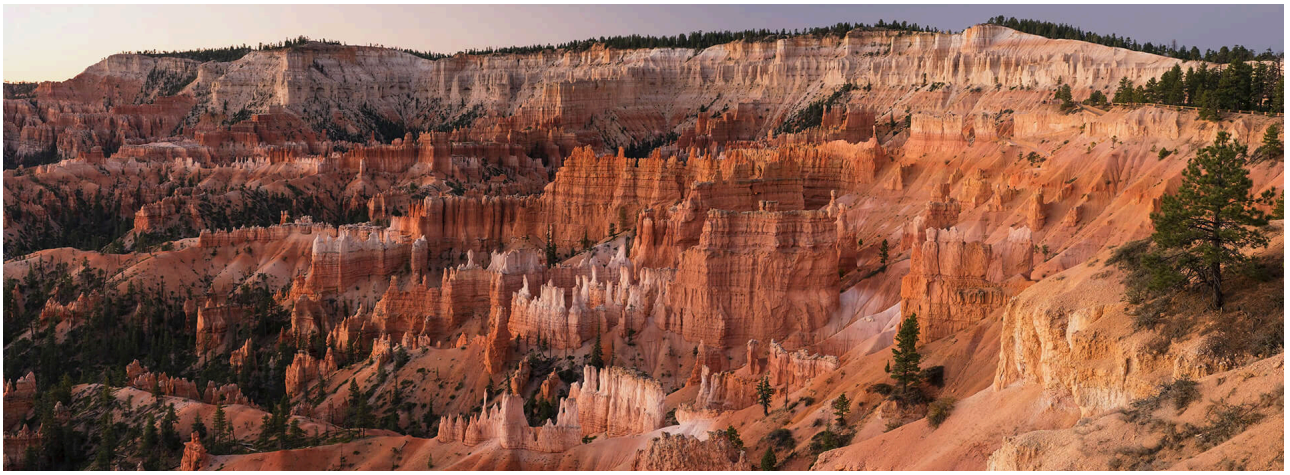


AISpider 爬虫模板生成

需求背景

1\已经存在通用爬虫采集模板,一个模板通过配置列表标题 Xpath\列表时间 Xpath\列表 a 标签 Xpath\详情页标题\详情页时间\详情页正文即可做到爬虫源采集.
2\我们希望一次性的开发完成对 1w+ 以上的招投标源的采集

最终结果:

实现了 1w+ 的爬虫模板源配置.

生产流程

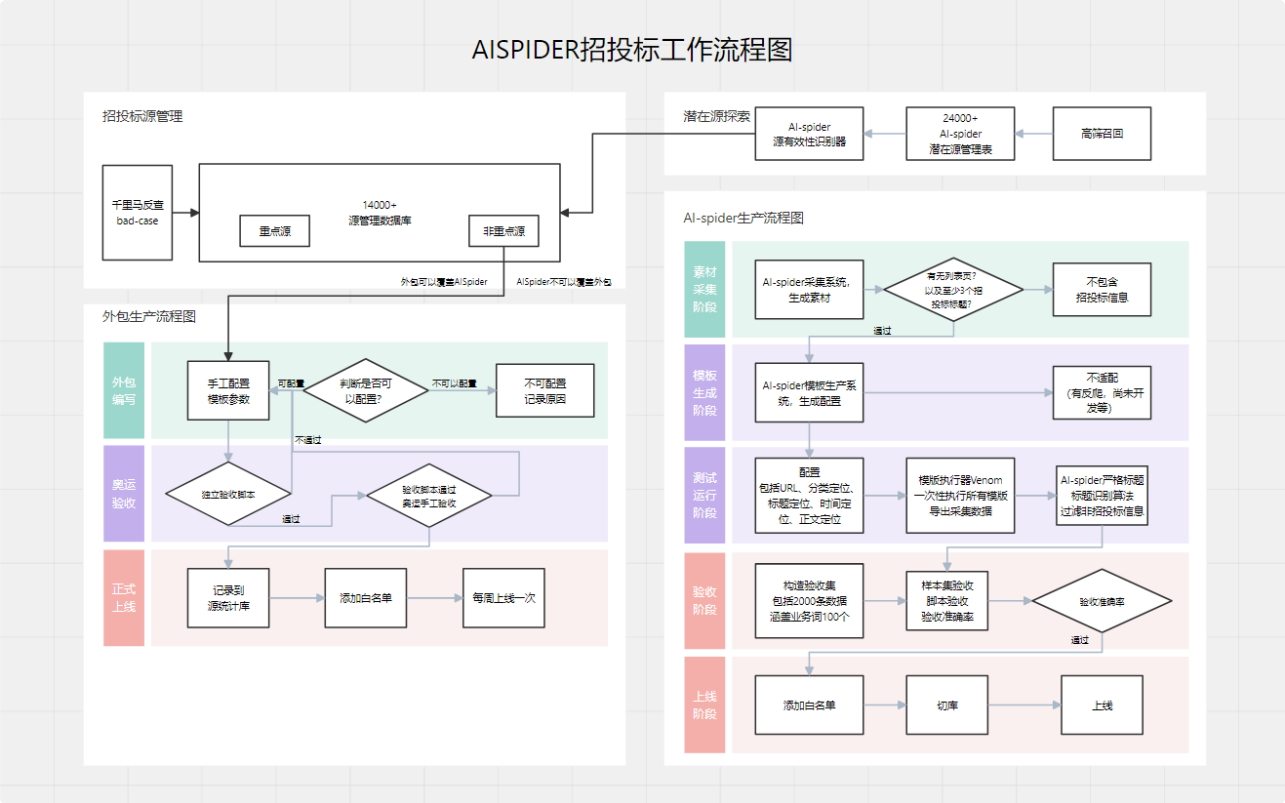
1\AISpider 源探索工具找源

2\AISpider 爬虫模板生成工具生成

3\标注人工核验准确

AISpider 招投标工作流程图

https://alidocs.dingtalk.com/i/nodes/gwva2dxOW4Kv3zYxcOjjKX3A8bkz3BRL?doc_type=wiki_draw



关键技术流程和代码解析

总纲:一句话说明模板生成器是如何运作的?

使用 webdriver 先下载网页,随后探索网页,找到**疑似招投标标题**的文本,对**该文本的共有父级标签**进行分析,如果数量大于阈值,则认为他是一个**招投标列表**.再根据其定位,**反写 Xpath**,最终生成爬虫模板

步骤一:列表页探索(spider_list_state)

核心代码:aispider_model_spider.do_driver_base.py 函数 do_work()

```
从首页进入,增加潜在任务入列表。任务表达分为两种模式，一个是url访问模式(type:url)，一个是父URL+对象点击模
    所有页面的请求优先级完全由关键词排序决定，并没有明确深度/广度优先
    有数据召回的页面将会存入knowledge_dict
```

关键函数:

self.get_xpath_str_with_title_check() 潜在 title 标签识别器

self.get_xpath_string(ttag, tt=tree)反写 xpath 的函数

self.is_title_name(text) 实现的标题识别

self.is_list_name(text) 实现的列表标题识别

步骤二:详情页采集(spider_detail_state)

核心代码:aispider_model_spider.do_driver_base.py 函数 do_work2()

```
纯文本

从首页进入,增加潜在任务入列表。任务表达分为两种模式，一个是url访问模式(type:url)，一个是父URL+关
所有页面的请求优先级完全由关键词排序决定，并没有明确深度/广度优先
有数据召回的页面将会存入knowledge_dict
```

关键函数:

MitmRun(self.mitmfile_path).start() 这个本是 AISpider 用以处理动态模板所做的尝试,可以通过中间人对所有包进行抓取,然后分析接口,从而使异步网页也可以动态生成结果.现在由于爬虫模板已支持动态渲染采集了,必要性不大了.

self.is_list_page() 判断是否是列表页

步骤三:模板生成(create_model_state)

核心代码:aispider_model_spider.do_driver_base.py 函数 do_work3()

关键函数:

get_html_list_msg_list() 解析 HTML 格式列表页

get_json_list_msg(self, page)解析 JSON 格式列表页

启动入口

ssh://git@gitlab.tangees.com:2224/spidermen/bidding-aispider-platform.git
开发一个新生成工具，需要继承基类
aispider_model_spider/common_base.py 基类
两个实现启动类:
/do_bidding_single.py (招投标实现,完整)
/DoFaYuan.py (开庭公告实现，不完整)