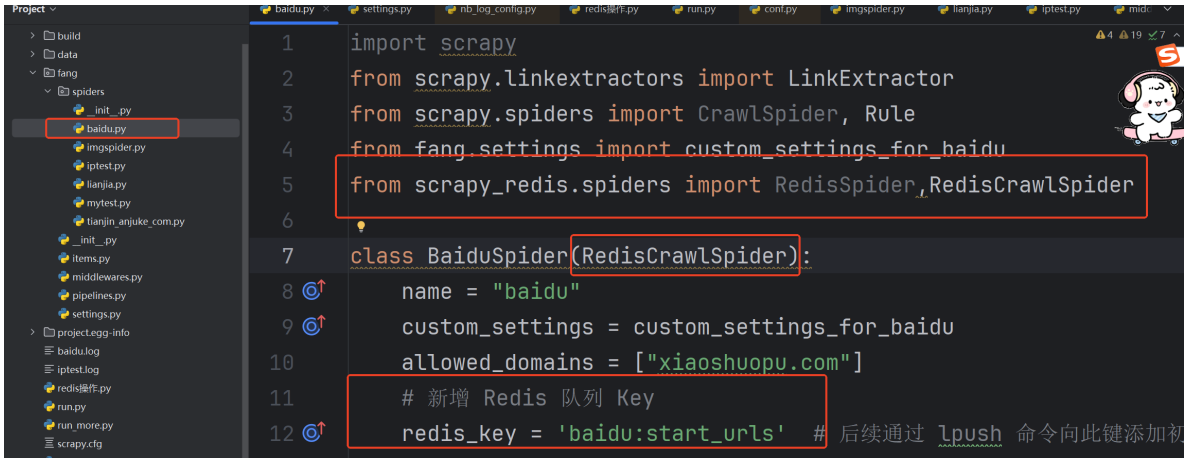
先装包

```
pip install scrapy-redis
```

爬虫部分的改造（crawl爬虫）

```
import scrapy
from scrapy.linkextractors import LinkExtractor
from scrapy.spiders import CrawlSpider, Rule
from fang.settings import custom_settings_for_baidu
from scrapy_redis.spiders import RedisSpider,RedisCrawlSpider

class BaiduSpider(RedisCrawlSpider):
    name = "baidu"
    custom_settings = custom_settings_for_baidu
    allowed_domains = ["xiaoshuopu.com"]
    # 新增 Redis 队列 Key
    redis_key = 'baidu:start_urls'  # 后续通过 lpush 命令向此键添加初始
    # start_urls = [f"https://www.xiaoshuopu.com/class_{type}/" for type in
range(1,4)]

    # rules = (Rule(LinkExtractor(allow=r"Items/"), callback="parse_item",
follow=True),)
    # 爬取规则配置
    rules = (
        Rule(
            LinkExtractor(
                allow=r'https://www.xiaoshuopu.com/xiaoshuo/\d+/\d+/',
                deny=r'https://www.xiaoshuopu.com/xiaoshuo/\d+/\d+/\d+\.html',
                # restrict_xpaths='//*[@id="at"]'
            ),
            callback='parse_books',
            follow=True
        ),
        Rule(
            LinkExtractor(
                allow=r'https://www.xiaoshuopu.com/xiaoshuo/\d+/\d+/\d+\.html',
                restrict_xpaths='//*[@id="at"]'
            ),
            callback='parse_item',
            follow=False
        ),

    )
    def parse_books(self, response):
        book_name = response.xpath('//h1/text()').extract()
        print('book_name:',book_name,response.url)

    def parse_item(self, response):
        title = response.xpath('//*[@id="amain"]/dl/dd[1]/h1/text()').extract()
        content = response.xpath('//*[@id="htmlContent"]//text()').extract()
        print('title:',title,response.url)
        # item = {}
        #item["domain_id"] = response.xpath('//input[@id="sid"]/@value').get()
```

```
        #item["name"] = response.xpath('//div[@id="name"]').get()
        #item["description"] = response.xpath('//div[@id="description"]').get()
        # return item
```



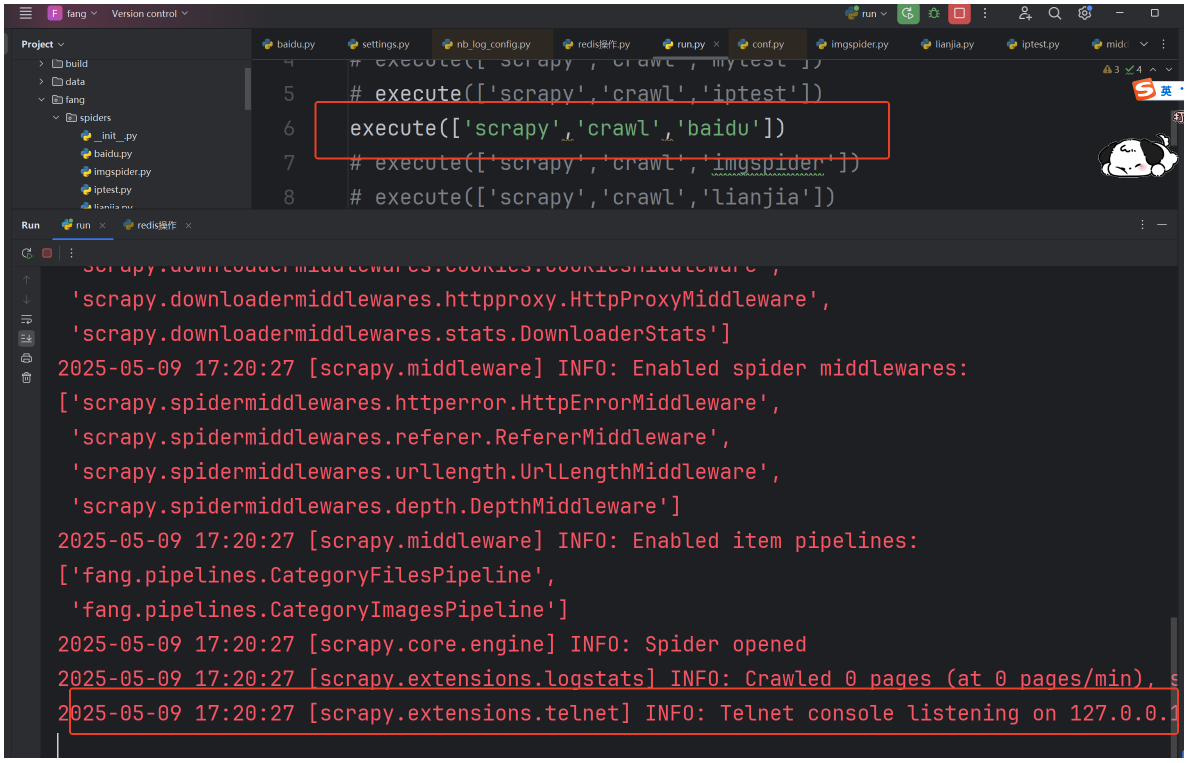
这里可以看到有个

custom_settings=custom_settings_for_baidu这块对应着

setting位置的配置

```
custom_settings_for_baidu = {
    # 启用 Redis 调度器
    'SCHEDULER': 'scrapy_redis.scheduler.Scheduler',
    # 启用 Redis 去重过滤器
    'DUPEFILTER_CLASS': 'scrapy_redis.dupefilter.RFPDupeFilter',
    # Redis 连接地址（根据实际环境修改）
    'REDIS_URL': 'redis://localhost:6379/0',
    # 持久化队列（重启爬虫不丢失任务）
    'SCHEDULER_PERSIST': True,
    # 可选：启用 Redis 存储 Item
    # 'ITEM_PIPELINES': {
    #     'scrapy_redis.pipelines.RedisPipeline': 300
    # }
}
```

这个时候运行启动文件，该爬虫不会爬取，而是在等待任务链接写入



再搞个python文件像redis数据库里写入起始的任务链接，因为现在的爬虫是从redis数据库里读取完成任务的

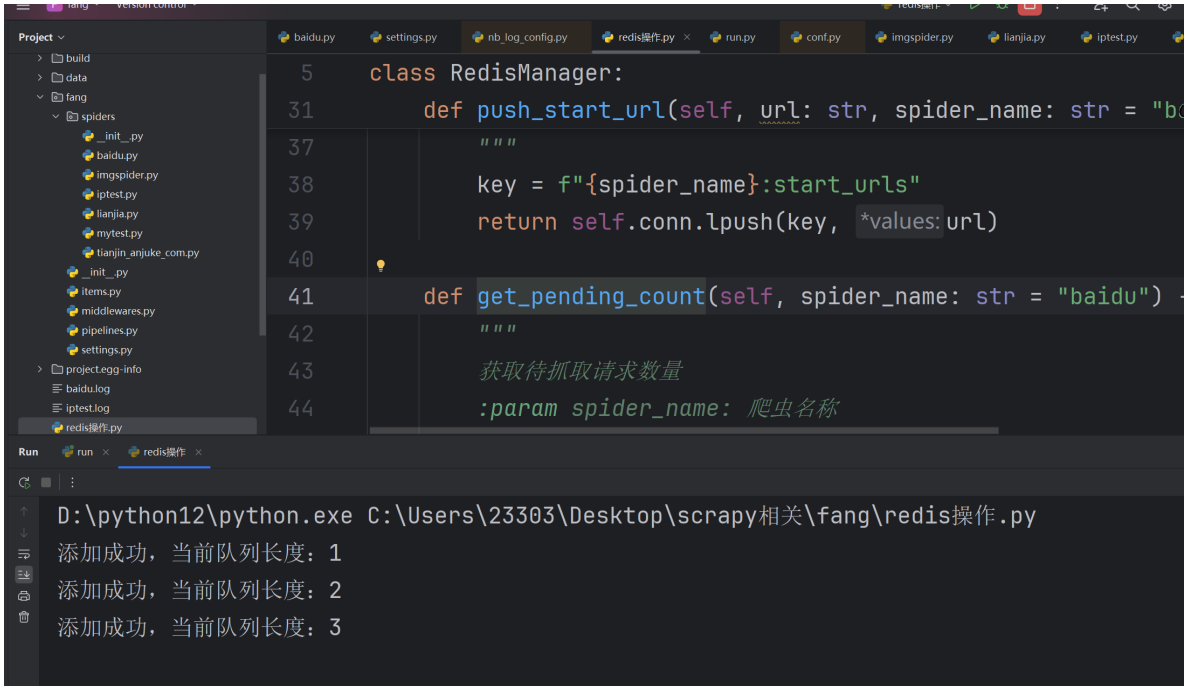
具体代码如下

```
import redis
from typing import Set, Optional


class RedisManager:
    """Redis 分布式爬虫操作封装工具类"""

    def __init__(self,
                 host: str = 'localhost',
                 port: int = 6379,
                 db: int = 0,
                 password: Optional[str] = None,
                 decode_responses: bool = True):
        """
        初始化Redis连接
        :param host: Redis服务器地址
        :param port: Redis端口
        :param db: 数据库编号
        :param password: 访问密码
        :param decode_responses: 是否自动解码响应
        """
        self.pool = redis.ConnectionPool(
            host=host,
            port=port,
            db=db,
            password=password,
            decode_responses=decode_responses
        )
        self.conn = redis.StrictRedis(connection_pool=self.pool)

    def push_start_url(self, url: str, spider_name: str = "baidu") -> int:
```

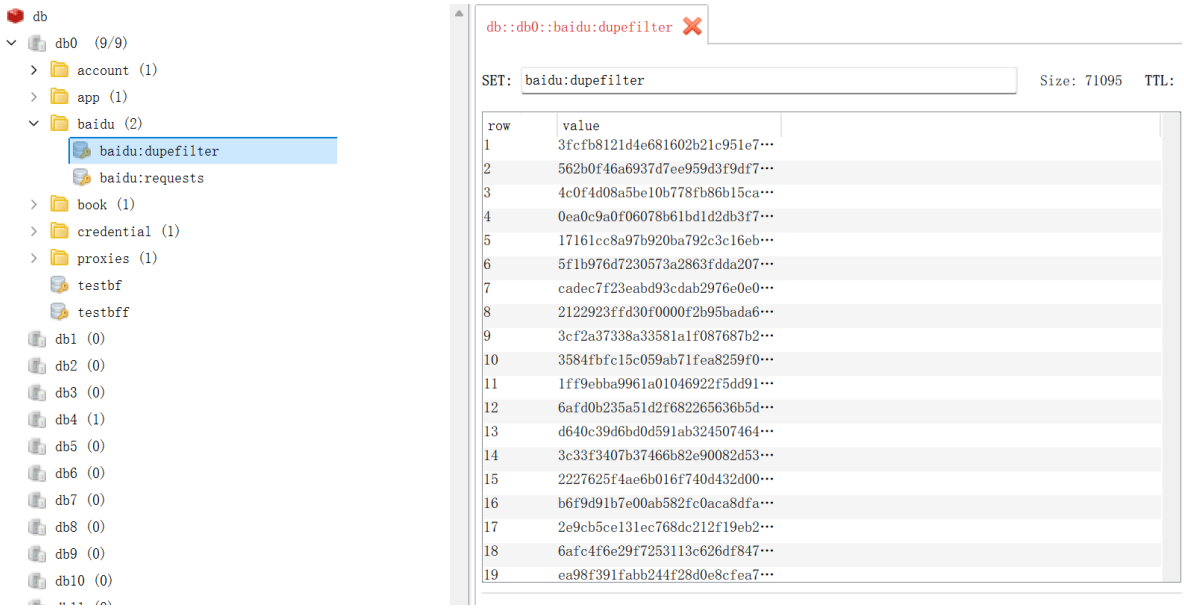
```python
        """
        向起始URL队列添加新URL
        :param url: 要添加的URL
        :param spider_name: 爬虫名称（默认为baidu）
        :return: 添加后队列长度
        """
        key = f"{spider_name}:start_urls"
        return self.conn.lpush(key, url)

    def get_pending_count(self, spider_name: str = "baidu") -> int:
        """
        获取待抓取请求数量
        :param spider_name: 爬虫名称
        :return: 队列当前长度
        """
        key = f"{spider_name}:requests"
        return self.conn.llen(key)

    def get_fingerprints(self, spider_name: str = "baidu") -> Set[str]:
        """
        获取去重指纹集合
        :param spider_name: 爬虫名称
        :return: 去重指纹集合
        """
        key = f"{spider_name}:dupefilter"
        return self.conn.smembers(key)

    def close(self):
        """关闭连接池"""
        self.pool.disconnect()


# 使用示例
if __name__ == "__main__":
    # 初始化连接
    redis_mgr = RedisManager(host='localhost',
                             port=6379,
                             db=0,
                             password='')

    try:
        start_urls = [f"https://www.xiaoshuopu.com/class_{type}/" for type in
range(1,4)]

        # 添加初始URL
        for url in start_urls:
            count = redis_mgr.push_start_url(url)
            print(f"添加成功，当前队列长度：{count}")

        # # 查看待抓取数量
        # pending = redis_mgr.get_pending_count()
        # print(f"待处理请求数：{pending}")
        #
        # # 获取去重指纹
        # fingerprints = redis_mgr.get_fingerprints()
        # print(f"当前去重指纹数量：{len(fingerprints)}")

    finally:
```

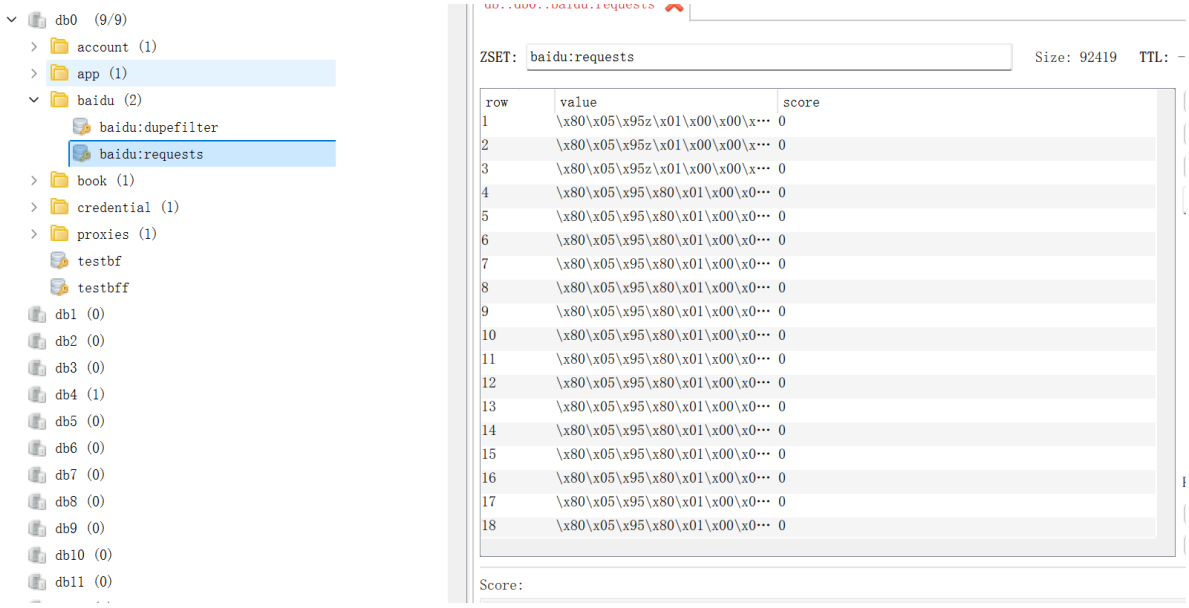
```
        redis_mgr.close()
```

我们目前仅仅调用添加初始任务的功能



刚才等代的爬虫会自己启动





查看redis可以看到去重队列和任务队列

如果是基础模版，那爬虫部分这么改

```
import scrapy
from fang.items import BookItem
from scrapy_redis.spiders import RedisSpider,RedisCrawlSpider
from fang.settings import custom_settings_for_mytest

class MytestSpider(RedisSpider):
    name = "mytest"
    allowed_domains = ["xiaoshuopu.com"]
    custom_settings = custom_settings_for_mytest
    redis_key = 'mytest:start_urls'
    # start_urls = ["https://www.xiaoshuopu.com/xiaoshuo/69/69434/"]

    def parse(self, response):
        chapter_url = response.xpath('//td[@class="L"]/a/@href').extract()
        for url in chapter_url:
            # 'https://www.xiaoshuopu.com/xiaoshuo/69/69434/32780156.html'
            yield
scrapy.Request(f'https://www.xiaoshuopu.com{url}',self.get_content)
    def get_content(self, response):
        item = BookItem()
        item['title'] = response.xpath('//h1/text()').extract_first()
        item['contents'] =
''.join(response.xpath('//div[@id="htmlContent"]//text()').extract())
        print(item)
        return item
```

其他的没什么区别