form-item>
   </el-form>
   <div slot="footer" class="dialog-footer">
    <el-button @click="addDialogVisible = false">取 消</el-button>
    <el-button type="primary" @click="add">确 定</el-button>
   </div>
  </el-dialog>
 </div>
</template>

<script>
export default {
 created() {
  this.getList()
  this.getAllTypeList()
 },
 data() {
  return {
   dataList: [],
   pagination: { page: 1, size: 1, total: 0 },
   citypes: [],
   // 增加资产对话框
   addDialogVisible: false,
   addForm: {
    citype: null
    // fields: [] // 注释这一句看看
   },
   addRules: {
    citype: [{ required: true, message: '请选择资产类型', trigger: 'blur' }]
   }
  }
 },
 methods: {
  handleCurrentChange(val) {
   console.log(`当前页: ${val}`)
   this.getList(val)
  },
  resetForm(name) {
   console.log(this.$refs[name])
   this.$refs[name].resetFields()
  },
  async getList(page = 1) {
   if (!page) {
    page = 1
   }
   const { data: response } = await this.$http.get('cmdb/cis/', {
    params: {
     page
    }
   })
   if (response.code) {
```

马哥教育 IT人的高薪职业学院

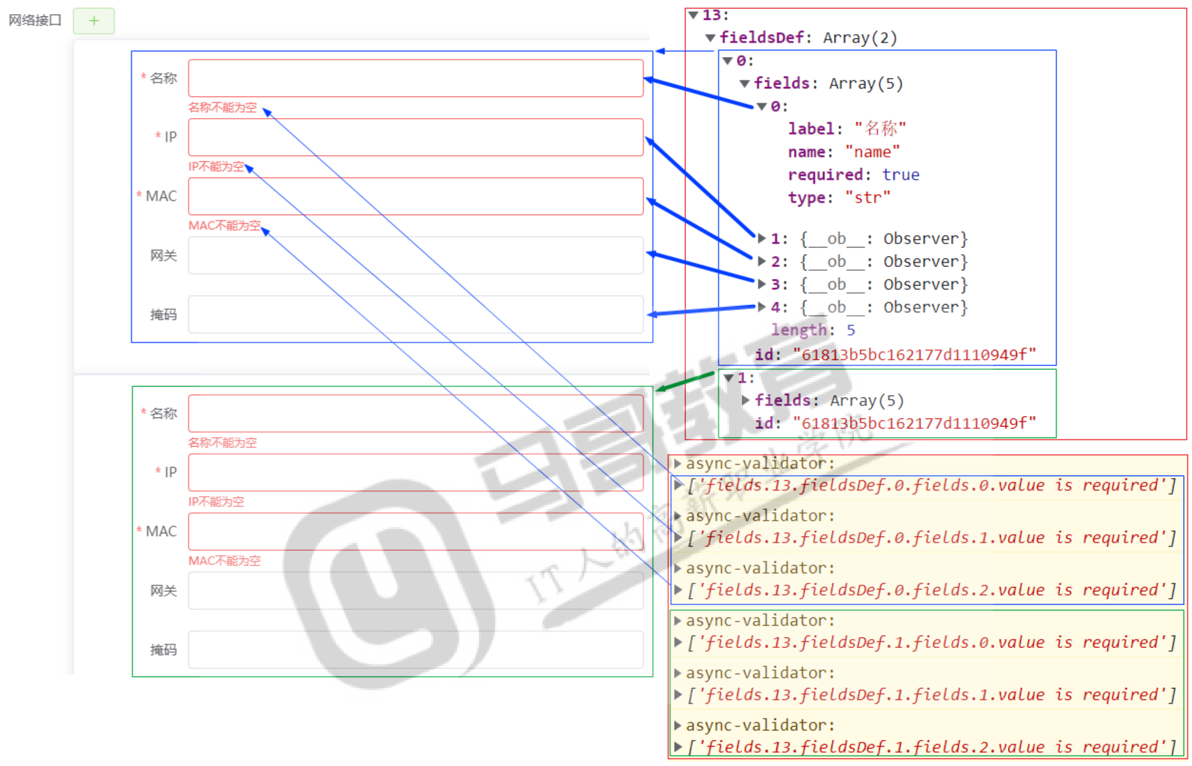
```
        return this.$message.error(response.message)
      }
      this.dataList = response.results
      this.pagination = response.pagination
    },
    async getAllTypeList(page = 1) {
      if (!page) {
        page = 1
      }
      const { data: response } = await this.$http.get('cmdb/citypes/all/')
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.citypes = response
    },
    async handleCiTypeChange() {
      const { citype: id } = this.addForm
      const { data: response } = await this.$http.get(`cmdb/citypes/${id}/`)
      if (response.code) {
        return this.$message.error(response.message)
      }
      // this.addForm.fields = response.fields
      this.$set(this.addForm, 'fields', response.fields) // 增加Reactive对象
      console.log(this.addForm) // 注释fields，观察注释前后并对比
      // {id, fields:[{name, label, type, required}]}
    },
    async handleAddChild(domain) {
      console.log(domain)
      console.log(domain.type.split(':'))
      const [, name, version] = domain.type.split(':')
      const { data: response } = await this.$http.get(`cmdb/citypes/${name}/${version}/`)
      if (response.code) {
        return this.$message.error(response.error)
      }
      if ('fieldsDef' in domain === false) {
        // domain.fieldsDef = [] // 这种新增方式无法触发视图更新
        this.$set(domain, 'fieldsDef', [])
      }
      domain.fieldsDef.push(response)
      console.log(this.addForm)
    },
    add() {
      console.log(this.addForm)
      console.log(this.addRules)
      const name = 'add'
      this.$refs[name].validate(async (valid) => {
        if (valid) {
          // const { data: response } = await this.$http.patch(`users/mgr/${id}/`, this.editUserForm)
          // if (response.code) {
          //   return this.$message.error(response.message)
          // }
          // this.addDialogVisible = false
          this.resetForm(name)
          // this.getUserList(this.pagination.page) // 刷新用户列表
        }
```

马哥教育 IT人的高薪职业学院

```
        })
      }
    }
  }
</script>

<style lang="less" scoped>
.el-form-item .el-card .el-form-item {
  margin-bottom: 22px; /*解决card内form item没有下方空白*/
}
</style>
```

特别注意`:prop="'fields.' + index + '.fieldsDef.' + i + '.fields.' + j + '.value'"`，只有这样才能对应addForm中的字段



```
▼13:
  ▼fieldsDef: Array(2)
    ▼0:
      ▼fields: Array(5)
        ▼0:
            label: "名称"
            name: "name"
            required: true
            type: "str"
        ▶1: {__ob__: Observer}
        ▶2: {__ob__: Observer}
        ▶3: {__ob__: Observer}
        ▶4: {__ob__: Observer}
          length: 5
        id: "61813b5bc162177d1110949f"
    ▼1:
      ▶fields: Array(5)
        id: "61813b5bc162177d1110949f"
▶async-validator:
▶['fields.13.fieldsDef.0.fields.0.value is required']
▶async-validator:
▶['fields.13.fieldsDef.0.fields.1.value is required']
▶async-validator:
▶['fields.13.fieldsDef.0.fields.2.value is required']
▶async-validator:
▶['fields.13.fieldsDef.1.fields.0.value is required']
▶async-validator:
▶['fields.13.fieldsDef.1.fields.1.value is required']
▶async-validator:
▶['fields.13.fieldsDef.1.fields.2.value is required']
```

可以看出，表单再复杂，也不过如此了，只要耐心，可以完成的。

增加CI

利用CiViewSet的POST方法，接收提交的数据。

src/components/cmdb/Cis.vue

```
<template>
  <div>
    <el-breadcrumb separator-class="el-icon-arrow-right">
      <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
      <el-breadcrumb-item>资产管理</el-breadcrumb-item>
      <el-breadcrumb-item>资产列表</el-breadcrumb-item>
    </el-breadcrumb>
    <el-card class="box-card">
      <el-row :gutter="20">
        <el-col :span="12">
          <el-button type="primary" @click="addDialogVisible = true">增加资产</el-button>
        </el-col>
      </el-row>
      <el-table :data="dataList" border style="width: 100%">
        <el-table-column type="index" label="序号"> </el-table-column>
        <el-table-column label="名称">
          <template #default="{ row }">
            {{ row.fields[0].value }}
          </template>
        </el-table-column>
        <el-table-column label="操作">
          <el-tooltip content="修改" effect="light">
            <el-button type="success" icon="el-icon-edit" size="mini"></el-button>
          </el-tooltip>
          <el-tooltip content="删除" effect="light">
            <el-button type="danger" icon="el-icon-delete" size="mini"></el-button>
          </el-tooltip>
        </el-table-column>
      </el-table>
      <el-pagination
        @current-change="handleCurrentChange"
        :current-page="pagination.page"
        :page-size="pagination.size"
        layout="total, prev, pager, next, jumper"
        :total="pagination.total"
      >
      </el-pagination>
    </el-card>
    <!-- 添加对话框 -->
    <el-dialog title="增加" :visible.sync="addDialogVisible" @close="resetForm('add')" width="90%">
      <el-form :model="addForm" :rules="addRules" ref="add" label-width="100px">
        <el-form-item label="名称" prop="citype">
          <el-select v-model="addForm.citype" placeholder="请选择" @change="handleCiTypeChange">
            <el-option v-for="item in citypes" :key="item.id" :label="item.label" :value="item.id"> </el-option>
          </el-select>
        </el-form-item>
        <!-- 动态表单项 -->
        <el-form-item
          v-for="(domain, index) in addForm.fields"
          :label="domain.label"
          :key="domain.name"
          :prop="'fields.' + index + '.value'"
          :rules="
            domain.required
              ? {
                  required: true,
                  message: domain.label + '不能为空',
                  trigger: 'blur'
                }
              : {}
          "
        >
          <div v-if="!domain.type.startsWith('list:')"><el-input v-model="domain.value"></el-input></div>
```

马哥教育 IT人的高薪职业学院

```
          <div v-else>
            <el-button icon="el-icon-plus" size="mini" plain type="success" @click="handleAddChild(domain)"></el-button>
            <el-card v-for="(def, i) in domain.fieldsDef" :key="`${domain.name}.def.${i}`">
              <!-- <div>{{ def }}</div> -->
              <!-- 动态子表单项 -->
              <el-form-item
                v-for="(subDomain, j) in def.fields"
                :label="subDomain.label"
                :key="subDomain.name"
                :prop="'fields.' + index + '.fieldsDef.' + i + '.fields.' + j + '.value'"
                :rules="
                  subDomain.required
                    ? {
                        required: true,
                        message: subDomain.label + '不能为空',
                        trigger: 'blur'
                      }
                    : {}
                "
              >
                <el-input v-model="subDomain.value"></el-input>
              </el-form-item>
            </el-card>
          </div>
        </el-form-item>
      </el-form>
      <div slot="footer" class="dialog-footer">
        <el-button @click="addDialogVisible = false">取 消</el-button>
        <el-button type="primary" @click="add">确 定</el-button>
      </div>
    </el-dialog>
  </div>
</template>

<script>
export default {
  created() {
    this.getList()
    this.getAllTypeList()
  },
  data() {
    return {
      dataList: [],
      pagination: { page: 1, size: 1, total: 0 },
      citypes: [],
      // 增加资产对话框
      addDialogVisible: false,
      addForm: {
        citype: null
        // fields: [] // 注释这一句看看
      },
      addRules: {
        citype: [{ required: true, message: '请选择资产类型', trigger: 'blur' }]
      }
```

马哥教育 IT高薪职业学院

```
    }
  },
  methods: {
    handleCurrentChange(val) {
      console.log(`当前页: ${val}`)
      this.getList(val)
    },
    resetForm(name) {
      console.log(this.$refs[name])
      this.$refs[name].resetFields()
    },
    async getList(page = 1) {
      if (!page) {
        page = 1
      }
      const { data: response } = await this.$http.get('cmdb/cis/', {
        params: {
          page
        }
      })
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.dataList = response.results
      this.pagination = response.pagination
    },
    async getAllTypeList(page = 1) {
      if (!page) {
        page = 1
      }
      const { data: response } = await this.$http.get('cmdb/citypes/all/')
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.citypes = response
    },
    async handleCiTypeChange() {
      const { citype: id } = this.addForm
      const { data: response } = await this.$http.get(`cmdb/citypes/${id}/`)
      if (response.code) {
        return this.$message.error(response.message)
      }
      // this.addForm.fields = response.fields
      this.$set(this.addForm, 'fields', response.fields) // 增加Reactive对象
      console.log(this.addForm) // 注释fields，观察注释前后并对比
      // {id, fields:[{name, label, type, required}]}
    },
    async handleAddChild(domain) {
      console.log(domain)
      console.log(domain.type.split(':'))
      const [, name, version] = domain.type.split(':')
      const { data: response } = await this.$http.get(`cmdb/citypes/${name}/${version}/`)
      if (response.code) {
        return this.$message.error(response.error)
      }
      if ('fieldsDef' in domain === false) {
```
马哥教育 IT人的高薪职业学院

```
        // domain.fieldsDef = [] // 这种新增方式无法触发视图更新
        this.$set(domain, 'fieldsDef', [])
      }
      domain.fieldsDef.push(response)
      console.log(this.addForm)
    },
    add() {
      // console.log(this.addForm)
      const name = 'add'
      this.$refs[name].validate(async (valid) => {
        if (valid) {
          const { data: response } = await this.$http.post('cmdb/cis/', this.addForm)
          if (response.code) {
            return this.$message.error(response.message)
          }
          this.addDialogVisible = false
          this.resetForm(name)
          this.getList() // 刷新列表
        }
      })
    }
  }
}
</script>

<style lang="less" scoped>
.el-form-item .el-card .el-form-item {
  margin-bottom: 22px; /*解决card内form item没有下方空白*/
}
</style>
```

马哥教育 IT人的高薪职业学院

至此，核心功能基本开发完成。望同学们能抛砖引玉，解决生产需求。