测试计划功能更新

测试计划基本模型、序列化器，视图已经创建好，接下来先为前端对接进行优化

首先是列出数据展示部分增加项目、测试人员、测试环境

可以看下API文档当前的列出展示数据，然后进行针对性的修改。

首先需要改的就是序列化器，入参出参统一在这里修改

PlanSerializer

```
"""
@author: haiwen
@date: 2021/6/26
@file: task.py
"""
from sqtp.models import Plan, Environment

from rest_framework import serializers
from .mgr import EnvironmentSerializer
from .auth import UserSerializer
from .hr3 import CaseSerializer
from sqtp.models import User, Case


class PlanSerializer(serializers.ModelSerializer):
    case_ids = serializers.PrimaryKeyRelatedField(queryset=Case.objects.all(),
many=True, required=False,write_only=True)
    cases = CaseSerializer(read_only=True,many=True)
    status = serializers.SerializerMethodField()
    environment = EnvironmentSerializer(read_only=True)
    executor = UserSerializer(read_only=True)
    create_time = serializers.DateTimeField(format='%Y-%m-%d %H:%M:%S',
read_only=True)
    update_time = serializers.DateTimeField(format='%Y-%m-%d %H:%M:%S',
read_only=True)
    executor_id = serializers.IntegerField(write_only=True)
    environment_id = serializers.IntegerField(write_only=True)
    create_by = UserSerializer(write_only=True,required=False)
    updated_by = UserSerializer(write_only=True,required=False)

    class Meta:
        model = Plan
        fields = ['case_ids', 'id', 'name', 'desc', 'status', 'exec_counts',
'cases', 'environment', 'executor',
                  'create_time', 'update_time', 'environment_id',
'executor_id','create_by','updated_by']

    def get_status(self, obj):
        return obj.get_status_display()

    def validate(self, attrs):
        executor_id = attrs.get('executor_id', 0)
        if not User.objects.filter(pk=executor_id).count():
```

```
            raise serializers.ValidationError(f'请传入正确的executor_id:
{executor_id}')
        environment_id = attrs.get('environment_id', 0)
        if not Environment.objects.filter(pk=environment_id).count():
            raise serializers.ValidationError(f'请传入正确的environment_id:
{environment_id}')

        return attrs

    def update(self, instance, validated_data):
        # 关联case
        case_ids = validated_data.pop('case_ids')  # 取出关联的case数据对象
        instance.cases.set(case_ids)  # 多对多关系关联
        # 设置属性--反射
        for k, v in validated_data.items():
            setattr(instance, k, v)
        instance.save()
        return instance

    class Meta:
        model = Plan
        fields = ['case_ids', 'id', 'name', 'desc', 'status', 'exec_count',
'cases', 'environment', 'executor',
                  'create_time', 'update_time', 'environment_id', 'executor_id']
```

接下来视图部分

PlanViewSet

```
"""
@author: haiwen
@date: 2021/6/26
@file: task.py
"""
from sqtp.models import  Project, Environment,Plan
from rest_framework import generics, viewsets
# 计划视图
from sqtp.serializers import PlanSerializer


class PlanViewSet(viewsets.ModelViewSet):
    queryset = Plan.objects.all()
    serializer_class = PlanSerializer
    # 同步创建者
    def perform_create(self, serializer):
        serializer.save(creator=self.request.user)

    # 同步更新者
    def perform_update(self, serializer):
        serializer.save(updater=self.request.user)
```

EnvironmentSerializer

创建测试计划需要关联环境数据，更新原有序列化器

```
# serializers/mgr.py

class EnvironmentSerializer(serializers.ModelSerializer):
    project_id = serializers.IntegerField(write_only=True)
    project = ProjectSerializer(read_only=True)
    category = serializers.SerializerMethodField()
    os = serializers.SerializerMethodField()
    status = serializers.SerializerMethodField()

    def get_status(self,obj):
        return obj.get_status_display()

    def get_os(self,obj):
        return obj.get_os_display()

    def get_category(self,obj):
        return obj.get_category_display()

    def validate_project_id(self, project_id):
        if not Project.objects.filter(pk=project_id).count():
            raise serializers.ValidationError('请传递正确的project_id')
        return project_id

    class Meta:
        model = Environment
        fields = '__all__'
```

新建环境，新建测试计划。

计划执行

allure测试报告

更新计划运行方法，增加报告的收集

安装pytest-allure报告插件

```
pip install allure-pytest
```

执行时生成报告缓存文件

```
hrun testcase\测开3期_case001.json --alluredir=report/tmp  #
```

allure报告的生成

```
allure generate report/tmp -o report/html
```

格式是：

```
allure generate 报告缓存目录 -o 报告文件目录
```

生成的报告文件直接打开是看不到内容的，需要通过服务器访问，我们可以将其暂存在静态文件的目录(dist)，然后浏览器访问相对目录即可，如：http://127.0.0.1:8081/report/allure/index.html

后面我们要做的就是将报告文件的路径分别存储好

执行测试计划其实是执行其关联的测试用例，在HR3中，用例之间不需要定义顺序，用例应该作为独立的个体存在，所以直接执行用例路径即可。

执行用例前，删除testcase中其他文件，防止出现干扰

```
# sqtp/utils.py

...

def setup_case_dir(case_path):
    empty_dir_files(case_path,'json','pyc','py')


def empty_dir_files(path, *suffix):
    for root,dirs,files in os.walk(path):
        for fi in files:
            if fi.split('.')[-1] in suffix:
                print(os.path.join(root,fi))
                os.remove(os.path.join(root,fi))
```

```
class PlanViewSet(viewsets.ModelViewSet):
    queryset = Plan.objects.all()
    serializer_class = PlanSerializer
    # 同步创建者
    def perform_create(self, serializer):
        serializer.save(creator=self.request.user)

    # 同步更新者
    def perform_update(self, serializer):
        serializer.save(updater=self.request.user)

    @action(methods=['GET'], detail=True, url_path='run', url_name='run_plan')
    def run(self,request,pk):
        plan = Plan.objects.get(pk=pk)  # 根据id获取计划
        plan.status = 1  # 状态设置为开始执行
        plan.save()
        setup_case_dir('testcase') # 初始化用例目录
        case_list=[] #用例路径
        for case in plan.cases.all(): # 取出关联用例  plan.cases.all()
            serializer = CaseSerializer(instance=case)  # 调用序列化器
            path = serializer.to_json_file()  # 生成用例文件
            case_list.append(path)
        # hr3运行测试用例
```

```
        exit_code = main_run([*case_list,'--alluredir=report/tmp'])  # 入参是列表
=pytest参数包括用例文件路径，Pytest命令选项
        # 记录测试计划状态和执行次数
        plan.exec_counts = plan.exec_counts + 1
        plan.status = 3
        plan.save()
        # 只要exit_code不是0，就是执行失败了
        if exit_code != 0:
            return Response(status=status.HTTP_500_INTERNAL_SERVER_ERROR,
                            data={'error': 'failed run case', 'retcode':
exit_code})
        return Response(data={'retcode': status.HTTP_200_OK, 'msg': 'run
success'})
```

测试报告收集

模型

新建模型用于存储报告信息。

```
# models/task.py

class Report(CommonInfo):
    # 关联计划
    plan = models.ForeignKey(Plan, on_delete=models.DO_NOTHING,
related_name='reports')
    # 报告路径
    path = models.CharField('报告路径', max_length=500)
    # 报告详情
    detail = models.TextField('报告详情')
    # 执行人
    trigger =
models.ForeignKey(settings.AUTH_USER_MODEL,on_delete=models.DO_NOTHING,null=True
)
```

```
# models/__init__.py.py

from .task import Plan,Report
```

迁移到数据库

```
python manage.py makemigrations
python manage.py migrate
```

序列化器

为了配合前端展示报告的数据，序列化器需要配置相关的字段

```
class ReportSerializer(serializers.ModelSerializer):
    plan = PlanSerializer(read_only=True)
    trigger = UserSerializer(read_only=True)
    create_time = serializers.DateTimeField(format='%Y-%m-%d %H:%M:%S',
read_only=True)
    update_time = serializers.DateTimeField(format='%Y-%m-%d %H:%M:%S',
read_only=True)

    class Meta:
        model = Report
        fields = ['id', 'plan', 'trigger', 'detail','create_time',
'update_time', 'path']
```

视图

```
class ReportViewSet(viewsets.ModelViewSet):
    queryset = Report.objects.all()
    serializer_class = ReportSerializer

    # 报告只提供查询功能
    def create(self, request, *args, **kwargs):
        return Response({'retcode':404,'msg':'no such method'},status=404)

    def update(self, request, *args, **kwargs):
        return Response({'retcode':404,'msg':'no such method'},status=404)
```

注册路由

```
#sqtp/urls.py
router.register(r'reports',views.ReportViewSet)
```