FME 在 Web 数据爬虫及可视化中的应用

一、背景

为什么要做爬虫？

首先请问：都说现在是"大数据时代"，那数据从何而来？

· 企业产生的用户数据：百度指数、阿里指数、TBI 腾讯浏览指数、新浪微博指数

· 数据平台购买数据：数据堂、国云数据市场、大数据交易所

· 政府/机构公开的数据：中华人民共和国国家统计局数据、世界银行公开数据、联合国数据

· 爬取网络数据：如果需要的数据市场上没有，或者不愿意购买，那么可以选择招/做一名爬虫工程师，自己动手丰衣足食。

爬虫是什么？

网络爬虫（又称为网页蜘蛛，网络机器人，在 FOAF 社区中间，更经常的称为网页追逐者），是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本。另外一些不常使用的名字还有蚂蚁、自动索引、模拟程序或者蠕虫。

二、原理

通用爬虫和聚焦爬虫

根据使用场景，网络爬虫可分为 **通用爬虫** 和 **聚焦爬虫** 两种。

· 通用爬虫

通用网络爬虫 是 搜索引擎抓取系统的重要组成部分。主要目的是将互联网上的网页下载到本地，形成一个互联网内容的镜像备份。

通用搜索引擎（Search Engine）工作原理

通用网络爬虫 从互联网中搜集网页，采集信息，这些网页信息用于为搜索引擎建立索引从而提供支持，它决定着整个引擎系统的内容是否丰富，信息是否即时，因此其性能的优劣直接影响着搜索引擎的效果。

· 聚焦爬虫

聚焦爬虫，是"面向特定主题需求"的一种网络爬虫程序，它与通用搜索引擎爬虫的区别在于： **聚焦爬虫在实施网页抓取时会对内容进行处理筛选，尽量保证只抓取与需求相关的网页信息。**

三、流程

第一步：获取页面

FME 网络爬虫的基本工作流程如下：

（1）首先选取种子 URL，通过种子 URL 对服务器发送首次请求。
（2）将首次请求获取到的数据存储到本地后，加以分析及计算，得到需要请求的总次数以及新的 URL。
（3）设置请求随机延迟。
（4）通过新构建的 URL，依次进入下一步请求并提取需要的数据内容。
（5）数据解析统计及成果输出。

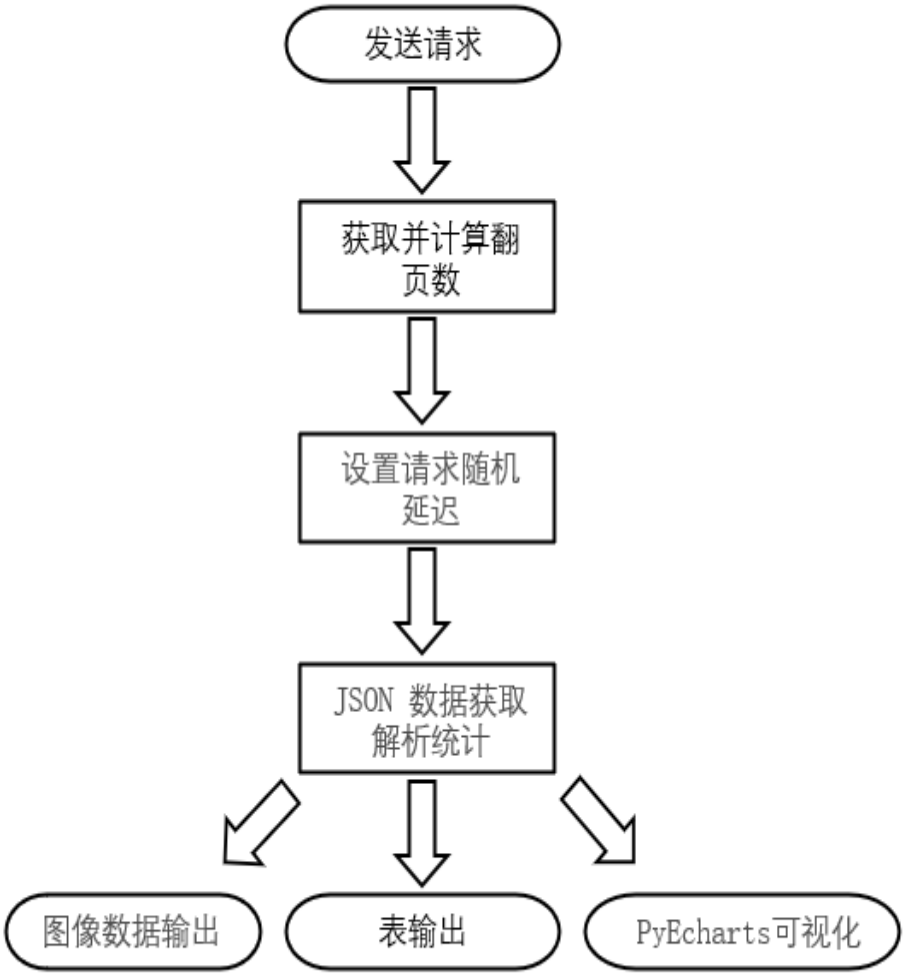

图 1：流程图

第二步：数据存储

通过爬虫爬取到的网页数据，将数据存入本地或是数据库（数据内容可以是字符串也可以是文件，根据具体需要提取相应内容）。其中的数据内容与用户浏览器访问的 HTML 所看到的内容是完全一样的。

第三步：数据清洗

将爬虫抓取回来的数据，进行各种步骤的预处理从而得到想要的数据。

- 提取文字
- 中文分词
- 链接关系计算
- 特殊文件处理
- …

四、案例

1. 农村乱占耕地建房中的应用

初期由于工作紧、任务重，软件公司开发的 Web 数据汇总平台功能简陋，缺少数据批量导出以及统计等功能。根据地方对统计功能的需求，通过 FME 搭配 Python 实现对平台的数据进行抓取以及统计、汇总、可视化展示。

1.1 数据来源说明：

该平台为外业数据采集汇总平台，数据均是通过外业采集填报，数据共分为以下三类：

（1）矢量数据：国家下发以及地方自提图斑
（2）属性数据：外业填报的表单
（3）图像数据：外业拍照举证的照片

由于案例需求为统计汇总，所以矢量数据部分可忽略。

1.2 整体思路：

通过对页面源码的分析发现，属性数据全部以 JSON 字符串的形式进行存储，并且一次请求只能获得一页的数据，也就是说需要找到翻页数属性，然后计算翻页数。图像数据部分则直接通过 URL 地址访问，对多个图片 URL 对比分析发现，每一个图片的地址不同之处是由 GEOID 属性进行区分，而 GEOID 属性正好存储在 JSON 字符串中，通过拼接 URL 既可获取相应图片。可视化部分则是把获取的数据统计之后传输给 PythonCaller 转换器进行展示。

模板搭建整体流程如下图 2：

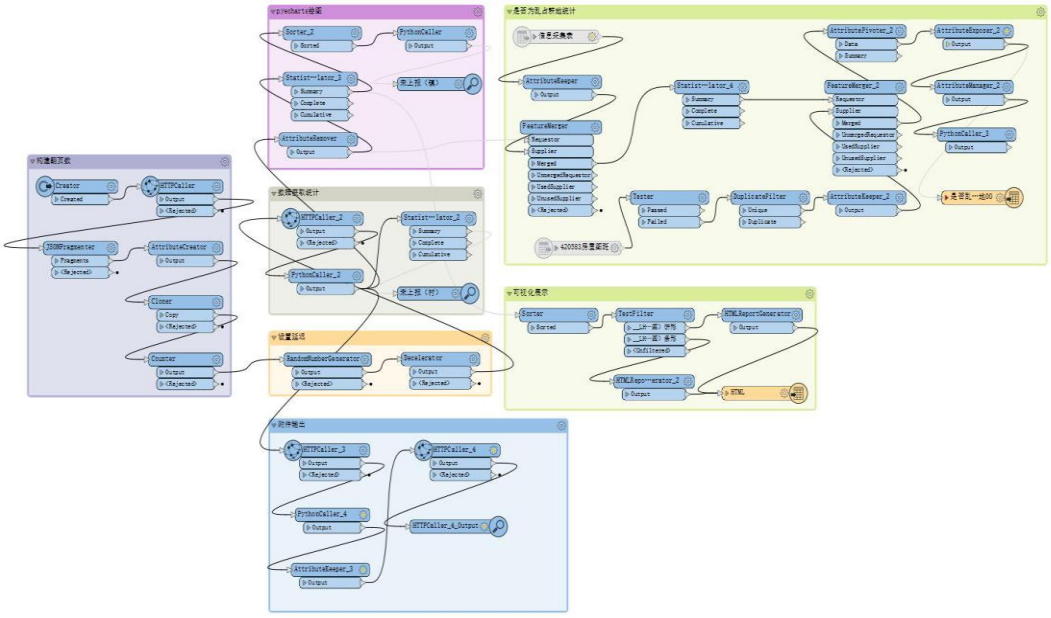

图 2：模板总图

1.3 模板详解：

（1）发送请求并计算翻页：

首先查看页面源码，找到数据存储位置的 URL 地址。通过 Creator 转换器创建一个要素传入 URL 地址来触发 HTTPCaller 转换器（该转换器类似于 Python 的 Requests 模块）对服务器发送一次请求获取到 JSON 字符串数据，提取 JSON 字符串中 TotalCount（数据总数量）属性，然后把 TotalCount 属性与每一页总条数 1000 进行简单的除法运算得到总页数（需请求总次数）。如下图 3 所示：

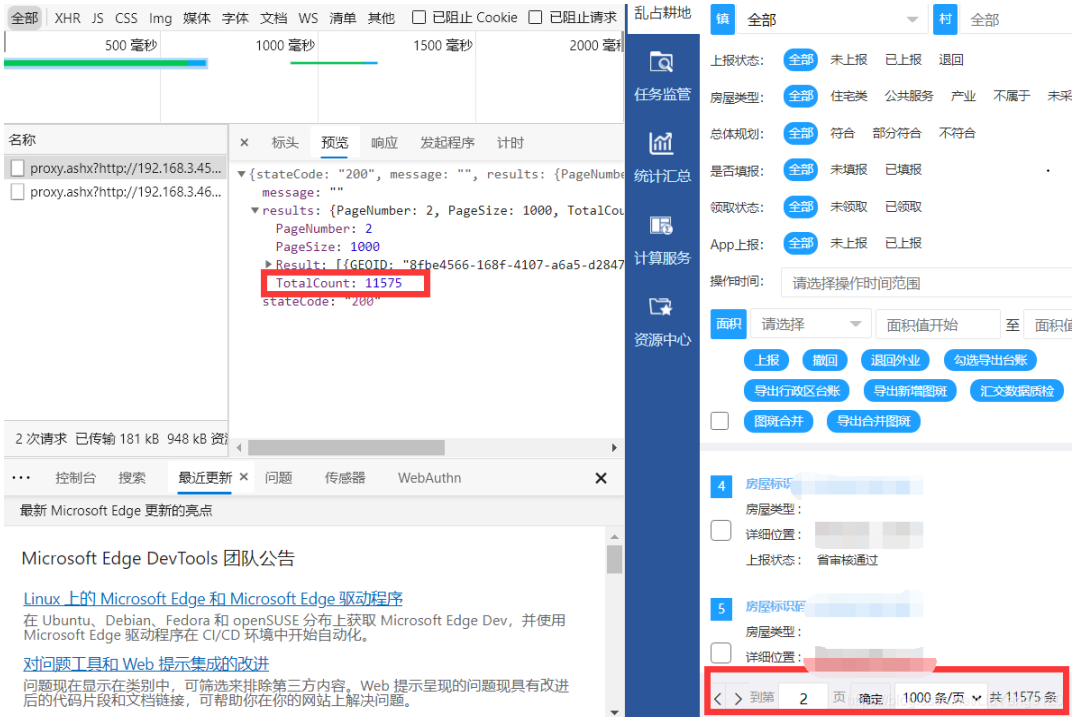

图 3：总页计算

计算出翻页数之后通过 Cloner 转换器生成需要请求的总次数（在 Python 中这一步骤一般通过循环完成），由于每一页 URL 不同之处在于 pageNumber 属性值的不同。对比每页 URL 发现 pageNumber 属性从 1 开始依次递增，则可通过 Counter 转换器生成每页的 pageNumber 属性值。模板如下图 4 所示：

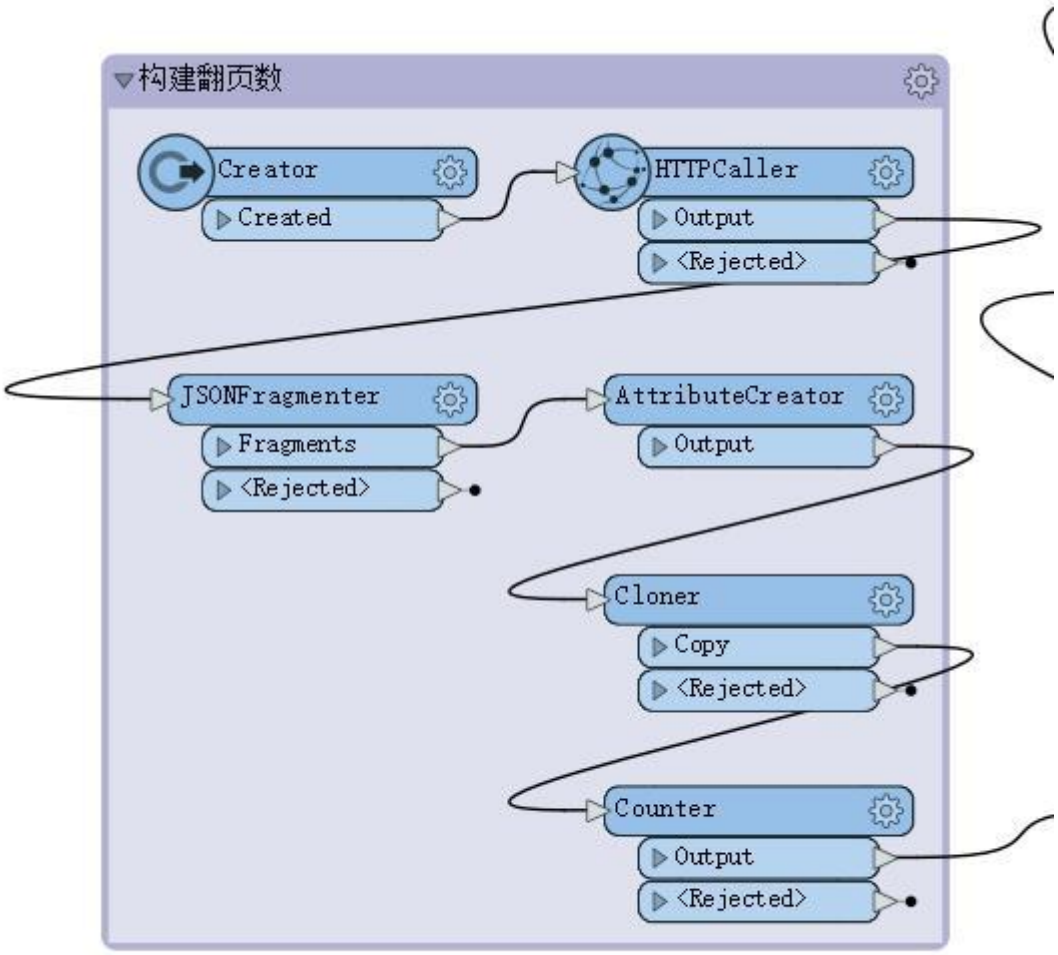

图 4：翻页计算

（2）设置随机延迟：

使用 RandomNumberGenerator 转换器生成相应区间的随机数，将生成的随机数属性作为参数传递给 Decelerator 转换器设置延迟，该步骤是为了防止对服务器请求速度过快，导致 ip 被封。设置随机延时，可以降低被封 ip 的风险。详见下图 5 和图 6：

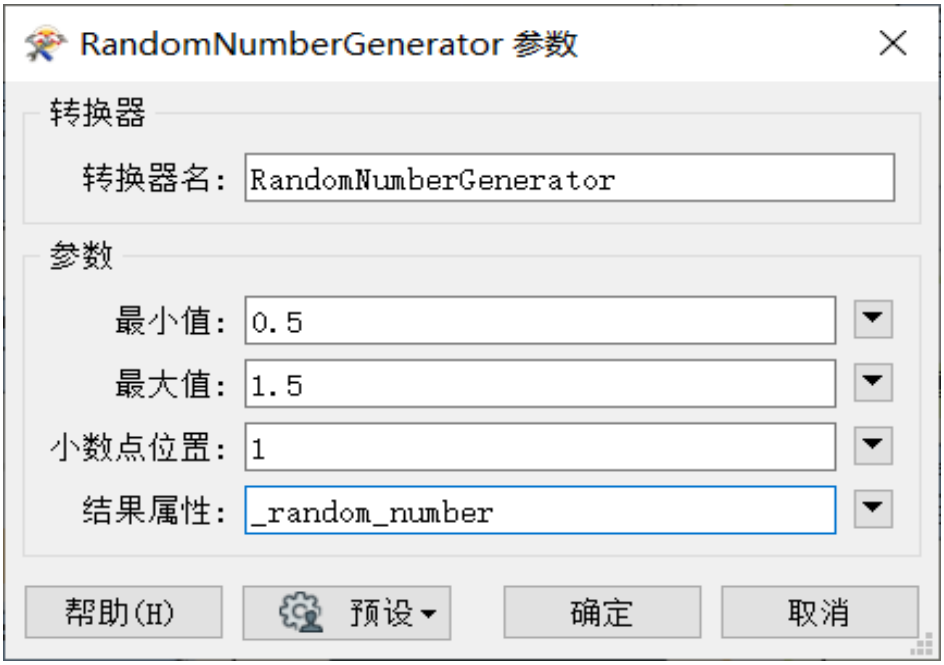

图 5：随机数参数

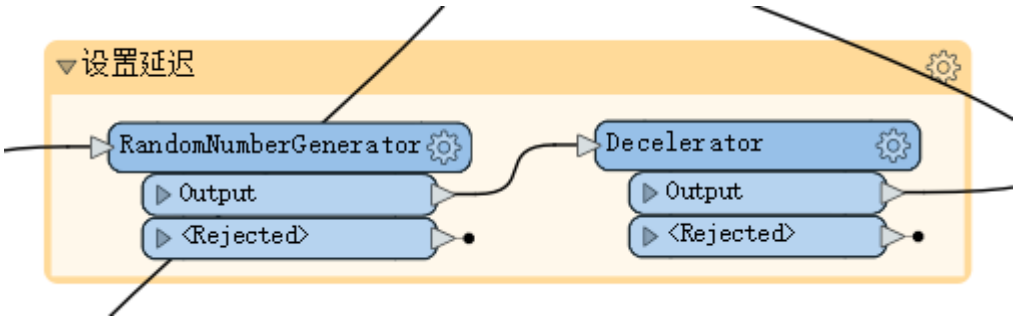

图 6：随机延迟

（3）JSON 数据解析统计：

把上方 Counter 转换器（图 4 步骤）生成的 pageNumber 属性值作为参数传递给 HTTPCaller 转换器，依次请求获取每页的 JSON 字符串数据。这里我是直接通过 PythonCaller 转换器进行解析 JSON 字符串提取所需数据（如果不熟悉 Python，FME 也提供有 JSONFragmenter、JSONFlattener 等针对于 JSON 字符串处理的转换器）。最终通过 StatisticsCalculator 转换器完成数据统计。详见下图 7 和图 8：

```python
import fme
import fmeobjects
import json

def processFeature(feature):
    pass

class FeatureProcessor(object):
    def __init__(self):
        pass
    def input(self,feature):

        j = json.loads(feature.getAttribute('_response_body'))
        j_list = j["results"]["Result"]
        for i in j_list:
            feature.setAttribute("CJXZQMC", i["CJXZQMC"])
            feature.setAttribute("CJXZQDM", i["CJXZQDM"])
            feature.setAttribute("XZXZQMC", i["XZXZQMC"])
            feature.setAttribute("GEOFWBH", i["GEOFWBH"])
            feature.setAttribute("GEOID", i["GEOID"])
            self.pyoutput(feature)
```

图 7：JSON 提取

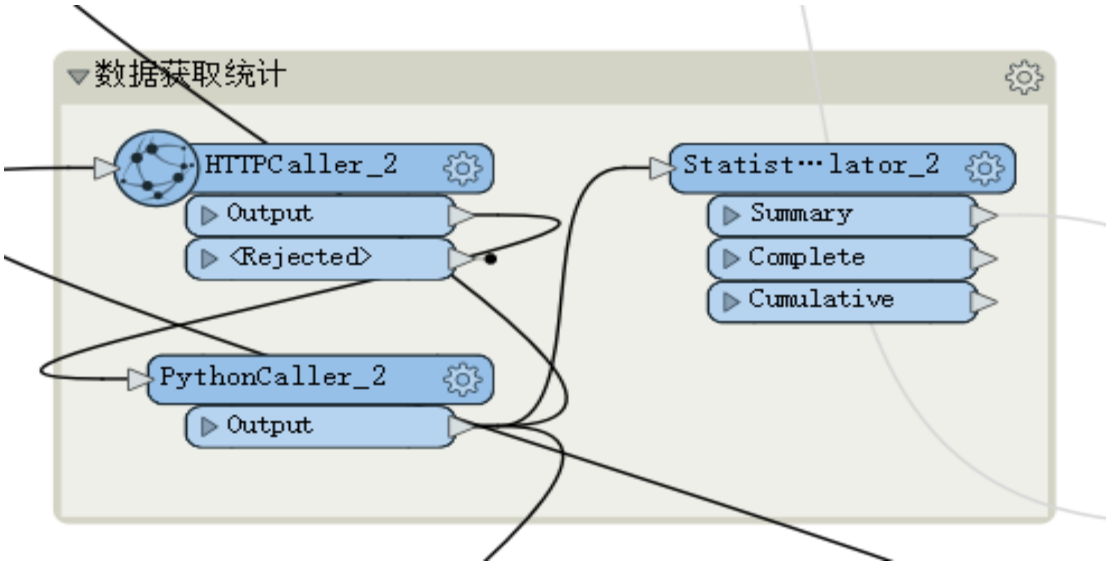

图 8：数据统计

（4）图像数据输出：

需要提取的照片如下图 9 所示，提取照片的过程主要是通过 HTTPCaller 转换器完成。首先分析不同照片的 URL 地址发现不同之处在于

GEOID 属性存在差异，而 GEOID 属性正好在之前获取的 JSON 字符串中。通过提取 JSON 字符串中的 GEOID 属性与照片 URL 地址进行动态拼接完成不同照片的获取。详见下图 10：

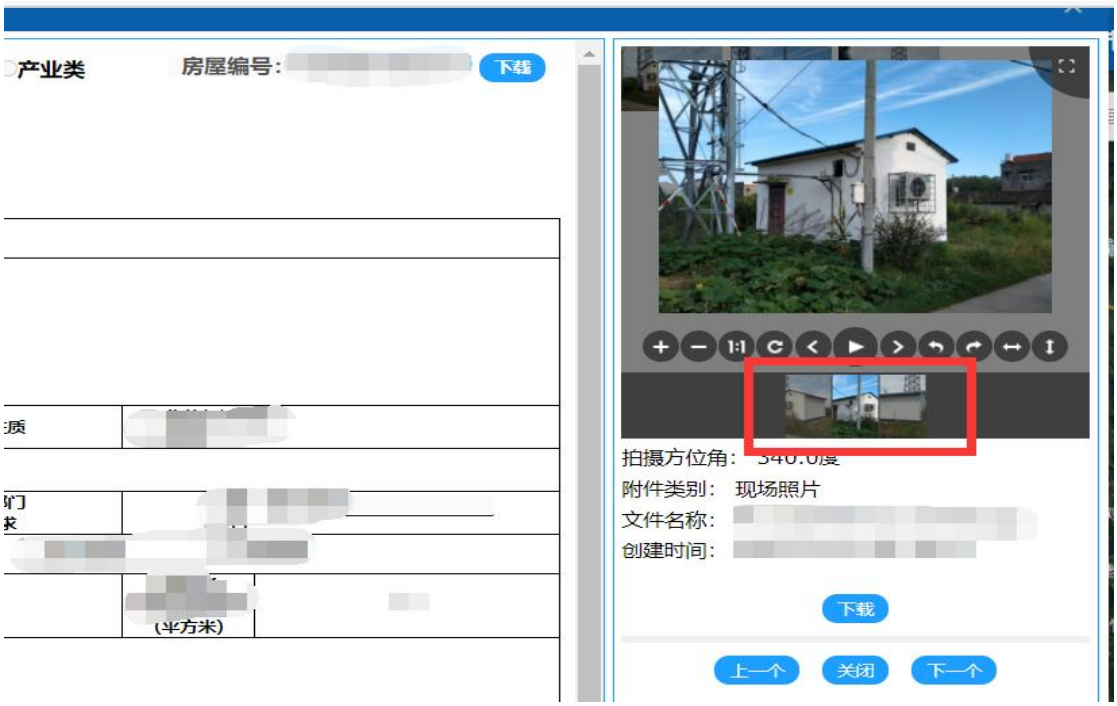

图 9：照片位置

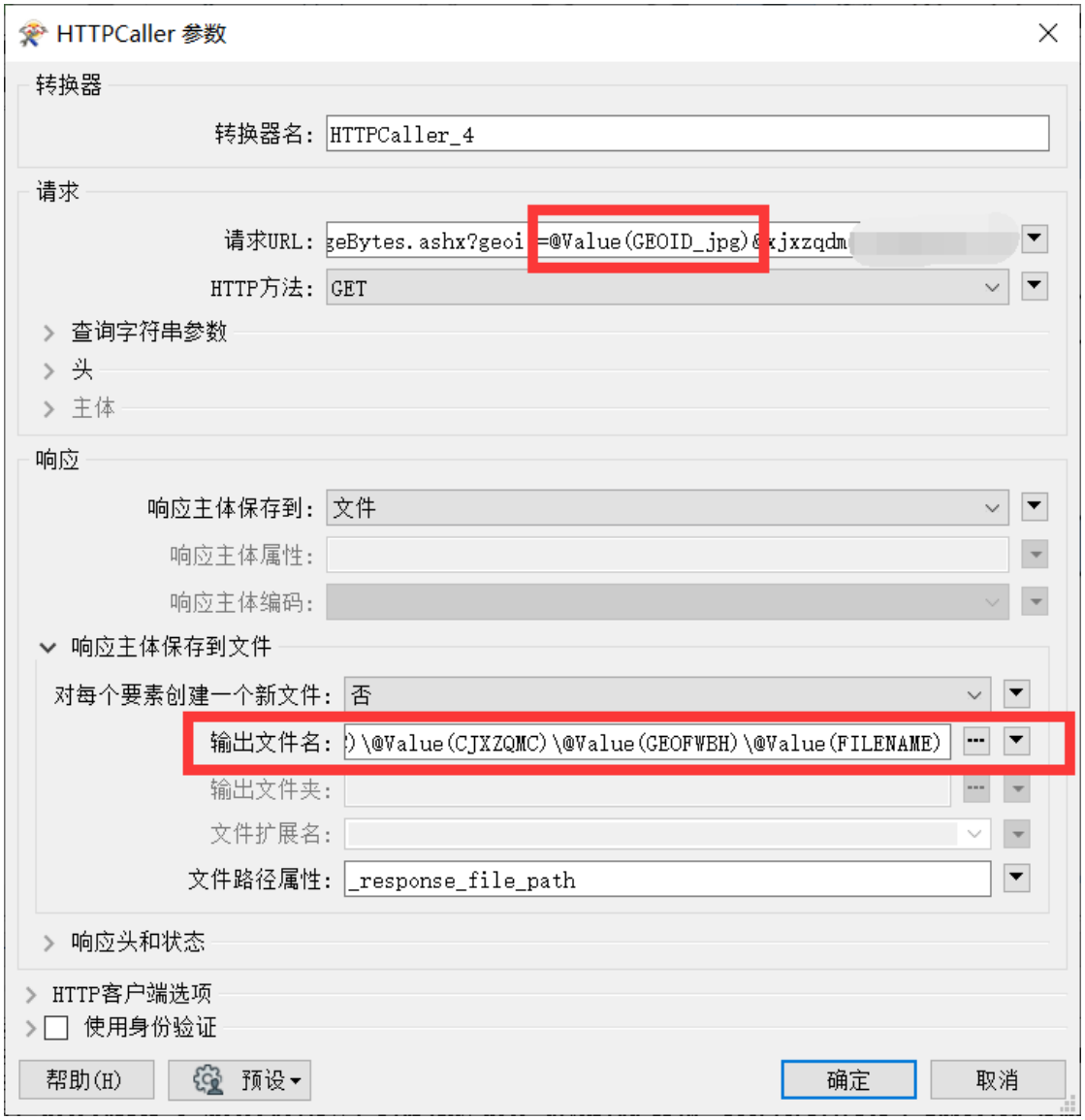

图 10：参数配置

最后实现的效果如下图 11 所示：

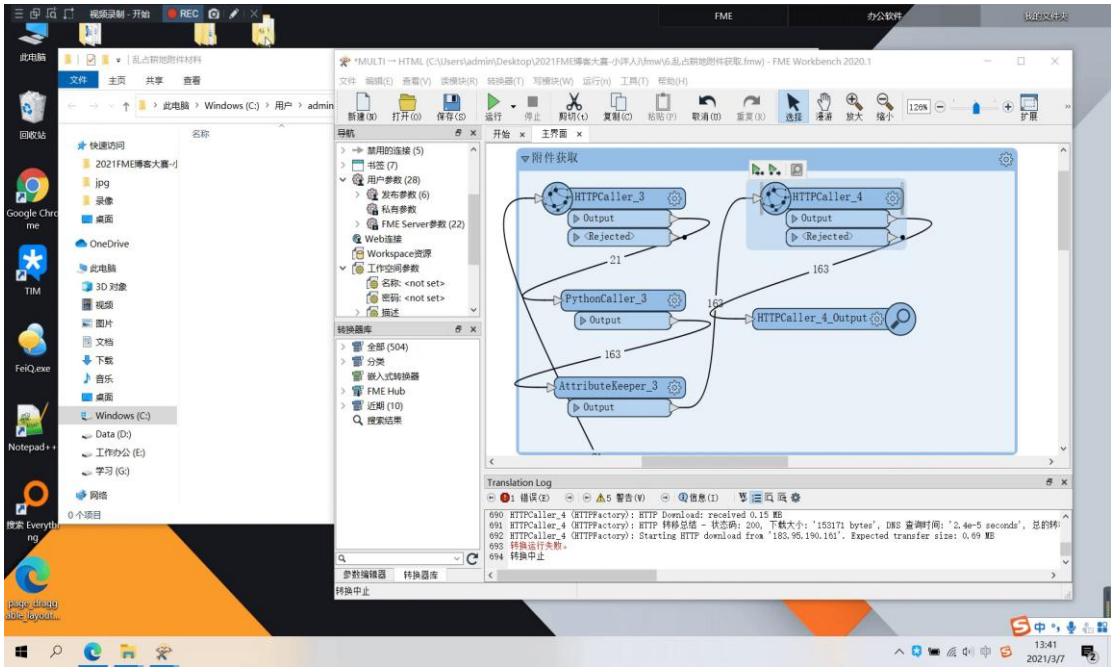

图 11：提取照片

（5）PyEcharts 可视化：

将统计的数据作为参数传递给 PythonCaller 转换器，通过 Python 强大的可视化库 PyEcharts 完成数据的可视化（PyEcharts 是一个用于生成 Echarts 图表的类库。Echarts 是百度开源的一个数据可视化 JS 库。[PyEcharts 库参考地址]）。最终生成的数据格式为 HTML 页面（如需统计其他内容，只需对传入的数据来源进行修改即可），如若将模板发布至 FME Server，定时生成此页面至服务器，即可完成一个快捷且实时监控的可视化页面。最终实现的效果如下图 12 所示：

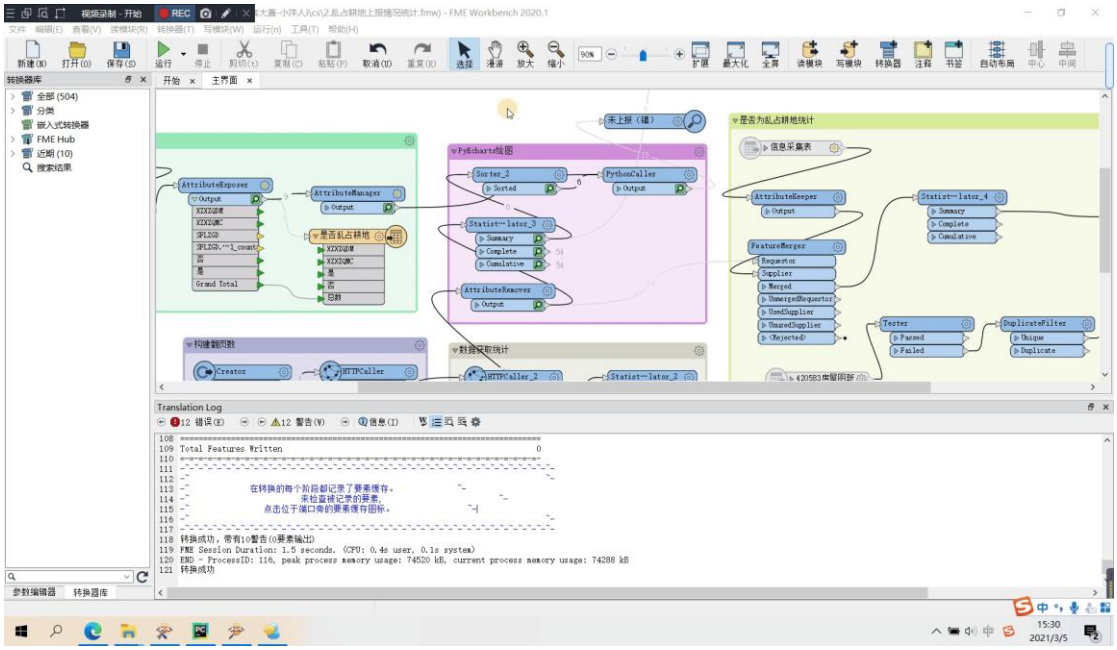

图 12：数据可视化

四、结语

整个模版最核心的地方就是 HTTPCaller 以及 PythonCaller 转换器的应用，HTTPCaller 就类似于对 Python Requests 模块工具化的封装，极大的降低了用户使用的难度以及便捷性。虽然 FME 也有相应可视化的转换器，但是效果难以达到预期。PythonCaller 的存在对于 FME 而言无疑是锦上添花，用户可以自由的通过 Python 扩展 FME 的功能，由此赋予了 FME 无限的可能。

转眼间接触 FME 已经有三四年了，在这段时间中深刻的认识到 FME 的强大之处，促使我在工作中大量的使用和学习 FME 这款软件。用户可以自由的通过 FME 以零代码的形式组建强大的 ETL 数据处理模型，通过自动化的形式完成复杂且繁琐的工作，给工作带来很多便利性。在提高工作效率的同时也保证了数据的质量。