flask + gunicorn的简单应用

构建conda虚拟环境：

```
conda create -n cnv python=3.9
conda activate cnv
pip install gunicorn

pip install click
pip install flask
pip install flask-wtf
pip install bootstrap-flask
pip install pandas
pip install numpy
pip install PyYAML
```

基础框架：

代码文件夹： demo

gunicorn.conf.py

wsgi.py

gunicorn.conf.py

```
workers = 8
bind = '0.0.0.0:5090'
accesslog = './gunicorn_access.log'
errorlog = './gunicorn_error.log'
loglevel = 'warning'
backlog = 2048
```

wsgi.py

```
from demo import create_app
app = create_app()
```

```
conda install snakemake
conda create --name flask2024 python=3.9
conda activate flask2024
conda install -c bioconda seqtk
conda install -c bioconda cutadapt
conda install samtools


pip install gunicorn
pip install click
pip install flask
pip install flask-wtf
pip install bootstrap-flask
pip install pandas
pip install numpy
pip install PyYAML


pip install Bio
pip install Levenshtein
pip install pysam

sudo firewall-cmd --permanent --add-port=6090/tcp
sudo firewall-cmd --reload
```

修改

biology.py

form.output_dir.data

snk_config.py

ref_ana

script_file

snakemake_cmd

修改snkfile

```
ssh-keygen -t ed25519 -C "1004808412@qq.com"
git checkout -b v1.0.1
git add --all
git commit -m "Merged remote changes"
git push -u origin v1.0.1
```

demo文件夹：

脚本文件夹：blueprint
css&js文件夹：static
html文件夹：templates
__init__.py
forms.py

__init__.py:

```
import os
import click
import json
from flask import Flask
from flask_bootstrap import Bootstrap4
from gmxcnv.blueprint.mapping import mapping_bp


class BaseConfig:
    SECRET_KEY = os.getenv('SECRET_KEY', 'dev key')
    OUT_PATH = '/home/wangzc/temp9'
    BOWTIE_DB_PATH = os.path.join(OUT_PATH, 'db', 'hg19')
    BISMARK_DB_PATH = os.path.join(OUT_PATH, 'db')
    USER_CONFIG = json.load(open(os.path.join(OUT_PATH, '.config'), 'rt'))
    RAW_PATH = USER_CONFIG['RAW_PATH']
    QS_PATH = USER_CONFIG['QS_PATH']
    QS_ACCESS = USER_CONFIG['QS_ACCESS']
    QS_SECRET = USER_CONFIG['QS_SECRET']
    QS_BUCKET =  USER_CONFIG['QS_BUCKET']
    QS_ZONE = USER_CONFIG['QS_ZONE']

def create_app(config_name=None):
    app = Flask('gmxcnv')
    app.config.from_object(BaseConfig)
    bootstrap=Bootstrap4()
    bootstrap.init_app(app)
    app.register_blueprint(mapping_bp)
    return app
```

forms.py

```
from flask_wtf import FlaskForm
from wtforms import SubmitField, SelectField, BooleanField, StringField
from wtforms.validators import DataRequired, Length, Optional, URL
from wtforms import SelectField, IntegerField

class MappingConfigForm(FlaskForm):
    data_dir = SelectField('选择原始数据目录', validate_choice=False, choices=[], validators=[DataRequired()],
description='选择原始数据所在的目录，该目录中存在*fastq.gz文件')
    next_data = SubmitField('下一级目录')
    output_dir = SelectField('选择结果输出目录', validate_choice=False, validators=[DataRequired()],
description='选择一个结果输出目录')
    query = SubmitField('结果查询')
    cpu_count = IntegerField('CPU数量', validators=[DataRequired()], description='CPU数量，必须为正整数，
不超过机器最大CPU数量')
    # db_dir = SelectField('选择数据库目录', validate_choice=False, validators=[DataRequired()], description='选
择基因组数据库目录')
    meth = BooleanField('甲基化数据')
    submit = SubmitField('开始比对')

class MappingCheckForm(FlaskForm):
    data_ana  = StringField('数据目录', render_kw={'readonly':True})
    output_ana = StringField('输出目录', render_kw={'readonly':True})
    cpu_ana = StringField('cpu数量')
    # db_ana = StringField('数据库目录')
    meth_ana = StringField('甲基化', render_kw={'readonly':True})
    submit_ana = SubmitField('开始比对')

class UploadForm(FlaskForm):
    res_dir = StringField('结果目录', render_kw={'readonly':True})
    upload_res = SubmitField('上传结果', id='upload', render_kw={'hidden':True})
```

blueprint文件夹

__init__.py	#空文件
mapping.py	#主文件
qingstor.py	#提交外部服务器
submit_cmd.py	#提交内部服务器
run_bedtools.sh	#命令脚本
hg19_genemind.bed	#备用文件

html文件夹：templates

基础框架：base.html
继承框架的项目文件夹：mapping

mapping.html	比对信息初始页面
check.html	检查参数信息
query.html	运行等待页面
upload.html	上传云端

html知识点：Flask+bootstrap Flask_WTF 定义模板

{% block content %}
{% endblock content %}

{% block footer %}
{% endblock footer %}

base.html

```
{% from 'bootstrap4/nav.html' import render_nav_item %}
<!DOCTYPE html>
<html>

<head>
  {% block head %}
    <title>{% block title %}{% endblock title %} </title>
    <link rel="icon" href="{{ url_for('static', filename='ico/heart.ico') }}">
        <link rel="stylesheet" href="{{ url_for('static', filename='css/blue.mini.css') }}" type="text/css">
        <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}" type="text/css">
    <script src="https://libs.baidu.com/jquery/2.1.4/jquery.min.js"></script>
  {% endblock head %}
</head>

{% block nav %}
  <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
    <div class="container">
      <div class="collapse navbar-collapse" id="navbarColor01">
        <ul class="navbar-nav mr-auto">
          {{ render_nav_item('mapping.index', '主页') }}
        </ul>
      </div>
    </div>
  </nav>
{% endblock nav %}

{% block content %}
{% endblock content %}

</body>
</html>
```

head部分定义了head模板，嵌套的title模板，引入两个css样式，一个缓存js
核心模板ender_nav_item，引用了模板nav
最后末尾将content定义为全局模板