# ScrapingIDE

## 名词解释

爬虫规则配置文件

## 设计目的

网络爬虫是根据人工配置的规则来抓取内容的，无法实现全自动化。当前的做法是通过Chrome DevTools来分析页面结构，然后手工完成规则的配置文件的编写。而配置文件通常需要为每个网站定制，编写工作量大，而且后期需要维护，效率低下。

ScrapingIDE的设计初衷是为了简化爬虫规则的编写工作，提升端到端的爬虫规则的编写效率。ScrapingIDE提供向导式的工具界面，简化规则的编写工作。

## 使用限制

ScrapingIDE无法实现的傻瓜式的全自动配置文件编写工作，使用ScrapingIDE必须了解基本的HTML/CSS概念，ScrapingIDE主要目的是为了提升效率，而无法简化爬虫的规则。

## 架构参考

* 页面提取

由于ScrapingIDE需要代替Chrome DevTools来分析页面，目前比较常见的Web分析工具主要是DevTools以及Firefox的Firebug。综合考虑，使用的跨平台的Firebug Lite，Firebug Lite使用JavaScript编写，并且使用BSD开源协议，符合我们使用的功能要求，可以方便的修改源代码。主体架构是围绕Firebug Lite展开的。

* 界面

页面整体展示使用的boostrap v3，主要是考虑了它的功能较为完善，界面较为美观，相关参考文档和例子比较多

* XML

XML展示部分使用的CodeMirror，易用性和可定制性较强。

* Webview

在Chrome应用中是不支持iframe的，取而代之的是使用webview，在ScrapingIDE中有两个部分使用了Webview，一个是嵌入的浏览器部分，而另外一个部分是导出xml的部分。

嵌入浏览器使用webview比较容易理解，主要是防止浏览的页面和框架部分有互相影响，使用webview可以很好的隔离，但代价是内嵌浏览器和主框架的交互相对比较复杂。

而导出xml使用webview的主要原因是CodeMirror和boostrap的CSS的有冲突，因此使用单独的Webview以作隔离。

## Chrome应用扩展

ScrapingIDE是基于Chrome 应用扩展实现的，Chrome应用扩展和Chrome插件类似，但是功能更加完善，权限控制更加严格。主要通过HTML/CSS/JavaScript实现。

目前Chrome应用扩展的国内的资料比较少，大部分资料是Chrome插件的，可参考的资料较少：

官方的文档（因为特殊原因访问不了，需要翻墙）：

<https://developer.chrome.com/apps/getstarted.html>

360之前翻译了CHROME插件的教程也可以参考一下：

<http://open.chrome.360.cn/extension_dev/overview.html>

百度的插件教程和CHROME的类似，也可以参考：

http://chajian.baidu.com/developer/extensions/getstarted.html

国内写的一个教程《Chrome扩展及应用开发》：

<http://www.ituring.com.cn/minibook/950>

Chrome应用扩展在GITHUB上的例子，非常有参考价值：

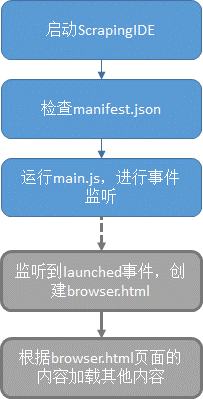
https://github.com/GoogleChrome/chrome-app-samples

## 代码结构

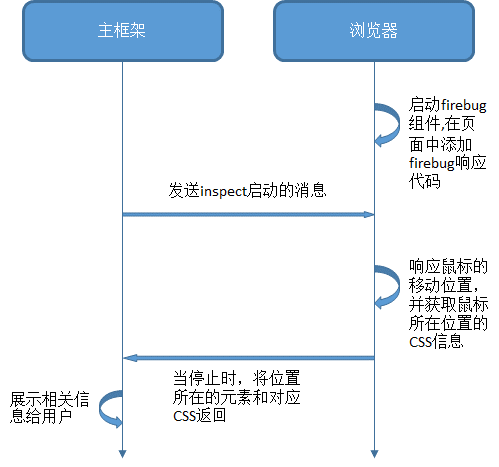
|  |  |
| --- | --- |
| 文件名 | 用途 |
| background.js | 用于在后台响应Firebug的请求 |
| bootstrap.css | Bootstrap v3的CSS文件 |
| bootstrap.js | Bootstrap v3的代码 |
| bootstrap-theme.css | Bootstrap v3的定制主题CSS |
| browser.css | Scraping IDE的全局CSS |
| browser.html | Scraping IDE的整体框架 |
| browser.js | 内嵌浏览器的框架代码 |
| browser\_main.js | 内嵌浏览器的主框架入口 |
| codemirror.css | Code mirror，用于格式化XML文件的排版 |
| codemirror.js | Code mirror，用于格式化XML文件的排版 |
| config.js | 内嵌浏览器的基本配置 |
| export.css | 对应export内嵌webview的导出XML页面的CSS |
| export.html | 对应export内嵌webview的导出XML页面 |
| export.js | 对应export内嵌webview的导出XML页面的脚本 |
| firebug.css | 内嵌浏览器的firebug lite的CSS |
| firebug-lite-beta.js | 内嵌浏览器的firebug lite的主要实现代码 |
| formatting.js | Code mirror的格式化插件 |
| googleChrome.js | Firebug lite的chrome适配代码 |
| jquery.js | Jquery框架代码 |
| load\_context.js | Firebug lite的chrome适配代码 |
| main.js | Scraping IDE应用入口 |
| manifest.json | Scraping IDE的CHROME应用配置文件（重要） |
| popup.js | 控制内嵌浏览器弹出页面的代码（后续废弃） |
| tabs.js | 内嵌浏览器的多标签页面实现代码 |
| title.js | 内嵌浏览器的标题更新实现 |
| vkbeautify.js | XML美化实现 |
| welcome.css | 缺省欢迎页面的CSS |
| welcome.html | 缺省欢迎页面 |
| xml.js | Code mirror的XML解析代码 |
|  |  |

## 关键流程图

* 启动流程



* 抓取页面流程



## 关键函数

Browser.prototype.init

Browser.prototype.initDialog

Browser.prototype.onProcessStyle

Browser.prototype.createNewModule

Browser.prototype.handleRefreshExport

Browser.prototype.handleSaveExport

FirebugChrome.initialize

Firebug.Inspector.startInspecting

Firebug.Inspector.stopInspecting

Firebug.Inspector.sendSelectorMsg

Firebug.Inspector.onInspectingClick