普林云爬虫测试方案

（第一版）

2016 年 4 月

北京至信普林科技有限公司

普林科技
PRINCETECHS

普林云爬虫测试方案

（第一版）

文档版本号	日期	作者	审核人	说明
版本 V1.0	2016/4/5	魏云蕾		创建文档

北京至信普林科技有限公司

目　录

1 概述

软件的错误是不可避免的，所以必须经过严格的测试。通过对软件的测试，尽可能的发现软件中的错误，从而减少系统内部各模块的逻辑，功能上的缺陷和错误，保证每个模块正确地实现其预期的功能。检查和排除系统结构或相应程序结构上的错误，使所有的系统单元配合合适，整体的性能和功能完整。并且使软件的功能与用户要求保持一致。

2 测试资源和测试环境

2.1 硬件配置

关键项	数量	性能要求	期望到位阶段
测试 PC 机	10 台		
数据库服务器	暂定一台		

2.2 软件配置

资料名称/类型	配置
操作系统环境	OS X EI Capitan 版本 10.11.1
功能性测试工具	手动测试及其他
性能测试工具	
测试管理工具	MantisBT
测试用例管理工具	Testlink

2.3 测试数据

本测试方案的测试数据来源测试需求、测试用例。

3 测试策略

系统测试类型及各种测试类型所采用的方法、工具等介绍如下：

3.1 功能测试

测试范围	验证业务功能正确
测试目标	核实所有功能均已正常实现，即是否与需求一致
采用技术	主要采用黑盒测试、边界测试、等价类测试等测试方法；包括集成测试和系统测试
开始标准	开发阶段对应的功能已完成并且测试用例设计完成
完成标准	测试用例通过并且最高级缺陷全部解决
特殊项	

3.2 性能测试

测试范围	爬取速度、爬取承载力、反爬能力测试
测试目标	满足各个性能指标
采用技术	
开始标准	功能测试完成，性能测试环境及数据准备就绪
完成标准	进行各种测试，满足性能指标
特殊项	

3.3 回归测试

测试范围	所有功能、用户界面、性能等测试类型
测试目标	核实执行所有测试类型后，功能、性能等均达到用户需求所要求的标准
采用技术	黑盒测试等
开始标准	每个合适的测试阶段上进行回归测试

完成标准	各复测内容均通过
特殊项	

4 测试实施阶段

测试范围	测试阶段			
	单元测试	集成测试	系统测试	验收测试
功能测试	○	√	√	验收负责人
性能测试	○	√	√	验收负责人
UI 测试	○	√	√	验收负责人
回归测试	每当被测试的软件或环境改变时在每个合适的测试阶段上进行回归测试。√			
备注：“√”表示有测试组执行，“○”表示有开发组执行；				

5 测试通过准则

被测系统无业务逻辑错误和二级的 bug。经确定的所有缺陷都得到了商定的解决结果。所涉及的测试用例已全部重新执行，已知的所有缺陷都已按照商定的方式进行了处理，而且没有发现新的缺陷。

注：

A: 严重影响系统运行的错误；

B: 功能方面一般缺陷，影响系统运行；

C: 界面布局不完美或轻型错误；

D: 不影响运行的错别字等；

E: 合理化建议。

6 集成测试用例设计

6.1 生成器测试

模块	说明	步骤	描述	预期
生成器测试	测试用户手工输入URL功能正确	Step1	手工输入一个小于8000字符的可用的URI	库中正确存储该URI信息（MongoDB数据库中CrawlerTask中存入内容）
		Step2	手工输入一个小于8000字符的非法的URI	库中不存储该URI的信息，并且将错误的URI写入错误日志中去
		Step3	手工输入一个大于8000字符的URI	库中不存储该URI的信息，并且将错误的URI写入错误日志中去
		Step4	重复输入同一个URI	该URI不会重复存储(在防重机制时间范围内，并且该URI的status为on）
		Step5	同时输入几个URI，每个URI使用回车做分隔符	输入的所有URI被存到数据库中
		Step6	同时输入几个URI，其中有已回车做分隔符，有的以逗号等做分隔符	回车做分隔符的（即一行一个的）成功添加到库中，一行几个（其他字符做分割符时），添加失败，在错误日志中输出

	测试导入 csv 文件功能正确	Step1	导入仅包含可用 URI 的字符串，字符串以回车做分隔符	所有 URI 成功的存 MongoDB 库中（MongoDB 数据库中 CrawlerTask 中存入内容）
		Step2	导入仅包含 URI，并且个别 URI 为非法的（格式不正确等），字符串以回车符做分隔符	正确的 URI 成功存入到 MongoDB 里，错误的 URI 存储在错误日志中
		Step3	导入仅包含 URI，并且有重复的 URI 文件，字符串以回车符做分隔符	重复的 URI 不会重复导入（在防重机制时间范围内，并且该 URI 的 status 为 on）
		Step4	导入仅包含 URI，并且个别 URI 为非法的（超过 8000 字符），字符串以回车符做分隔符	正确的 URI 成功存入到 MongoDB 里，错误的 URI 存储在错误日志中
		Step5	导入仅包含可用 URI 的字符串，字符串以回车做分隔符,其中有个别行一行有两个以上 URI，并且用逗号做分隔符	其他符合格式的成功导入到 MogoDB 中，不符合格式的导入不成功，并存储在错误日志中
	测试导入 txt 文件功能正确	Step1	导入仅包含可用 URI 的字符串，字符串以回车做分隔符	所有 URI 成功的存 MongoDB 库中（MongoDB 数据库中 CrawlerTask 中存入内容）
		Step2	导入仅包含 URI，并且个别 URI 为非法的（格式不正确等），字符串以回车符做分隔符	正确的 URI 成功存入到 MongoDB 里，错误的 URI 存储在错误日志中

		Step3	导入仅包含 URI，并且有重复的 URI 文件，字符串以回车符做分隔符	重复的 URI 不会重复导入（在防重机制时间范围内，并且该 URI 的 status 为 on）
		Step4	导入仅包含 URI，并且个别 URI 为非法的（超过 8000 字符），字符串以回车符做分隔符	正确的 URI 成功存入到 MongoDB 里，错误的 URI 存储在错误日志中
		Step5	导入仅包含可用 URI 的字符串，字符串以回车做分隔符,其中有个别行一行有两个以上 URI，并且用逗号做分隔符	其他符合格式的成功导入到 MogoDB 中，不符合格式的导入不成功，并存储在错误日志中
		Step6	导入 doc、xlsx 等非 cvs，txt 后缀名结尾的文件	系统输出错误日志，文件类型不对
	测试导入python 脚本生产 URI 功能正确	Step1	上传工商信息网 URL Python 脚本，执行生成器	脚本输出的 URL 符合格式规则（省份等对应正确），并将脚本输出的所有 URI 存到对应的 mongoDB 中 （MongoDB 数据库中 CrawlerTaskGenerator 中存入内容）
		Step2	上传百度+搜索的 URL Python 脚本，执行生产器	脚本输出的 URL 符合格式规则，并将脚本输出的所有 URI 存到对应的 mongoDB 中
		Step3	上传人民法院公告 URL Python 脚本，执行生产器	脚本输出的 URL 符合格式规则，并将脚本输出

				的所有 URI 存到对应的 mongoDB 中
	测试导入 shell 脚本生产 URI 功能正确	Step1	上传工商信息网 URL Python 脚本，执行生成器	脚本输出的 URL 符合 32 个站点格式规则，并将脚本输出的所有 URI 存到对应的 mongoDB 中（MongoDB 数据库中 CrawlerTaskGenerator 中存入内容）
		Step2	上传百度+搜索的 URL Python 脚本，执行生产器	脚本输出的 URL 符合格式规则，并将脚本输出的所有 URI 存到对应的 mongoDB 中
		Step3	上传人民法院公告 URL Python 脚本，执行生产器	脚本输出的 URL 符合格式规则，并将脚本输出的所有 URI 存到对应的 mongoDB 中
	测试生成器定时器系统功能正确	Step1	指定生成器间隔 5 分钟执行一次	生成器按照定时器设置执行，有相同的不重复添加（在防重机制时间范围内，并且该 URI 的 status 为 on）
		Step2	指定生成器 1 小时执行一次	生成器按照定时器设置执行，并修改已变动的 URL，没有变动的不重复添加（在防重机制时间

				范围内，并且该 URI 的 status 为 on）
		Step3	指定未执行过的生成器a 当前日期的下午三点执行	生成器在指定时间被执行，并修改已变动
		Step4	指定生成器 a，当前日期的下午 5 点执行	生成器在指定时间被执行，并修改已变动的 URL，没有变动的不重复添加

6.2 下载器测试

模块	说明	步骤	描述	预期
下载器测试	验证工商信息下载器正确	Step1	新建 job 任务，手工添加工商信息网的某公司的 URL，指定工商信息爬取对应下载器	该 URL 信息被存储在 MongoDB 中
		Step2	启动该 job 下的下载器，进行数据爬取	爬取的数据存到 MongoDB 中，与页面数据一致，无缺少重复不正确信息
		Step3	导入 csv 文件（库中不存在的 32 条 URI 不同省份的）	该文件中 URL 存储在 MongoDB 中
		Step4	重新启动下载器，进行该 job 下所有 URI，数据爬取	爬取的数据存储到 MongoDB 中， 并且已存在的爬取数据不重复添加（在防重机制时间范围内）。 数据与页面数据一致，无缺少重复不正

				确信息
		Step5	导入 txt 文件（库中不存在的 100 条 URI）	该文件中 URL 存储在 MongoDB 中
		Step6	重新启动下载器，进行该 job 下所有 URI，数据爬取	爬取的数据存储到 MongoDB 中， 并且已存在的爬取数据不重复添加（在防重机制时间范围内）。 数据与页面数据一致，无缺少重复不正确信息
		Step7	执行 python 生成器脚本，导入（500 条库中不存在的 URI）	生产的 URI 正确存储在 MongoDB 中
		Step8	重新启动下载器，进行该 job 下多有 URI，数据爬取	爬取的数据存储到 MongoDB 中， 并且已存在的爬取数据不重复添加（在防重机制时间范围内）。 数据与页面数据一致，无缺少重复不正确信息
		Step9	执行 python 生成器脚本，导入（1 万条库中不存在的 URI）	爬取的数据存储到 MongoDB 中， 并且已存在的爬取数据不重复添加（在防重机制时间范围内）。 数据与页面数据一

				致，无缺少重复不正确信息 （爬取出错的 URL 存储在错误日志中）
	验证百度+关键词下载器正确	Step1	新建 job 任务，手工添加某公司+违约关键词的百度 URI，指定工商信息爬取对应下载器	该 URL 信息被存储在 MongoDB 中
		Step2	启动该 job 下的下载器，进行数据爬取	爬取的数据存到 MongoDB 中，与页面数据一致，无缺少重复不正确信息（在防重机制时间范围内）
		Step3	导入 csv 文件（库中不存在的 10 条 URI）	该文件中 URL 存储在 MongoDB 中
		Step4	重新启动下载器，进行该 job 下所有 URI，数据爬取	爬取的数据存储到 MongoDB 中， 并且已存在的爬取数据不重复添加。数据与页面数据一致，无缺少重复不正确信息（在防重机制时间范围内）
		Step5	导入 txt 文件（库中不存在的 100 条 URI）	该文件中 URL 存储在 MongoDB 中
		Step6	重新启动下载器，进行该 job 下所有 URI，数据爬取	爬取的数据存储到 MongoDB 中， 并且已存在的爬取数

				据不重复添加。数据与页面数据一致，无缺少重复不正确信息（在防重机制时间范围内）
		Step7	执行 python 生成器脚本，导入（500 条库中不存在的 URI）	生产的 URI 正确存储在 MongoDB 中
		Step8	重新启动下载器，进行该 job 下多有 URI，数据爬取	爬取的数据存储到 MongoDB 中， 并且已存在的爬取数据不重复添加。数据与页面数据一致，无缺少重复不正确信息（在防重机制时间范围内）
		Step9	执行 python 生成器脚本，导入（1 万条库中不存在的 URI）	爬取的数据存储到 MongoDB 中， 并且已存在的爬取数据不重复添加。数据与页面数据一致，无缺少重复不正确信息（在防重机制时间范围内） （爬取出错的 URL 存储在错误日志中）
	验证人民法院公告网下载器正确	Step1	新建 job 任务，手工添加人民法院公告网某公司的 URI，指定工	该 URL 信息被存储在 MongoDB 中

			商信息爬取对应下载器	
		Step2	启动该 job 下的下载器，进行数据爬取	爬取的数据存到 MongoDB 中,与页面数据一致，无缺少重复不正确信息（在防重机制时间范围内）
		Step3	导入 csv 文件(库中不存在的 10 条 URI）	该文件中 URL 存储在 MongoDB 中
		Step4	重新启动下载器，进行该 job 下所有 URI，数据爬取	爬取的数据存储到 MongoDB 中， 并且已存在的爬取数据不重复添加。数据与页面数据一致，无缺少重复不正确信息（在防重机制时间范围内）
		Step5	导入 txt 文件（库中不存在的 100 条 URI）	该文件中 URL 存储在 MongoDB 中
		Step6	重新启动下载器，进行该 job 下所有 URI，数据爬取	爬取的数据存储到 MongoDB 中， 并且已存在的爬取数据不重复添加。数据与页面数据一致，无缺少重复不正确信息（在防重机制时间范围内）
		Step7	执行 python 生成器脚本，导入	生产的 URI 正确存储

			（500 条库中不存在的 URI）	在 MongoDB 中
		Step8	重新启动下载器，进行该 job 下多有 URI，数据爬取	爬取的数据存储到 MongoDB 中， 并且已存在的爬取数据不重复添加。数据与页面数据一致，无缺少重复不正确信息（在防重机制时间范围内）
		Step9	执行 python 生成器脚本，导入（1 万条库中不存在的 URI）	爬取的数据存储到 MongoDB 中， 并且已存在的爬取数据不重复添加。数据与页面数据一致，无缺少重复不正确信息（在防重机制时间范围内） （爬取出错的 URL 存储在错误日志中）

6.3 解析器测试

模块	说明	步骤	描述	预期
解析器测试	验证解析器解析工商爬取数据功能正确	Step1	新建 job 任务爬取工商信息，配置生成器、下载器、解析器（指定对应格式的标准解析器 Json、Html、PDF、Image、Binary），启动完成生成、下载	生成的不同公司 URI 存储在 MongoDB 中，爬取的数据正确存储在 MongoDB 中。

			任务	
		Step2	配置解析器（指定需要提取的关键字及存储的mysql数据库以及对应表）	配置表保存配置结果
		Step3	执行该 job 下的解析器	解析好的数据存储在配置的 mysql 对应的数据库中，结构与配置一致。 （解析中出错的信息，保存在错误日志中，最后统一回收，根据重启策略重启，如果重启策略失效，生产报警日志结束解析）
		Step4	检查 mysql 数据中的数据	Mysql 中存储的数据与 mongoDB 中数据一致，并与爬取的页面数据一致
		Step5	修改mongoDB中存储的部分源数据	部分源数据被修改
		Step6	再次启动该 job 下的解析器	未改变的数据不再被解析重复存储（在防重机制时间范围内），数据改变的进行更新
	验证解析器解析百度+关键字 爬取数据功	Step1	新建 job任务爬取百度+关键字，配置生成器、下载器、解析器(指	生成的不同公司 URI 存储在 MongoDB 中，

	能正确		定对应格式的标准解析器Json、Html、PDF、Image、Binary)，启动完成生产、下载任务	爬取的数据正确存储在MongoDB中。
		Step2	配置解析器（指定需要提取的关键字及存储的mysql数据库以及对应表）	配置表保存配置结果
		Step3	执行该job下的解析器	解析好的数据存储在配置的mysql对应的数据库中，结构与配置一致。 （解析中出错的信息，保存在错误日志中，最后统一回收，根据重启策略重启，如果重启策略失效，生产报警日志结束解析）
		Step4	检查mysql数据中的数据	Mysql中存储的数据与mongoDB中数据一致，并与爬取的页面数据一致
		Step5	修改mongoDB中存储的部分源数据	部分源数据被修改
		Step6	再次启动该job下的解析器	未改变的数据不再被解析重复存储（在防重机制时间范围内),

				数据改变的进行更新
	验证解析器解析人民法院公告网 爬取数据功能正确	Step1	新建 job 任务爬取人民法院公告网数据，配置生成器、下载器、解析器（指定对应格式的标准解析器 Json、Html、PDF、Image、Binary），启动完成生产、下载任务	生成的不同公司 URI 存储在 MongoDB 中，爬取的数据正确存储在 MongoDB 中。
		Step2	配置解析器（指定需要提取的关键字及存储的mysql 数据库以及对应表）	配置表保存配置结果
		Step3	执行该 job 下的解析器	解析好的数据存储在配置的 mysql 对应的数据库中，结构与配置一致。 （解析中出错的信息，保存在错误日志中，最后统一回收，根据重启策略重启，如果重启策略失效，生产报警日志结束解析）
		Step4	检查 mysql 数据中的数据	Mysql 中存储的数据与 mongoDB 中数据一致，并与爬取的页面数据一致
		Step5	修改mongoDB 中存储的部分源数据	部分源数据被修改

		Step6	再次启动该 job 下的解析器	未改变的数据不再被解析重复存储（在防重机制时间范围内），数据改变的进行更新

6.4 IP 代理测试

模块	说明	步骤	描述	预期
IP 代理测试	验证 IP 代理抓取功能正确	Step1	输入 IP 代理网址，进行页面 IP 数据爬取	页面显示数据和库中存储的数据一致。（id：主键自增 ip_port： 唯一，ip:port province： 省份，没有为 other is_valid ： 是否可用 bool）
		Step2	重复爬取同一个代理页面的 IP 信息	重复的ip信息不回添加到库中，若可用性改变，则及时更新
		Step3	爬取不同代理页面的 IP 信息	爬取的IP信息与页面显示一致
	验证 IP 有效性判断机制正确	Step1	通过验证接口，输入被测 IP 及验证 URL（www.baidu.com）	返回结果与库中结果一致
		Step2	输入同一被测 IP，输入不同的验证链接（www.sogou.com/www.jd.com）等	查看返回结果是否一致

		Step3	选择不同 IP 地址，进行验证	查看返回结果是否和库中存储一致
	验证请求分配 IP 接口功能正确	Step1	通过请求 IP 接口,输入请求数量，地区	接口返回请求地区的 IP，数量与请求的数量一致，并且可用性为 ture
		Step2	不输入数量，地区等参数	系统默认返回指定数量的 ip 地址;（数量待定） 该返回的ip可用性为true，但地区不定
		Step3	只有请求数量，不请求地区	系统返回ip地址与请求的数量一致，地区不定
		Step4	只有请求地区，不请求请求	系统返回ip地址与请求的地区一致，数量为系统默认
	验证 IP 轮询机制功能正确	Step1	执行对库中 IP 有效性轮询机制	库中 IP 信息有效性进行更新
		Step2	通过有效性验证接口，验证 URL（www.baidu.com）,验证库中有效性被更改的 IP 地址	和更新后的 IP 有效性一致
		Step3	通过有效性验证接口，验证 URL（ www.sogou.com/www.jd.com 等),验证库中有效性被更改的 IP 地址	和更新后的 IP 有效性一致
		Step4	通过有效性验证接口，验证 URL	结果与库中存储一致

			（www.baidu.com），验证库中有效性没被修改的 IP	
		Step5	执行对代理 IP 页面爬取定时爬取机制	库中已存在的IP不进行添加，只更新可用性， 添加库中没有的IP信息 页面中的所有 IP，在库中都已存储，并且标记可用性，信息正确
		Step6	通过有效性验证接口，验证 URL（www.baidu.com），验证可用性被更新的 IP	和更新后的IP有效性一致
		Step7	通过有效性验证接口，验证 URL（ www.sogou.com/www.jd.com 等），验证可用性被更新的 IP	和更新后的IP有效性一致
		Step8	通过有效性验证接口，验证 URL（www.baidu.com），验证新添加的 IP 地址的有效性	结果与库中结果一致
		Step9	通过有效性验证接口，验证（ www.sogou.com/www.jd.com 等），验证新添加的 IP 地址的有效性	结果与库中结果一致
		Step10	通过有效性验证接口，验证 URL（www.baidu.com），验证库中有效性没被修改的 IP	结果与库中存储一致

6.5 验证码破解测试

模块	说明	步骤	描述	预期
验证码破解测试	验证数字和字母验证码破解功能有效	Step1	输入1个数字或字母验证码图片	输出正确的数字和字母，并且顺序正确
		Step2	重复输入不同的数字或字母验证码图片	输出正确的数字和字母，并且顺序正确
	验证汉字成语验证码破解功能有效	Step1	输入1个汉字成语验证码图片	输出正确的汉字成语，汉字顺序正确
		Step2	重复输入不同的汉字成语验证码图片	输出正确的汉字成语，汉字顺序正确
	验证四则运算验证码破解功能有效	Step1	输入1个四则运算验证码图片	输出四则运算结果正确
		Step2	重复输入不同的四则运算验证码图片	输入四则运算结果正确
	验证带九宫格的汉字验证码功能有效	Step1	输入1个九宫格汉字验证码图片	输出汉字正确，顺序正确
		Step2	重复输入不同的九宫格汉字验证码图片	输出汉字正确，顺序正确
		Step3	输入不是以上四种验证码类型图片	系统报错，将错误信息写入错误日志
		Step4	重复输入四种不同的验证码图片	验证成功的输出正确结果； 验证失败的，将失败日志写入错误日志中

6.6 自然语言处理（NLR）测试

模块	说明	步骤	描述	预期
自然语言处理（NLR）测试	验证自然语言处理功能正确	Step1	设置需要提取的关键信息为公司名称、违约时间、违约类型、违约事件	设置结果保存成功
		Step2	输入包含该关键词信息的非结构化的信息	输出提取内容，准确，没有错误
		Step3	输入包含部分该关键词信息的非结构化的信息	输出提取内容，内容对应正确，没有对应内容的显示为空
		Step4	输入不不含该关键词信息的非结构化的信息	输入提取内容为空
		Step5	设置需要提取的关键信息为被告（债务人）、破产时间、破产事件、受理法院	设置结果保存成功
		Step6	输入包含该关键词信息的非结构化的信息	输出提取内容，准确，没有错误
		Step7	输入包含部分该关键词信息的非结构化的信息	输出提取内容，内容对应正确，没有对应内容的显示为空
		Step8	输入不不含该关键词信息的非结构化的信息	输入提取内容为空
		Step9	设置需要提取的关键信息为姓名、年龄、学历、工作经验	设置结果保存成功
		Step10	输入包含该关键词信息的非结构化的信息	输出提取内容，准确，没有错误

		Step11	输入包含部分该关键词信息的非结构化的信息	输出提取内容，内容对应正确，没有对应内容的显示为空
		Step12	输入不含该关键词信息的非结构化的信息	输入提取内容为空
		Step13	设置需要提取的关键信息为其他任何需要提取的内容	设置结果保存成功
		Step14	输入包含该关键词信息的非结构化的信息	输出提取内容，准确，没有错误
		Step15	输入包含部分该关键词信息的非结构化的信息	输出提取内容，内容对应正确，没有对应内容的显示为空
		Step16	输入不不含该关键词信息的非结构化的信息	输入提取内容为空

6.7 图片文件结构化（OCR）测试

模块	说明	步骤	描述	预期
图片文件结构化(OCR)测试	验证图片文件结构化测试功能正确	Step1	输入不同格式图片，图像中只包含简体中文，颜色为黑白	输出文本内容，并和图片中显示文字基本一致
		Step2	输入不同格式图片，图像中只包含简体中文，颜色为彩色	输出文本内容，并和图片中显示文字基本一致
		Step3	输入不同格式图片，图像中只包含英文，颜色为黑白	输出文本内容，并和图片中显示文字基本一致

		Step4	输入不同格式图片，图像中只包含繁体，颜色为黑白	输出文本内容，并和图片中显示文字基本一致
		Step5	输入不同格式图片，图像中包含简体中文、英文、繁体中文，颜色为黑白	输出文本内容，并和图片中显示文字基本一致
		Step6	输入图像，图像文本内容为竖排	输出文本内容，并和图片中显示文字基本一致
		Step7	输入图像，图像文本为横排，有分栏	输出文本内容，并和图片中显示文字基本一致
		Step8	输入图像，图像文本为竖排，有分栏	输出文本内容，并和图片中显示文字基本一致
		Step9	输入图像，图像内容为 图文混排	输出文本内容，并和图片中显示文字基本一致
		Step10	输入相同分辨率的图像，图像中文本内容的字体大小分别为不同大小的字体	输出文本内容，并和图片中显示文字基本一致
		Step11	输入相同字体大小的图像，但图像的分辨率不同，	输出文本内容，并和图片中显示文字基本一致

7 系统测试用例设计

模块	说明	步骤	描述	预期
百度+关键词数据爬取测试	验证百度+关键词数据爬取所有工作机制正确	Step1	新建 job 任务爬取公司名称+关键字（违约）， 配置生成器（生成 500 条 URI），下载器，解析器（配置需要提取的字段等），自然语言解析器、OCR	所有配置信息正确存储
		Step2	启动任务，进行数据爬取工作	1. 生成的URI 正确的存储在mongoDB 中； 2. 爬取的源数据正确存储在mongoDB 中； 3. 解析后的数据正确存储在 mysql 中，且与页面显示一致； 4. 生成器、下载器、解析器执行过程中有报错，将信息保存在错误日志中 5. 爬取失败的根据重启策略进行重启，重新爬取，重启策略机制失效的，输出到警告日志中
		Step3	重新启动 job 任务	1. 生成器:URI 方式变化的进行剩下，没有变化的不重复存储； 2. 下载器:爬取源数据发生变化的进行更新，没变化的不重复存储； 3. 解析器:未改变的数据不再被解析重复存储，数据改变的进行更新
		Step4	重启启动 job 任务（添加 5000 条 URL）	1. 生成的URI 正确的存储在mongoDB 中；

				2. 爬取的源数据正确存储在mongoDB中； 3. 解析后的数据正确存储在 mysql中，且与页面显示一致； 4. 生成器、下载器、解析器执行过程中有报错，将信息保存在错误日志中 5. 生成器:URI方式变化的进行剩下，没有变化的不重复存储； 6. 下载器:爬取源数据发生变化的进行更新，没变化的不重复存储； 7. 解析器:未改变的数据不再被解析重复存储，数据改变的进行更新
工商信息数据爬取测试	验证工商信息数据爬取所有工作机制正确	Step1	配置生成器（生成 500 条URI)，下载器，解析器（配置需要提取的字段等),IP 代理，验证码破解器、自然语言解析器、OCR	所有配置信息正确存储
		Step2	启动任务，进行数据爬取工作	1. 生成的URI 正确的存储在mongoDB中； 2. 爬取的源数据正确存储在mongoDB中； 3. 解析后的数据正确存储在 mysql中，且与页面显示一致； 4. 生成器、下载器、解析器执行过程中有报错，将信息保存在错误日志中

				5. 爬取失败的根据重启策略进行重启，重新爬取，重启策略机制失效的，输出到警告日志中
		Step3	重新启动 job 任务	1. 生成器:URI方式变化的进行剩下，没有变化的不重复存储； 2. 下载器:爬取源数据发生变化的进行更新，没变化的不重复存储； 3. 解析器:未改变的数据不再被解析重复存储，数据改变的进行更新
		Step4	重启启动 job 任务(添加 5000 条 URL)	1. 生成的URI 正确的存储在mongoDB中； 2. 爬取的源数据正确存储在mongoDB中； 3. 解析后的数据正确存储在 mysql 中，且与页面显示一致； 4. 生成器、下载器、解析器执行过程中有报错,将信息保存在错误日志中 5. 生成器:URI方式变化的进行剩下，没有变化的不重复存储； 6. 下载器:爬取源数据发生变化的进行更新，没变化的不重复存储； 7. 解析器:未改变的数据不再被解析重复存储，数据改变的进行更新
人民法院数据爬取测	验证人民法院数据爬取所有	Step1	配置生成器（生成 500 条 URI)，下载器，解析器（配置需要提取的字段等),IP 代理，	所有配置信息正确存储

试	工作机制正确		验证码破解器、自然语言解析器、OCR	
		Step2	启动任务，进行数据爬取工作	1. 生成的URI正确的存储在mongoDB中； 2. 爬取的源数据正确存储在mongoDB中； 3. 解析后的数据正确存储在 mysql中，且与页面显示一致； 4. 生成器、下载器、解析器执行过程中有报错，将信息保存在错误日志中 爬取失败的根据重启策略进行重启，重新爬取，重启策略机制失效的，输出到警告日志中
		Step3	重新启动 job 任务	1. 生成器:URI方式变化的进行剩下，没有变化的不重复存储； 2. 下载器:爬取源数据发生变化的进行更新，没变化的不重复存储； 3. 解析器:未改变的数据不再被解析重复存储，数据改变的进行更新
		Step4	重启启动 job 任务（添加 5000 条 URL）	1. 生成的URI正确的存储在mongoDB中； 2. 爬取的源数据正确存储在mongoDB中； 3. 解析后的数据正确存储在 mysql中，且与页面显示一致； 4. 生成器、下载器、解析器执行过程

				中有报错，将信息保存在错误日志中 5. 生成器:URI方式变化的进行剩下，没有变化的不重复存储； 6. 下载器:爬取源数据发生变化的进行更新，没变化的不重复存储； 解析器:未改变的数据不再被解析重复存储，数据改变的进行更新

8 性能测试用例设计

8.1 基准测试

基准测试				
名称	基准测试		程序版本	Ver：1.0
测试目的	测试一台机器在不同量的 URI 下，爬取百度数据各项性能指标，为增量测试的性能指标分析做基准参考，同时确保运行环境及生成器、下载器、解析器等配置没有缺陷。 测试一台机器在不同量的URI，不适用 IP 代理基准，爬取工商信息数据的各性能指标，为反爬测试的性能指标分析做基准参考，同时确保运行环境及生成器、下载器、解析器等配置没有缺陷。			
特殊说明				
前提条件	应用程序已经部署，对应生成器、解析器、下载器等配置正确、相应数据已经提供			
步骤	**操作**	**执行总时间**	**成功率**	**平均每条爬取时间**
1	1 台机器，一条 URI，爬取百度数据			
2	1 台机器，1000 条 URI，爬取百度数据			
3	1 台机器，1 万条 URI，爬取百度			

	数据			
4	1 台机器，包含各省的工商网站 URI 1000 条，爬取工商数据			
5	1 台机器，包含各省的工商网站 URI 2 万条，爬取工商数据			
6	1 台机器，包含各省的工商网站 URI 5 万条，爬取工商数据			

8.2 增量测试

增量测试			
名称	增量测试	程序版本	Ver：1.0
测试目的	测试不同机器、不同 URI 量下爬取百度数据的各性能指标值，分析程序承载力		
特殊说明			
前提条件	应用程序已经部署，对应生成器、解析器、下载器等配置正确、相应数据已经提供		

步骤	操作	执行总时间	成功率	平均每条爬取时间
1	5 台机器，5 万条 URI，爬取百度数据			
2	5 台机器，10 万条 URI，爬取百度数据			
3	10 台机器，10 万条 URI，爬取百度数据			
4	10 台机器，20 万条 URI，爬取百度数据			

8.3 反爬测试

反爬测试				
名称	反爬测试		程序版本	Ver：1.0
测试目的	测试一台机器、不同 URI 量下爬取工商数据的各性能指标值，分析程序反爬能力			
特殊说明				
前提条件	应用程序已经部署，对应生成器、解析器、下载器等配置正确、相应数据已经提供			
步骤	**操作**	**执行总时间**	**成功率**	**平均每条爬取时间**
1	1 台机器，1000 条 URI，爬取工商数据			
2	1 台机器，2 万条 URI，爬取工商数据			
3	1 台机器，5 万条 URI，爬取工商数据			