

Fig. 1.

webStraktor

---

## Contents

1.	Preface.....	8
2.	Technical overview.....	9
2.1	webStraktor components .....	9
2.2	webStraktor application package .....	10
2.3	\$WebStraktorRootDir folder structure.....	11
2.4	Subfolder structure .....	12
2.4.1	\$WebStraktorRootDir/Out .....	12
2.4.2	\$WebStraktorRootDir/Java .....	13
2.4.3	\$WebStraktorRootDir/Java/webStraktor .....	13
3.	Operation .....	14
3.1	Installing.....	14
3.2	Compiling .....	14
3.2.1	Pre-requisites.....	14
3.2.2	Ant Compilation script.....	14
3.3	Running.....	15
3.3.1	Modifying Build.xml.....	15
3.3.2	webStraktor run script.....	15
3.4	Alternative operation approach .....	16
3.4.1	Pre-requisite .....	16
3.4.2	HowTo.....	16
4.	Configuration.....	18
4.1	webStraktor configuration file.....	18
4.2	Web proxy server configuration .....	20
4.2.1	webStraktor-Proxies.xml .....	20
4.2.2	active-proxies.xml.....	21
4.3	Useragents.xml .....	22
5.	Quick tour of the webStraktor application .....	23
5.1	Introduction .....	23
5.2	Command line options.....	23
5.3	Major use cases .....	24
5.3.1	Perform regression test.....	24
5.3.2	Perform crawling session.....	24

5.3.3	Perform web proxy maintenance .....	24
6.	The webStraktor scripting language .....	25
6.1	Summary .....	25
6.2	General comments on the scripting language .....	26
6.3	Scripting syntax .....	26
6.4	Input parameters .....	27
6.5	Scripting grammar .....	27
6.5.1	Preprocessor directives .....	27
6.5.2	Blocks .....	28
6.5.3	Controls .....	29
6.5.4	Constructs .....	31
6.5.5	Block Commands .....	35
6.5.6	Code commands .....	35
6.6	FLEX GUI commands .....	36
7.	Tips and tricks .....	38
7.1	Web Proxy server usage .....	38
7.1.1	Web Proxy Server interaction .....	38
7.1.2	Using a web proxy server in webStraktor .....	40
7.1.3	Maintaining the proxy list .....	40
7.1.4	Testing the web proxy server list .....	42
7.2	Specifying webStraktor FORM constructs .....	42
7.2.1	Example 1 .....	43
7.2.2	Example 2 Hidden fields .....	43
7.3	How to use XPATH expressions .....	45
7.4	Web browser signatures .....	45
7.4.1	Bypassing the HTTP Client whilst debugging .....	46
7.5	How to use HTTP queries .....	46
8.	Developer notes .....	48
8.1	Activity diagram .....	48
8.2	The Robot Exclusion Protocol .....	49
8.3	Internal character set .....	50
8.4	Syntax tree and object model .....	50
8.5	Logging .....	51
8.6	Trace .....	52

9.	Step by Step tutorial.....	55
9.1	Introduction .....	55
9.2	Pre-requisites.....	55
9.3	Fetching a web page .....	55
9.4	Adding a FORM .....	56
9.5	Creating an Output file .....	59
9.6	Web crawling .....	61
9.7	Downloading images.....	64
9.7.1	Concluding the script .....	64
9.7.2	Fine-tuning the script .....	65
10.	Java sources.....	67
11.	webStraktor GUI.....	69
11.1	Introduction .....	69
11.2	Compilation.....	69
11.3	Quick introduction .....	70
11.4	Quick usage guide .....	70
11.4.1	Reverse engineering a script .....	70
11.4.2	Create a script.....	71
12.	webStraktor monitor.....	72
12.1	Introduction .....	72
12.2	Compilation.....	72
12.3	Quick introduction .....	72
13.	Known issues .....	75
14.	Further enhancements.....	76
14.1	Code enhancements .....	76
14.2	Outstanding items to be documented in this manual .....	76

---

## Figures

Figure 1 webStraktor components.....	9
Figure 2 webStraktor context.....	10
Figure 3 \$WebStraktorRootDir folder structure .....	11
Figure 4 \$WebStraktorRootDir/Out folder structure.....	12
Figure 5 \$WebStraktorRootDir/Java folder structure .....	13
Figure 6 \$WebStraktorRootDir/Java/webStraktor folder .....	13
Figure 7 Extending Eclipse's build path .....	17
Figure 8 Eclipse run configuration example .....	17
Figure 9 Major use cases .....	24
Figure 10 webStraktor application context.....	77

---

## Notices

Copyright (c) 2014 - webStraktor

webStraktor is free software

Permission is granted to copy, distribute and/or modify this software under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA to obtain the GNU General Public License

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

GNU General Public License: [www.gnu.org/copyleft/gpl.html](http://www.gnu.org/copyleft/gpl.html)

Contact details for copyright holder: [webstraktor@gmail.com](mailto:webstraktor@gmail.com)

---

## Trademarks

Web SQL Client relies on the following freely available technology

- Java SDK 1.5 (or higher)
- Apache HTTPClient (<http://hc.apache.org/httpcomponents-client-ga/>) which is now part of the Apache HTTP Components project.
- MigLayout (<http://www.miglayout.com/>) is a layout manager used to display webStraktor script code, XML and other files. webStraktor GUI uses the xmiglayout15-swing.jar

The webStraktor java source code has been developed to be deployed on various platforms and operating systems. The webStraktor java source code has been tested on the following platforms and Operating Systems.

Platform	Operating system
Intel – AMD	Windows 7 (32 and 64 bit)

The picture of Ramsay's Argon Extractor and a couple of jars on the cover of this manual is available on [http://www.ucl.ac.uk/chemistry/history/chemical\\_history/slides/ramsay\\_apparatus](http://www.ucl.ac.uk/chemistry/history/chemical_history/slides/ramsay_apparatus)

---

## Release note

Release history of this document

Date	Document version	webStraktor version
2014-04-20	V01	Version 1 – Build 2014_04-20

---

## Comments welcome

Mail your comments or defects reports to : [webstraktor@gmail.com](mailto:webstraktor@gmail.com)

---

# 1. Preface

webStraktor is a programmable World Wide Web data extraction client. Its purpose is to scrape HTML based content via the HTTP protocol and extract relevant information. webStraktor features a scripting language to facilitate the collection, the extraction and the storage of information available on the web, including images. The scripting language uses elements of the Regular Expression and xPath syntax. The webStraktor scripting language has a small instruction set and its syntax that is easy to master.

The standard webStraktor output format is XML based, either in ASCII, UTF-8 or ISO-8859-1 (Latin1) code pages.

webStraktor relies on the Apache HttpClient for retrieving content via the HTTP protocol. It adheres to the Robots Exclusion Protocol and it can be configured to operate in an anonymous way by connecting to the predominant types of proxy servers.

webStraktor extends the functionality of web crawlers, web spiders or web bots by integrating scraping and crawling capabilities and it provides exhaustive logging and tracing information.

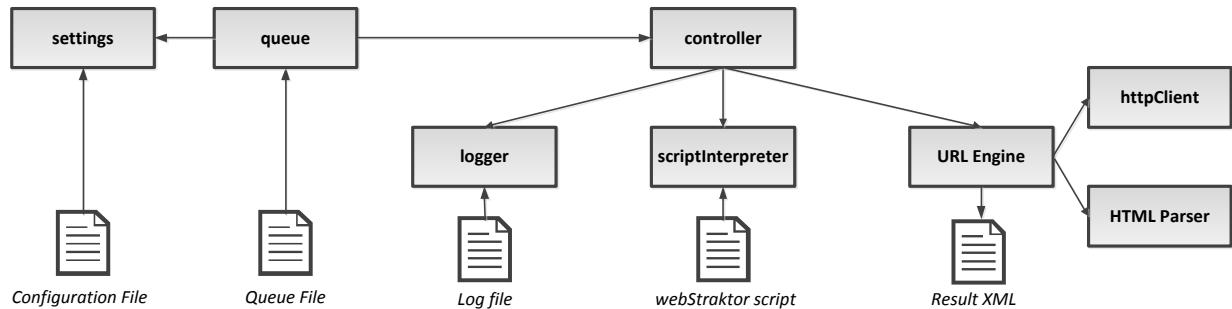


---

## 2. Technical overview

---

### 2.1 webStraktor components



**Figure 1** webStraktor components

The webStraktor architecture comprises the following components

The main component is the Queue component. The Queue component processes the crawler scripts on the queue file. The queue component and other webStraktor components are configured through the Settings component. This component uses configuration settings that are stored in a XML file, e.g. webStraktor.xml. The queue file is a list of webcrawler scripts (webStraktor scripts) that are required to be processed.

The Queue component initiates a Controller component for each webStraktor script that is defined in the queue file. The crawler scripts use a simple procedural scripting vocabulary and syntax.

The Controller uses the Script Interpreter component to transform webStraktor scripts into tokens and a syntax tree. The URL Engine processes the code on the syntax tree.

The URL Engine relies on the Apache HttpClient for fetching HTML pages from the World Wide Web. The URL Engine is able to interpret HTML Form statements and can redirect its HTTP requests to a web proxy server. The URL Engine performs basic text manipulations and writes the result of the instructions defined in the web crawler scripts into the XML result file(s).

The HTML Parser component performs XPATH queries.

Logging information is stored in a log file by the Logger component.

---

## 2.2 webStraktor application package

webStraktor is part of a larger application context which has been depicted on the following diagram.

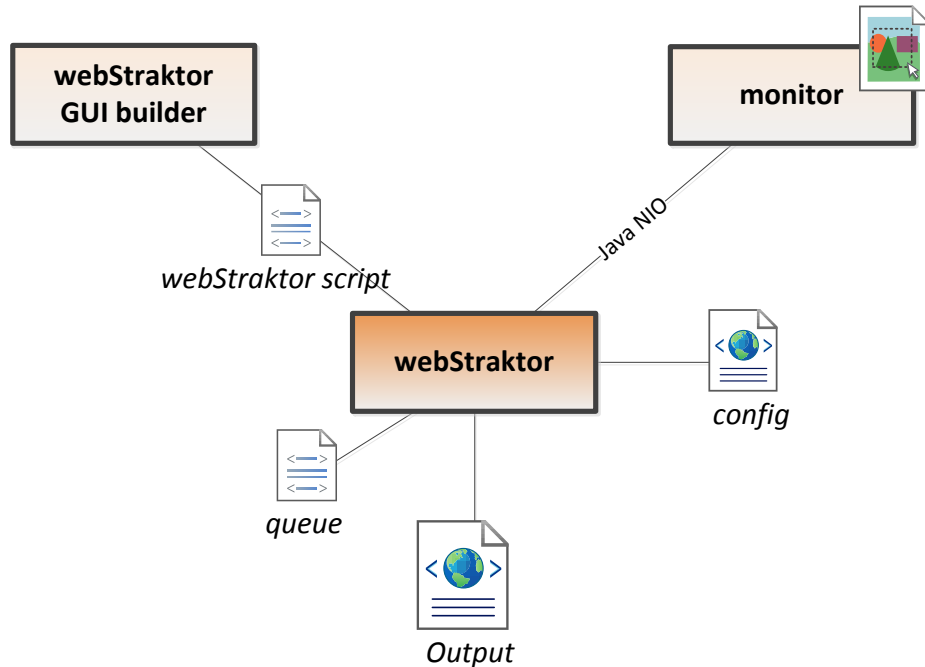


Figure 2 webStraktor context

webStraktor is complemented by two additional applications

- the webStraktor GUI builder is a java Swing based IDE (Integrated Development Environment). It enables to develop webStraktor scripts by dragging and dropping webStraktor scripting elements and connecting those elements.
- The webstraktor monitor is a java Swing application that implements a TCP/IP based Inter Process Communication server using java NIO. webStraktor will transmit its trace information when stepping through a crawler script. The monitor application displays this trace information in a textual or a graphical format, e.g. a sequence diagram or a time diagram.

---

## 2.3 \$WebStraktorRootDir folder structure

The \$WebStraktorRootDir is folder where the webStraktor application is deployed. This folder is comprised of the following subfolders and files.

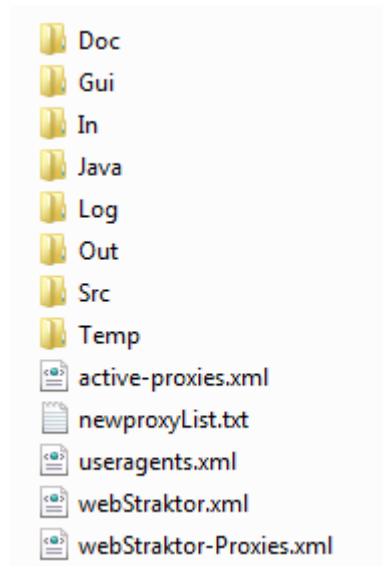


Figure 3 \$WebStraktorRootDir folder structure

Folder or File Name	Narrative
Doc	Comprises the webStraktor Manual and other documents.
Gui	Comprises the working are for webStraktor GUI
In	webStraktor will store all HTML content fetched from the web in this folder. The files names start by the name of the webStraktor script that created the file followed by a sequence number and have a .txt suffix.
Java	The java source code files, ant build files and compiler batch files.
Log	Comprises the various log files created by webStraktor
Out	This folder has the XML output files and the XML consolidation files.
Src	webStraktor scripts are stored in this folder.
Temp	Working storage area for webStraktor
active-proxies.xml	This is a configuration file that keeps track of the availability of web proxy servers
newProxyList.txt	This is a file that is used to automatically load additional web proxy servers into the webStraktor-proxies configuration file
webStraktor.xml	This is the main webStraktor configuration file
webStraktor-	This configuration file defines the web proxy servers that webStraktor can

proxies.xml	use. It also defines the type of proxy, its cookie policy, form input fields and action.
Useragents.xml	The list of User Agent signatures that can be used

## 2.4 Subfolder structure

### 2.4.1 *\$WebStraktorRootDir/Out*

This is the structure of the Out folder.



Figure 4 *\$WebStraktorRootDir/Out* folder structure

Folder or File Name	Narrative
Ascii	<p>webStraktor stores the output XML files which are created here and uses the ASCII code page.</p> <p>The name of the output XML file is defined via a pre-compiler directive in the webStraktor crawler script (OUTPUTFILENAME).</p> <p>There is one output XML file for each script and for each differing input parameter.</p> <p>For example : random-books-franzen.xml; is the file created by the random-books script when the input parameter is set to franzen.</p> <p>There is also a consolidated XML file, which accumulates the latest version of each output file per webStraktor script. Consolidated files have the “cons” suffix.</p> <p>For example : random-books-cons.xml.</p>
Blob	Folder where image files are downloaded
Latin1	XML output files in ISO-8859-1 or Latin1 code page
Utf8	XML output files in UTF-8 code page
Other	If the KEEPOUTPUTFILE compiler directive is enabled, temporary output files are not removed from this folder.

### 2.4.2 *\$WebStraktorRootDir/Java*

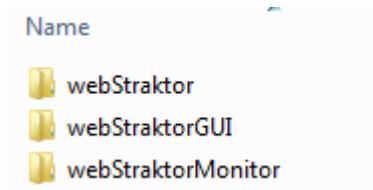


Figure 5 *\$WebStraktorRootDir/Java* folder structure

Folder Name	Narrative
webStraktor	Java source code, Ant build file, compile and run scripts.
webStraktorGUI	Java source code, etc. for the webStraktor GUI development tool.
webStraktorMonitor	Java source code, etc. for the webStaktor monitor GUI tool.

### 2.4.3 *\$WebStraktorRootDir/Java/webStraktor*

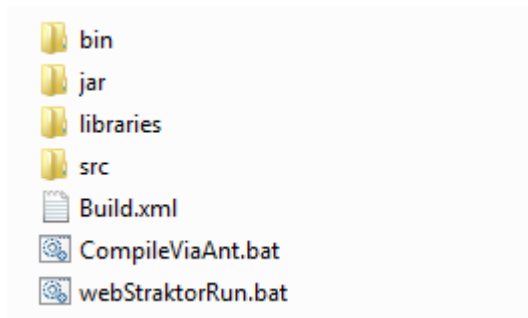


Figure 6 *\$WebStraktorRootDir/Java/webStraktor* folder

Folder or File Name	Narrative
Bin	Ant puts the .class files in this folder.
Jar	webStraktor.jar
Libraries	Comprises the Apache HTTPClient jars
Src	webStraktor source code
Build.xml	Ant build file
CompileViaAnt.bat	Batch file to compile webStraktor via Ant
WebStraktorRun	Batch file to run webStraktor via Ant

---

## 3. Operation

---

### 3.1 Installing

The installation of webStraktor is rather straightforward.

- Create a webStraktor Root Directory (\$WebStraktorRootDir) on the host system, for example D:\webStraktor.
- Expand the contents of the webStraktor software distribution (i.e. the ZIP file) into this directory.

This should create a folder structure identical to the one described in the previous section of this manual.

---

### 3.2 Compiling

#### 3.2.1 Pre-requisites

The following additional software packages are required to be deployed on the host system.

- A Java Software Development kit : Java SDK 1.5 or higher
- The Apache Ant build tool : Ant 1.8.2 or higher

The \$WebStraktorRootDir/Java/libraries folder comprises the Apache HTTPClient jar files, so there is no need not download or install these jars independently.

The \$WebStraktorRootDir/Java/webStraktor directory contains webStraktor's Ant build file (Build.xml)

#### 3.2.2 Ant Compilation script

Below are the contents of the compilation script (WebStraktorCompileViaAnt.bat). A sample compilation script is available in \$webRootDir/Java/webStraktor. You might however need to adapt it as follows.

- the PATH environment variable, so that it includes a correct reference to your Java compiler. In this case the Java compiler is located in D:\javaprogram\jdk\_1.5.0\_02\bin.
- The ANT\_HOME environment variable so that it point to the folder where Ant has been installed. In this case it is c:\temp\Ant\apache-ant-1.8.2, you will more than likely need to adapt this.
- the WEBSTRAKTOR\_HOME environment variable should be set to the \$WebStraktorRootDir, in this case D:\webStraktor.

```

set JAVA_HOME=D:\javaprogram\jdk_1.6.0_24
set ANT_HOME=C:\temp\Ant\apache-ant-1.8.2
set WEBSTRAKTOR_HOME=D:\webStraktor
PATH=%JAVA_HOME%\bin;%ANT_HOME%\bin;%PATH%
REM
D:
CD %WEBSTRAKTOR_HOME%\Java\webStraktor
ant build jar webStraktor -f ./build.xml
pause

```

Just run the above script. It will create the webStraktor jar file in the \$WebStraktorRootDir/Java/webStraktor/jar directory and upon successful completion try to run it.

## 3.3 Running

### 3.3.1 Modifying Build.xml

You will need to modify the Ant Build.xml file in order to define the webStraktor commandline parameters. Look for the <target> node in the build.xml build file and modify where required.

```

<target name="webStraktor">
    <java classname="webStraktorMain" failonerror="true" fork="yes">
        <arg line="-D d:\webStraktor -C webStraktor.xml -Q queue.txt" />
        <classpath refid="webStraktor.classpath" />
    </java>
</target>

```

### 3.3.2 webStraktor run script

The script below is a typical Ant script for running webStraktor once it has been compiled

You might need to adapt it.

- The PATH environment variable, so that it includes a correct reference to your Java compiler.
- The ANT\_HOME environment variable
- the WEBSTRAKTOR\_HOME environment variable should be set to the \$WebStraktorRootDir, in this case D:\webStraktor.

```
set JAVA_HOME=D:\javaprogram\jdk_1.6.0_24
set ANT_HOME=c:\temp\Ant\apache-ant-1.8.2
set WEBSTRAKTOR_HOME=D:\webStraktor
PATH=%JAVA_HOME%\bin;%ANT_HOME%\bin;%PATH%
D:
CD %WEBSTRAKTOR_HOME%\Java\webStraktor
ant webStraktor
exit /B
```

Caution : If webStraktor has been deployed on a drive differing from D: make sure to adapt the 5<sup>th</sup> line in the above script.

---

## 3.4 Alternative operation approach

### 3.4.1 Pre-requisite

The following additional software package is required for this alternative approach

- Eclipse IDE for Java Developers, Helios distribution or above (<https://www.eclipse.org/>)

### 3.4.2 HowTo

A user-friendly and low complex approach for compiling and running webStraktor is to install the webStraktor package as described in the previous sections and subsequently create an Eclipse project for the Java webStraktor sources.

The Eclipse IDE for Java Developers distribution will suffice. Use Helios or a more recent release.

You will need to manually import all java source code files from \$WebStraktorRootDir/Java/webStraktor/Src in your Eclipse project's workspace and also extend the project's Build Path with the Apache HTTPclient jars (Right click project name -> Build Path).

These jars can be found in the \$WebStraktorRootDir/Java/Libraries folder.



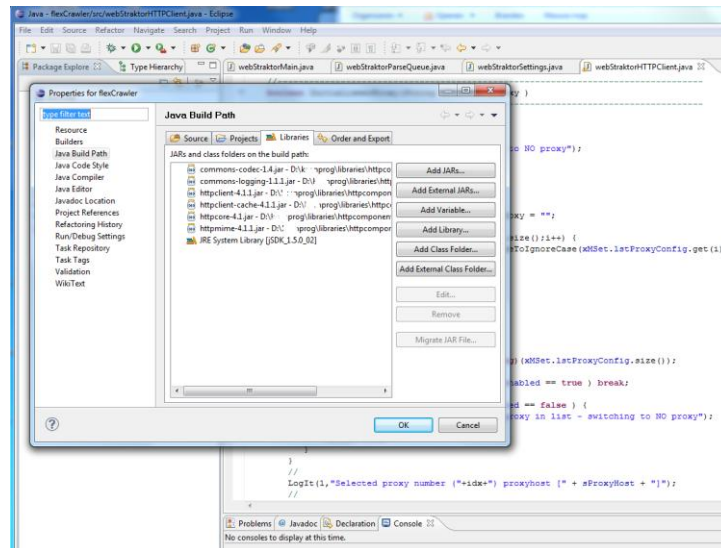


Figure 7 Extending Eclipse's build path

You will also need to configure the project's run configuration (Run -> Run Configurations) in order to be able to use the various webStraktor command line parameters.

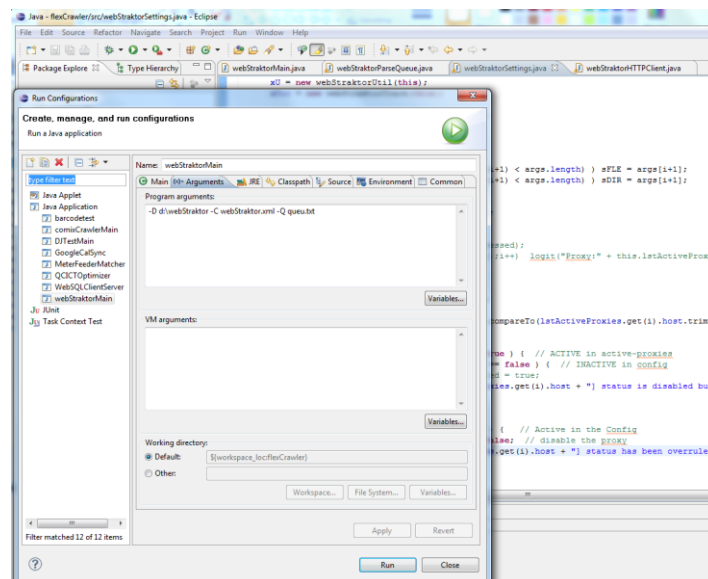


Figure 8 Eclipse run configuration example

---

## 4. Configuration

The webStraktor application can be configured via a configuration file formatted in XML. In addition to the main configuration file webStraktor also requires two web proxy server configuration files.

Just like many other applications, webStraktor uses command line parameters, for example to input the name of the configuration file and the folder of this configuration file to the webStraktor application.

An example of an webStraktor XML file can be found at the end of this section. The name of this configuration file can be chosen freely.

Notation. The original name of the webStraktor project was flexCrawler, so some configuration settings might still refer to it.

---

### 4.1 webStraktor configuration file

The following settings that primarily affect webStraktor's behavior can be configured in the main configuration file.

Setting	Comment	Options / example
LOGFILENAME	The name of the major logfile.	webStraktor.log
DATEFORMAT	The format to be used in the logfiles.	DD-MMM HH:MI:SS-MIL
TARGETCODEPAGE	UTF-8 or Latin1. WebStraktor stores its results in UTF-8 format. In the event Latin1 is specified the results will also be stored in Latin1 format in the \$WebRootDir/Out/Latin1 folder	{Latin1,UTF8}
OverRuleRobotSpecification	Enables to overrule the /robots.txt protocol settings	Default is no.
CrawlerDelay	The period expressed in seconds between two successive web page retrievals	Default is 5 seconds
IPCPort	This is the Interprocess Communication TCP/IP Port which is used to transfer trace information from webStraktor to the	Default is 8001

	monitor application.	
ProxyTestURL	The URL of a page to be used when testing the proxy.	e.g. <a href="http://www.lecourrierdusud.ca">www.lecourrierdusud.ca</a> Be courteous and make sure that you specify a web site that is able to sustain a heavy workload.
ProxyPassPhrase	This is a text snippet present on the ProxyTestURL. It is used to assess whether a proxy webserver is working.	e.g. <i>Votre journal</i>
AssessRobotsTxt	Toggle that enables to instruct webStraktor to adhere to the robots.txt protocol	Default is yes
FullReportHTMLMode	When set HTML pages are dumped in the \$WebStraktorRootDir/In folder.  The HTML is dumped in the DOM (Document Object Model) format, which can be used to define XPATH queries.	Default is no

#### Notation.

- Although XML is case sensitive, webStraktor will accept case insensitive tags.
- Boolean values can be expressed in a case insensitive way as follows : true, yes, 1, false, no, 0.

This is an example webStraktor.xml configuration file. This is also the default configuration: the implementation of the robot exclusion protocol is active and the CrawlerDelay is set to 4 seconds.

```

<webStraktorCONFIG>
  <LOGFILENAME>webStraktorLog.txt</LOGFILENAME>
  <DATEFORMAT>DD-MMM-YY HH:MI:SS:MIL</DATEFORMAT>
  <LOGDATEFORMAT>DD-MMM HH:MI:SS-MIL</LOGDATEFORMAT>
  <TARGETCODEPAGE>UTF-8</TARGETCODEPAGE>
  <OverRuleRobotSpecification>no</OverRuleRobotSpecification>
  <CrawlerDelay>4</CrawlerDelay>
  <ProxyTestURL>www.lecourrierdusud.ca</ProxyTestURL>
  <ProxyPassPhrase>Votre journal</ProxyPassPhrase>
  <AssessRobotsTxt>yes</AssessRobotsTxt>
  <FullReportHTMLMode>yes</FullReportHTMLMode>
  <IPCPort>8001</IPCPort>
</webStraktorCONFIG>

```

---

## 4.2 Web proxy server configuration

webStraktor relies on two files for configuring its web proxy server capabilities.

- webStraktor-Proxies.xml; this file defines the web Proxy servers that can be used.
- active-proxies.xml; webStraktor can assess whether a web proxy server is still active. The status of a web proxy server is stored in this file.

The structure of these configuration files is defined for reference purposes. Although the content of both files can be entered manually, it is advised to have webStraktor generate these files (see the command line options section and the section 5.2 for detailed instructions on webStraktor's automated configuration capabilities).

### 4.2.1 webStraktor-Proxies.xml

The following table defines the webStraktor configuration settings for web proxy servers.

The proxy settings are stored between <proxy> and </proxy> tags in the webStraktor-proxies.xml file. The tag of the root node of this file is <ProxyList>.

Tag	Comment	Example
Host	Hostname of the proxy	hidej.info
Action	Defines the POST action of a page	Typical for Glype proxies : includes/process.php?action=update
Input	Defines the input field that will be used in the POST action	Glype proxies typically have a single input field named "u"
Active	Defines whether or not the proxy can be used by webStraktor	Default is yes
Priority	Currently not used	
CookiePolicy	Defines the HTTP cookiepolicy (currently not used)	Default is : Browser Other policies supported are : {RFC_2965,BEST_MATCH,NETSCAPE,RFC_2109}
CookieEnabled	Defines whether or not a cookie is enabled.  Currently not used	Default no

The following text snippet has been cut from a webStraktor generated web proxy configuration file.

```

<?xml version="1.0" encoding="UTF-8" ?>
<!-- webStraktor generated Proxy Config File -->
<!-- 12-Apr-14 08:06:38:116 -->
<ProxyList>
  <Timestamp>1397286398116</Timestamp>
  <Proxy>
    <Host>filterhome.info</Host>
    <Action>includes/process.php?action=update</Action>
    <Input>u</Input>
    <Active>no</Active>
    <Priority>0</Priority>
    <Cookiepolicy>browser</Cookiepolicy>
    <CookieEnabled>no</CookieEnabled>
  </Proxy>
  .. etc ..

```

### 4.2.2 active-proxies.xml

The webStraktor application keeps track of the status of the web proxy servers it uses in the \$WebStraktorRootDir/active-proxies.xml file.

Proxy	Node tag
Host	The DNS name of the web proxy server, either with or without the www prefix.
Status	{active,inactive,disabled}
Error	<p>If webStraktor encounters an error when performing a heathcheck on a web proxy server, the HTTP error is logged in this field.</p> <p>For example. CLN - ACTION RAW[includes/process.php?action=update] CLN - ACTION PROCESSED[http://www.austria2.com/includes/process.php?action=update] PRS - Could not find pass phrase [Votre journal] in file [c:\temp\procCrawler\In\flexProxyTest.txt-1.txt] CTR - Error when processing [http://www.lecourrierdusud.ca] with script [c:\temp\procCrawler\Src\flex-Proxy-Test-9.txt]&lt;/error&gt;</p>

This is an example active-proxy file. The root node is defined by the <ActiveproxyReport> tag. There is also a timestamp logging just under the root node level which is used by the webStraktor application to log the last time the availability of the web proxies has been verified and to warn the user via the log files to consider refreshing the active-proxies.xml file.

```

<?xml version="1.0" encoding="UTF-8" ?>
<!-- webStraktor active proxy report -->
<!-- 12-Apr-14 08:06:38:116 -->
<ActiveproxyReport>
  <Timestamp>1397286398116</Timestamp>
  <proxy>
    <host>filterhome.info</host>
    <status>disabled</status>
  </proxy>
  <proxy>
    <host>www.docoja.com</host>
    <status>disabled</status>
  </proxy>

```

### 4.3 Useragents.xml

This configuration is used to define a list of user agents signatures that can be used to inject in the HTTP header. See the section on Browser Signatures.

User agent signatures are defined by specifying a browser code and the user agent signature (see the xml below). The webStraktor scripts can set the browser code.

```

<?xml version="1.0" encoding="UTF-8" ?>
<!-- webStraktor User Agent Signatures -->
<useragentlist>
  <useragent>
    <browser>default</browser>
    <signature>Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.1.6) Gecko/20091201
Firefox/3.5.6 (.NET CLR 3.5.30729</signature>
  </useragent>
  <useragent>
    <browser>firefoxw7</browser>
    <signature>Mozilla/5.0 (Windows NT 6.1; WOW64; rv:28.0) Gecko/20100101
Firefox/28.0</signature>
  </useragent>
  <useragent>
    <browser>iew7</browser>
    <signature>Mozilla/4.0 (Mozilla/4.0; MSIE 7.0; Windows NT 5.1; FDM; SV1; .NET CLR
3.0.04506.30)</signature>
  </useragent>
</useragentlist>

```

---

## 5. Quick tour of the webStraktor application

---

### 5.1 Introduction

This section walks you through the main use cases of the webStraktor application, but commences by commenting on the command line options.

---

### 5.2 Command line options

These are the commandline options which are required

`-D <FolderName> -C <ConfigurationFile.xml>`

In addition to the required command line options, one of the following options needs to be added

- `-Q <QueueFileName.txt>`; will run multiple webStraktor scripts that are defined in the Queue file
- `-P <webStraktorScript.txt>`; will run a single webStraktor script
- `-ADDPROXY <ListOfProxies.txt>`; will add the web proxies that are defined in the ListOfProxies.txt file and update the webStraktor-proxies and active-proxies configuration files.
- `-PROXYTEST`; will validate whether the web proxies defined in the webStraktor-proxies file are still active.
- `-REGRESSSIONTEST`; will perform a basic regression test script. This can be used to perform a basic post installation test.

For example

- `java -jar webStraktor.jar -D d:\webStraktor -C webStraktor.xml - Q myQueue.txt` will execute all webStraktor scripts defined in `d:\webstraktor\myQueue.txt`.
- `java -jar webStraktor.jar -D d:\webStraktor -C webStraktor.xml - ADDPROXY MyProxyList.txt` will try to add the proxies defined in `$WebstraktorRootDir/MyProxyList.txt` to the webStraktor-Proxies.xml configuration file.
- `java -jar webStraktor.jar -D d:\webStraktor -C webStraktor.xml - PROXYTEST` can be used to verify the availability of web proxy servers.

---

## 5.3 Major use cases

The following diagram depicts webStraktor's the major use cases.

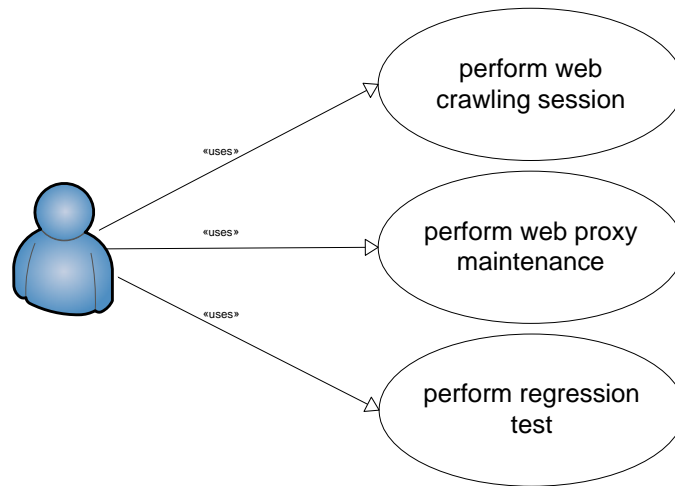


Figure 9 Major use cases

### 5.3.1 Perform regression test

webStraktor has a built-in regression test which is used for post-installation testing purposes or merely to perform a non-regression test on newly released webStraktor code.

Just run `$WebStraktorRootDir/Java/webStraktor/webStraktorRegressionTest.bat`. This batch file launch Ant and run the regressionTest section in its Build.xml file.

### 5.3.2 Perform crawling session

A couple of sample webStraktor scripts are part of the standard code distribution. Once webStraktor has been installed and tested (see the above use case) you can testdrive the Random House webStraktor script.

Just double click on `$WebStraktorRootDir/Java/webStraktor/webStraktorRun.bat`. webStraktor will run for a couple of seconds and create a file in `$WebStraktorRootDir/out/Utf8/Random-House-fransen.xml`.

Just run `$WebStraktorRootDir/Java/webStraktor/webStraktorRun.bat`. This batch file launch Ant and run the webStraktor section in its Build.xml file. This option will launch the script defined in the `$WebStraktorRootDir/queue.txt`. Out of the box this will run the Random House crawler script.

### 5.3.3 Perform web proxy maintenance

Just run `$WebStraktorRootDir/Java/webStraktor/webStraktorProxyTest.bat`. This batch file launch Ant and run the proxyTest section in its Build.xml file. This will assess the availability of the web proxies defined in `$WebStraktorRootDir/webStraktor-proxies.xml`.



---

## 6. The webStraktor scripting language

---

### 6.1 Summary

webStraktor uses scripts in order to crawl the world wide web and extract information from HTML based web content. The webStraktor scripting language is procedural. Its syntax is similar to third generation programming languages such as C and Java. The scripting language supports common procedural concepts such as functions, controls and operators.

Below is a basic webStraktor script. It connects to [www.isbnsearch.org](http://www.isbnsearch.org). It executes a query on an ISBN number (1), stores the title (3) of the book and the name of the author (4) in a file named \$webStraktorRootDir/Out/AuthorViaISBN.xml (2).

```
-- ISBN script
OUTPUTFILENAME AuthorViaISBN
CONSOLIDATE true
KEEPOUTPUTFILE true
LOGLEVEL 9
SIMULATION false
PROXY yes
--
MAIN
  REM $1 ISBN number
  URL http://www.isbnsearch.org/isbn/$1

  XPATH
  /html/body/div[@id="page"]/div[@id="content"]/div[@id="book"]/div[@class="bookinfo"]

  VAR title
    XPATH /div/h2
    CODE {getHTMLContent();trim();}
  END VAR

  VAR Author
    STARTPATTERN Author:
    ENDPATTERN </p>
    CODE {getHTMLContent();trim();}
  END VAR

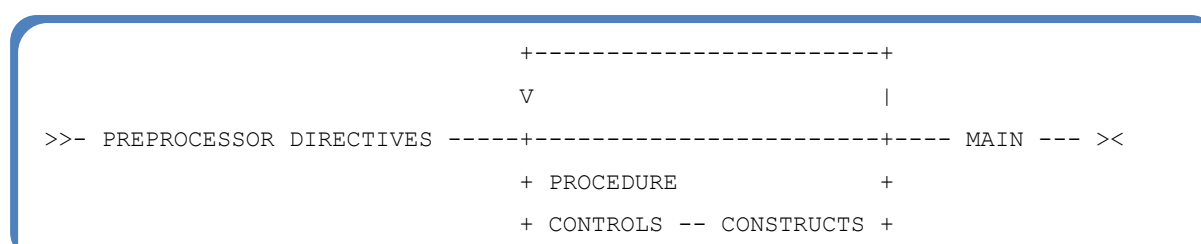
END MAIN
```

## 6.2 General comments on the scripting language

- The webStraktor scripting language is not case-sensitive for the names of commands and operators. The names of the identifiers of PROCEDURE, VAR, LINK and BLOB are case-sensitive.
- There are no end of line delimiters required.
- Comments are to be preceded by either one of the following operators : REM, -- or //
- Most of the commands have a code-value pair structure, e.g. (END,MAIN), (VAR,variablename), etc.
- A script can contain as many functions as needed. All functions are within the scope of all other functions. Nesting is not supported.
- Each webStraktor script must have a main code block. It starts with MAIN and ends with END MAIN.
- The instructions in the script are executed in the order of appearance within a block of code. A block of code is confined to the boundaries of a PROCEDURE, the MAIN block or the overarching and implicit ROOT block.
- Notation. Although not required, it is advised to enclose the value of webStraktor command-value pair in double quotes.
- Notation. You need to escape double quotes by \ when double quotes are part of the value. For example STARTPATTERN "<div class=\"list container\">".

### 6.3 Scripting syntax

A webStraktor script starts with Preprocessor directives and is comprised of a sequence of code blocks. These code blocks can be procedures, a combination of controls and constructs. Each script must have a main block.



Similarly a procedure is comprised of controls and constructs

The MAIN block generally starts by defining a FORM construct or by a URL statement in order to retrieve web content. It is good coding practice that FORM or URL statements are immediately followed by a PASSPHRASE command to assess whether the page was correctly retrieved from the world wide web.

```
>>- MAIN -----+-----+-----+-----+----- ..etc.. --END MAIN-- ><
+ URL + + PASSPHRASE +
+ FORM +
```

## 6.4 Input parameters

Input parameters can be used to deliver information from the webStraktor Queue component to the script. An input parameter can be referenced via \$n.

In the previous example script the instruction “URL [http://www.isbnsearch.org/isbn/\\$1](http://www.isbnsearch.org/isbn/$1)” enables to define the value of an ISBN number in the queue file or on the command line. Its value will be used when the script is run.

The order of the values on the command line defines the order of substitution.

## 6.5 Scripting grammar

This section explains the various commands that are supported by the webStraktor scripting language.

### 6.5.1 Preprocessor directives

Not all the lines in a webStraktor script are executable statements. Some of the statements are "preprocessor directives". These are instructions that drive the execution time behavior of the webStraktor Controller and its subcomponents.

Notation. The following values can be used when defining a toggle : {yes,no,on,off,true,false,1,0}

Preprocessor directive	Description
BROWSER	This is the code which will be used to lookup the User Agent signature. If the BROWSER code is not specified the default User Agent will be used (see section on Browser Signatures).
CONSOLIDATE	This is a toggle. When set the output of the same webStraktor scripts are consolidated into a single output file located in the \$OutputFile folder. The consolidate option enables to run the same script with different input parameters and to collate the various results into a single file.
KEEPOUTPUTFILE	This is a toggle. When set the HTML content that has been fetched by the HTTPClient is not deleted from the \$InputFolder. This option enables to examine the HTML content or to use the content of this file when developing or debugging a webStraktor script.

LOGLEVEL	Ranges from 0 to 9. The most detailed log level is 9.
OUTPUTFILENAME	This is the name of the file which will hold the results of the webcrawler session.
PROXY	<p>This can be both a toggle or an instruction.</p> <p>When used as a toggle, it instructs the HTTPClient to either use or not use a HTTP Proxy.</p> <p>When used as a directive one can define the name of a HTTP proxy.</p>
ROBOT	<p>Possible values = {true,false,default}</p> <p>This setting enables to overrule the AssessRobotsTxt setting in the major configuration file.</p> <p>Use case : some web sites do not provide a /robots.txt file, so when the global AssessRobotsTxt is enabled an error occurs when trying to open the first page of the site.</p>
SIMULATION	This is a toggle. When set the HTTPClient is disabled and the results of previous webcrawling sessions that have been stored in the \$InputFolder are re-used instead of accessing the content via the HTTPClient. Before this option can be used one must have run the webStraktor script once with SIMULATION mode off and set the KEEPOUTPUTFILE switch to yes.

### 6.5.2 Blocks

Function	Description
MAIN	<p>MAIN</p> <p>payload</p> <p>END MAIN</p> <p>This command enables to define the main body of a webStraktor script.</p> <p>In general there ought to be a URL or a FORM statement at the beginning of a MAIN in order to ensure that at least content is fetched from the world wide web.</p>
PROCEDURE	<p>PROCEDURE {procedure name}</p> <p>payload</p> <p>END PROCEDURE</p>



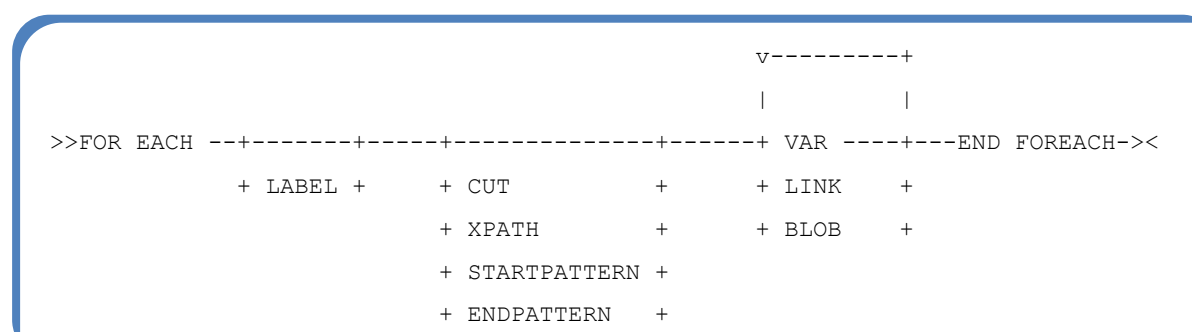
	<p>CUT</p> <p>STARTPATTERN "&lt;div class=\"gallery_content\"&gt;"</p> <p>ENDPATTERN "&lt;div id=&lt;\"catalog_footer\"&gt;"</p> <p>END CUT</p>
CALL	<p>CALL {Procedure name}</p> <p>Enables to invoke a procedure. For example : CALL MyProcedure</p>
FOREACH	<p>FOREACH</p> <p>    payload</p> <p>END FOREACH</p> <p>Often web content is structured in tabular or other repetitive formats. The FOREACH enables to iterate through a set of commands that relate to each of the rows of such a table.</p>
PASSPHRASE	<p>PASSPHRASE {string to be evaluated}</p> <p>The PASSPHRASE command should be used immediately after the URL command. It is a command that evaluates whether a given string pattern is part of the HTML content that was fetched by the URL command. The prime objective of the passphrase is to ensure that the web content that was fetched matches the content the logic or flow of the script anticipates.</p> <p>If the passphrase is not resolved correctly the script aborts.</p> <p>For example.</p> <p>MAIN</p> <p>    URL        http://www.worldjam.eu/search?q=john</p> <p>    PASSPHRASE  "&lt;!DOCTYPE html PUBLIC"</p> <p>END MAIN</p> <p>This code snippet will fetch a page from worldjam.eu. The PASSPHRASE command will evaluate whether or not the string &lt;!DOCTYPE html PUBLIC is part of the HTML content that was fetched.</p>
URL	<p>URL {url}</p> <p>This command instructs the HTTPclient to fetch and to store the contents of a given URL.</p> <p>The HTML content is stored in a file in the \$InputFolder with the name</p>

	<p>that is the same of the name of the webStraktor followed by a sequence number.</p> <p>The URL can contain input parameters, defined by \$n. For example in "URL <a href="http://www.isbnsearch.org/isbn/\$1">http://www.isbnsearch.org/isbn/\$1</a>" \$1 will be substituted by the first command line parameter on the queue or immediately after – P &lt;scriptname&gt;.</p>
--	---

### 6.5.3.1 FOR-EACH syntax diagram

A FOR-EACH can start by defining a name via the LABEL or STORE command. In most cases a subset of the HTML markup code is then selected via CUT, XPATH or START/ENDPATTERN and finally the values of the variables, the images or links are extracted.

There can be one or many constructs in the FOR-EACH.



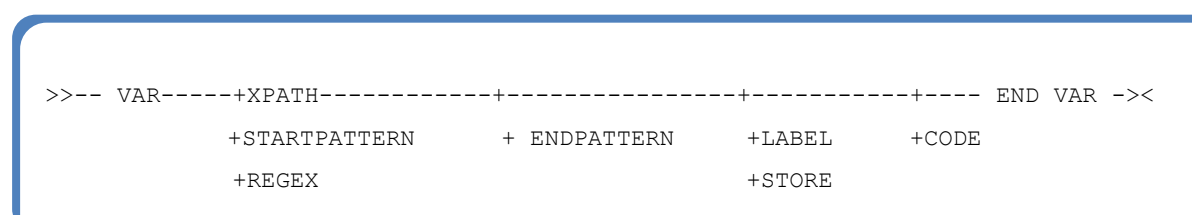
## 6.5.4 Constructs

These are the constructs currently supported by the webStraktor script language.

- BLOB or IMAGE
- FORM
- LINK
- VAR

### 6.5.4.1 VAR

The main construct in the script is the VAR construct. It enables to define a variable and its content.



Example VAR

---

```

VAR author
  LABEL AuthorCode
  STARTPATTERN <author>
  ENDPATTERN </author>
  CODE {TRIM();LOWERCASE()}
END VAR

```

---

#### 6.5.4.2 LINK

The LINK construct is similar to the VAR construct. This construct is used to extract a URL and load the contents of referenced page via the CALL command. The CALL command calls another function that processes the content of the URL. The CALL command is required.

```

>>-- LINK--+XPATH-----+-----+-----+-----CALL---END LINK -><
      +STARTPATTERN  +ENDPATTERN  +LABEL  +CODE
      +REGEX                                     +STORE

```

#### Example LINK

---

```

LINK linkOne
  STARTPATTERN "<a"
  ENDPATTERN "</a>"
  CODE { get("href") }
  STORE linkOne
  CALL procedureTwo
END LINK

```

---

#### 6.5.4.3 BLOB or IMAGE

The BLOB (Binary Large Object) construct is similar to the VAR construct. It is used to extract the URL of an image and download the image in the \$WebStraktorRootDir/Out/Blob folder.

```

>>--BLOB--+XPATH-----+-----+-----+-----END BLOB -><
      IMAGE  +STARTPATTERN  +ENDPATTERN          +LABEL +CODE
              +REGEX                                     +STORE

```

#### Example BLOB statement



---

```

BLOB BookImage
    LABEL          AlbumCoverImage
    STARTPATTERN   "<div style=\"float: left;\">"
    ENDPATTERN     "class=\"imgThumbAlbumCover_image\""
    CODE           {GET("href");TRIM()}
END BLOB

```

---

#### 6.5.4.4 FORM

The FORM construct is used to submit data from the webStraktor client into an HTML form either via POST or GET method. The webStraktor URL parser component determines which method to use by evaluating the markup code of the HTML form.

```
>>----- FORM ----- FORMNAME ----- URL --- INPUT ----- END FORM -----<<
```

##### 6.5.4.4.1 First FORM example

The following piece of HTML code shows

- a typical HTML form featuring a data entry field for an ISBN number
- a submit button named “Search” using the GET method.

---

```

<form name="input" action="demo_form_action.do" method="get">
  ISBN Number <input type="text" name="isbn" type="submit" value="Search">
</form>

```

---

This is the webStraktor FORM statement corresponding to the above HTML form.

---

```

FORM
    FORMNAME      input
    URL           http://www.dosomething.com/search
    INPUT         isbn=$1
END FORM

```

---

##### 6.5.4.4.2 Second FORM example

There can be multiple INPUT fields on an HTML and consequently webStraktor FORM.

---

```

FORM
    FORMNAME      3
    URL            http://www.dosomething.com/search
    INPUT          SearchISBN=$1
    INPUT          SearchLanguage=Hungarian
END FORM

```

---

See Tips & Tricks section for the detailed discussion on how to write webStraktor FORM statements

#### 6.5.4.5 Construct commands

Command	Description
CALL	CALL <procedurename> CALL is a required command in the LINK construct.
CODE	CODE { command; * }  The code command enables to perform a number of string operations on the content of a variable. The code commands need to be separated by a semi-colon and must be enclosed in curly brackets.  See the next section for a discussion on the various commands.
LABEL	LABEL <name> is used to provide a name to a variable. The name is the tag that is going to be used in the output file.
STORE	Identical to LABEL
STARTPATTERN	This is the startpattern of the snippet of text that will be cut from the HTML code.
ENDPATTERN	This is the endpattern of the bits of text that will be cut from the HTML code of the web page.
XPATH	For example XPATH "/html/body/div[@id="page"]/div[@class="bookinfo"]".  This is the access path to the code block in the DOM that is defined by the above XPath instruction. In this case a "div" block labeled "bookinfo" hanging off another "div" block labeled "page", which is located under the main "body" of an HTML page.
REGEX	Regular expression used to cut a snippet of text from the HTML file.
FORMNAME	This is the name of form as it is defined on the HTML page
URL	This is the URL of the HTML based webpage on which the FORM is defined. webStraktor will first fetch this page, fill in the required input

	fields and submit the form either via POST or GET.
INPUT	This is the name of INPUT field on the HTML form and the value that is required to be put into that field.

### 6.5.5 Block Commands

The following commands can be used within a block.

Command	Description
ENDPATTERN	This is used to define the end of a snippet of HTML text in a PROCEDURE, FOR-EACH or MAIN block
IGNORE	This can be used if one does not want to store the content of a LINK, VARIABLE or BLOB.
LABEL	Puts a label or a name onto a FOR-EACH This name or label will also be used in the XML output file's node tag.
MAXITERATIONS	MAXITERATIONS n If the Nth occurrence of a block is reached, the FOR EACH loop stops
OCCURs	OCCURS N,M,etc Process the Nth, Mth, etc occurrence of a block in a FOR EACH
SKIP	SKIP N,M,etc Skips the Nth, Mth, etc occurrence of a block in a FOR EACH
STARTPATTERN	Is used to define the start of a snippet of HTML text in a PROCEDURE, FOR-EACH or MAIN block.
TRACEID	Irrelevant when webStraktor scripts are created manually. TRACEID command is used to provide a unique identifier to a webStraktor command.
XPATH	Is used to define a snippet of HTML text in a PROCEDURE, FOR-EACH or MAIN block.

### 6.5.6 Code commands

Commands	Description
----------	-------------

GET	GET(TAG) or GET("TAG") Extracts the value that is related to the HTML tag.
GETFIELD	GET(n,DELIMITER) e.g. GetField(2,' ') Extracts the Nth field within a string delimited by DELIMITER
GETHTMLCONTENT	Removes all HTML tags
GETINPUTPARAMETER	GetInputParameter(n) Displays the Nth input parameter
KEEPNUMBER(S) KEEPNUMBERS	Remove all non-numeric characters from a string
KEEPDECIMAL(S) KEEPFLOAT(S)	This will remove all non-numeric characters from a string, except the decimal point.
LOWER	Converts a string to lowercase
REGEX	Executes a regular expresseion This is an experimental command. It still needs to be tested thoroughly.
REMOVE	Remove("ABC") or Remove (ABC) Removes the string ABC from the string
TITLECASE	Converts the string to Titlecase
TRIM	Removes trailing and preceding spaces
UPPER	Converts the string to uppercase

Note.

Commands can be combined into series and commands are executed from left to right.

For example the CODE { getHMLContent() ; remove ("author") ; trim() ; UPPER()} applied onto  
 "<TD>author</TD><TD> Emile Zola </TD>" will result in "EMILE ZOLA"

Notation

It is advised to put the command parameters between double quotes. You need to escape a double quote by \ (backward slash) if a double quote is a required character of the parameter's value, e.g.  
 ENDPATTERN "<class=\"inventory\_detail\_list\">".

---

## 6.6 FLEX GUI commands

An experimental GUI based java Swing application is available for developing webStraktor scripts in a graphical manner, e.g. dragging and dropping components.

This GUI application stores additional directives in the webStraktor scripts. These GUI directives are all preceded by the comment operator ("—") and all commence by "@FLEX-".

Command	Comment
--@FLEX-COORDINATES	X and Y coordinate of the webStraktor component on the IDE canvas.
--@FLEX-CREATEDAT	Creation timestamp of the webStraktor component
--@FLEX-MODELITEM	Metadata information on a webStraktor component.
--@FLEX-MODIFIEDAT	Last modification timestamp of the webStraktor component.
--@FLEX-PROJECT.CREATED	Creation timestamp of the project (script)
--@FLEX-PROJECT.MAJORVERSION	Major version of the Project
--@FLEX-PROJECT.MINORVERSION	Minor version of the Project
--@FLEX-PROJECT.MODIFIED	Last modification timestamp of the project (script)
--@FLEX-PROJECT.NAME	The name of the webStraktor project (script)
--@FLEX-PROJECT.ROUNDTRIPS	Scripts are created in an iterative manner in the GUI development tool. A roundtrip comprises the graphical development of a webStraktor script; the generation of the script's code and a reverse-engineering step of the code back into GUI format.
--@FLEX-PROJECT.USER	The Operating System username of the person that created the webStraktor project.

---

## 7. Tips and tricks

---

### 7.1 Web Proxy server usage

There might be rightful reasons or even legal grounds for using a web proxy server. This section comments on how to use web proxy servers in webStraktor scripts.

The webStraktor application can be used in combination with various types of web proxy servers, however webStraktor works best on Glype based web proxies. See <http://www.glype.com/> for more information on this type of proxy server.

#### 7.1.1 Web Proxy Server interaction

At the start of a new crawling session, webStraktor will

- randomly select a web proxy server from its web proxy server list or use the web proxy defined on the pre-compiler PROXY directive in a webStraktor script
- contact the web proxy server and fetch its main page
- use the input instructions defined in webStraktor-proxies.xml configuration file and submit the requested URL into the HTML form on the web proxy server's main page

The following text fragment is the logging which was created by webStraktor when trying to access basketball-reference.com via the [www.australia-proxy.com](http://www.australia-proxy.com) web proxy.

---

```
12-Apr 06:45:39-751 FRM - FORM Name=1 Id= Method=POST
Action=includes/process.php?action=update
12-Apr 06:45:39-751 FRM - FORM Name=2 Id= Method= Action=
12-Apr 06:45:39-751 FRM - INPUT Name=u Id=url_textbox Type=text
Value=
12-Apr 06:45:39-751 FRM - INPUT Name= Id=button Type=submit Value=Go!
12-Apr 06:45:39-751 FRM - INPUT Name=encodeURL Id=encodeURL
Type=checkbox Value=
12-Apr 06:45:39-751 FRM - INPUT Name=encodePage Id=encodePage
Type=checkbox Value=
12-Apr 06:45:39-751 FRM - INPUT Name=allowCookies Id=allowCookies
Type=checkbox Value=
12-Apr 06:45:39-751 FRM - INPUT Name=stripJS Id=stripJS Type=checkbox
Value=
12-Apr 06:45:39-751 FRM - INPUT Name=stripObjects Id=stripObjects
Type=checkbox Value=
12-Apr 06:45:39-751 FRM - FORM Name=3 Id= Method= Action=
12-Apr 06:45:39-751 CLN - ACTION RAW[includes/process.php?action=update]
12-Apr 06:45:39-751 CLN - ACTION PROCESSED[http://www.australia-
proxy.com/includes/process.php?action=update]
12-Apr 06:45:39-751 CLN - ==> POST [http://www.australia-
proxy.com/includes/process.php?action=update] params
[http://www.basketball-reference.com/robots.txt]
```

---

In most cases the web proxy responds by a HTTP redirect (302) command. webStrakor will then get the HTML page it is redirected to. This can be observed in the following excerpt from the log file.

---

```
12-Apr 06:45:39-953 CLN - Lastlocation =http://www.australia-  
proxy.com/w.php/2RnYe7io/sCYR73pZ/4HkP3tqr/VnzQ6p18/W0p3nNQg/Pp7jqdar/ONPwR  
0sL/b0/fnorefer  
12-Apr 06:45:39-953 CLN - POST returned http error code -> (302)  
12-Apr 06:45:39-953 CLN - Redirecting to [http://www.australia-  
proxy.com/w.php/2RnYe7io/sCYR73pZ/4HkP3tqr/VnzQ6p18/W0p3nNQg/Pp7jqdar/ONPwR  
0sL/b0/fnorefer]  
12-Apr 06:45:39-953 CLN - ====> FETCHING : http://www.australia-  
proxy.com/w.php/2RnYe7io/sCYR73pZ/4HkP3tqr/VnzQ6p18/W0p3nNQg/Pp7jqdar/ONPwR  
0sL/b0/fnorefer  
12-Apr 06:45:39-953 CLN - Lastlocation =http://www.australia-  
proxy.com/w.php/2RnYe7io/sCYR73pZ/4HkP3tqr/VnzQ6p18/W0p3nNQg/Pp7jqdar/ONPwR  
0sL/b0/fnorefer
```

---

### 7.1.2 Using a web proxy server in webStraktor

It suffices to set the pre-compiler directive PROXY to YES or to define a web proxy server's DNS name to enable web proxy usage on the webStraktor script to crawl web pages via a proxy server.

---

```
REM Transfer the robots.txt file from NYC to $WebStraktorRootDir/Log/nyc-
robots.txt

OUTPUTFILENAME robot-test
CONSOLIDATE      true
KEEPOUTPUTFILE  true
LOGLEVEL        9
SIMULATION      false
PROXY           yes
BROWSER         firefoxw7

MAIN
  REM           Just to test the robot header option
  URL           http://www.nyc.com
END MAIN
```

---

### 7.1.3 Maintaining the proxy list

webStraktor relies on two configuration files to manage web proxies.

- webStraktor-proxy.xml
- active-proxies.xml

See the “Configuration” section in this manual for detailed information on the content of these configuration files

The content of the active-proxies.xml complements and overrules the content of the webStraktor-proxies file.

It is not uncommon that web proxy servers are discontinued. webStraktor therefore uses the active-proxy configuration file to keep track of the availability (or unavailability) of a web proxy server.

You can manually maintain these web proxy server configuration files; however it is advised to use webStraktor for performing this chore.

#### 7.1.3.1 Adding a web proxy server

These are the steps to follow when adding a web proxy server

##### 7.1.3.1.1 Step 1 – Create a web proxy list manually

There are quite a few sites that maintain Glype proxy list and their health state. Visit such a site and make a note of the web proxy servers that look promising to use.



Alternatively you can run the PROG-GlypeList crawler script provide in \$WebStraktorRootDir/Src. A copy of this script is available in the appendix of this document.

Create a text file (e.g. newproxylist.txt) comprising a list of web proxy servers, which you want to assess and add to the webStraktor-proxy list. Store the file on the \$WebStraktorRootDir folder.

For example

---

```
www.thewheatbelly.com
www.supergreenstuff.com
justanotherproxy.org
```

---

**\*\*** You need not specify the www prefix in this file.

#### 7.1.3.1.2 Step 2 – automatically append webStraktor-Proxies

Next you need to run webStraktor in ADDPROXY mode

For example : -D d:\webStraktor -C webStraktor.xml -ADDPROXY newProxyList.txt

webStraktor will assess whether the proxies defined in NewProxyList text file are active or not. Active proxies will be added to the webStraktor-proxy.xml file and the active-proxies.xml file will also be updated to reflect their status.

These are the lines added in webStraktor-Proxies.xml for the supergreenstuff.com proxy.

```
= <Proxy>
    <Host>www.supergreenstuff.com</Host>
    <Action>includes/process.php?action=update</Action>
    <Input>u</Input>
    <Active>yes</Active>
    <Priority>0</Priority>
    <Cookiepolicy>browsers</Cookiepolicy>
    <CookieEnabled>no</CookieEnabled>
</Proxy>
```

Note. webStraktor will make a back-up copy of the existing webStraktor-Proxies.xml file in the \$WebStraktorRoot directory by copying the old webStraktor-Proxies.xml file and adding a timestamp, e.g. webStraktor-Proxies-140412080024.xml. You can revert to the previous state by replacing the recently created webStraktor-Proxies.xml file by the previous webStraktor-Proxies-NNNNNNNNNNNN.xml file.

Note. When webStraktor is running in ADDPROXY mode it writes logging information in the \$WebStraktorRootDir/Log/Add-Proxy.txt file. Use this log file for debugging purposes.

#### **7.1.4 Testing the web proxy server list**

WebStraktor can automatically validate its web proxy server list. It is recommended to occasionally perform this sanity test. Testing the availability of web proxy servers can be achieved by via the following command line options

```
-D d:\webStraktor -C webStraktor.xml -PROXYTEST
```

The validation mechanism is straightforward. webStraktor dynamically creates a basic crawler script that uses the “ProxyTestURL” and “ProxyPassPhrase” information in the webStraktor.xml configuration for each web proxy server on the proxylist. These scripts are put on a queue and the queue is ran past webStraktor. The scripts are generated in the \$WebStraktorRootDir/Src folder, but are removed at the end of the test-run. This is an example web proxy availability assessment script.

---

```
REM Generated Script to test proxies
PROXY www.supergreenstuff.com
LOGLEVEL 9
KEEPOUTPUTFILE NO
CONSOLIDATE NO
SIMULATION NO
OUTPUTFILENAME flexProxyTest

main
  URL http://www.lecourrierdusud.ca
  PASSPHRASE "Votre journal"
end main
```

---

Note. A gentle reminder is added at the end of the log file when the web proxy server list has not be refreshed in a while.

---

## **7.2 Specifying webStraktor FORM constructs**

This section provides guidance for correctly writing webStraktor FORM constructs.

A well working approach for specifying the FORM statement and its attributes is to use the webStraktor log file and inspect the report of on HTML form parameters.

Start by entering a syntactically correct FORM statement in a webStraktor script, next run the script in webStraktor and examine the log information to find the correct form number and the names of its input fields.

### 7.2.1 Example 1

For example enter the following dummy FORM statement

---

```
FORM
  URL          www.xyzabc.com
  FORMNAME     noidea
  INPUT        fieldone=1
END FORM
```

---

This is an excerpt from the webStrakor log file created when you run the above script.

---

```
FORM11-Apr 14:08:08-333 FRM - FORM Name=2 Id=opForm Method=GET
Action=/crrncycnvrtr/cnvrt/
11-Apr 14:08:08-333 FRM -      INPUT    Name=Amount Id=amount Type=text
Value=1
11-Apr 14:08:08-333 FRM -      INPUT    Name=From Id=from Type=select
Value=CAD
11-Apr 14:08:08-333 FRM -      INPUT    Name=To Id=to Type=select Value=JPY
```

---

You now have the required information for modifying the syntactically correct dummy form into a functional FORM statement, i.e. copy the form number, the names of the 3 input fields (amount, to, from) from the log into the script as follows:

---

```
FORM
  URL www.xyzabc.com
  FORMNAME 2
  INPUT amount=1
  INPUT from=EUR
  INPUT to=USD
END FORM
```

---

### 7.2.2 Example 2 Hidden fields

This is one of the 3 HTML forms used on the main page of [www.basketball-reference.com](http://www.basketball-reference.com)

This is the HTML code

---

```
<form method="get" id="f" name="f" action="/player_search.cgi">
  <input type="hidden" name="pid" value="" data-search-id>
  <input x-webkit-speech type="search" placeholder=""
    id="search" name="search" class="search long typeahead">
  <input type="submit" value="Search" class="submit">
</form>
```

---

There is a hidden input field on the HTML form, i.e. "pid". In general you will need to carefully assess whether or not a value is required in that field. webStraktor will by default put an empty value in hidden fields prior to submitting the form.

This is the extract from the script's logfile.

---

```
12-Apr 06:30:24-429 CLN - ====> FORM : http://www.basketball-
reference.com/
12-Apr 06:30:24-445 FRM - FORM Name=f Id=f Method=GET
Action=/player_search.cgi
12-Apr 06:30:24-445 FRM - INPUT Name=pid Id= Type=hidden Value=
12-Apr 06:30:24-445 FRM - INPUT Name=search Id=search Type=search
Value=
12-Apr 06:30:24-445 FRM - INPUT Name= Id= Type=submit Value=Search
12-Apr 06:30:24-445 FRM - FORM Name=f_big Id=f_big Method=
Action=/player_search.cgi
12-Apr 06:30:24-445 FRM - INPUT Name=pid Id= Type=hidden Value=
12-Apr 06:30:24-445 FRM - INPUT Name=search Id=search Type=search
Value=
12-Apr 06:30:24-445 FRM - INPUT Name= Id= Type=submit Value=Search
12-Apr 06:30:24-445 FRM - FORM Name=f_footer Id= Method=GET
Action=/player_search.cgi
12-Apr 06:30:24-445 FRM - INPUT Name=search Id=search_footer
Type=text Value=
12-Apr 06:30:24-445 FRM - INPUT Name= Id= Type=submit Value=Search
12-Apr 06:30:24-445 CLN - ACTION=http://www.basketball-
reference.com/player_search.cgi
12-Apr 06:30:24-445 CLN - ====> FETCHING : http://www.basketball-
reference.com/player_search.cgi?pid=&search=addison
```

---

This is the corresponding webStraktor FORM construct

---

```
FORM
  FORMNAME f
  URL      http://www.basketball-reference.com/
  INPUT    pid=""
  INPUT    search=$1
END FORM
```

---

There is a hidden input field on the HTML form. The webStraktor client will by default put empty values in all hidden fields (see the log line FETCHING for the HTTP query string).

In this case the site does not require a value in the hidden field, so the INPUT pid="" statement could have been omitted in the script. In other cases it might be required to provide a value in the hidden field, e.g. session identifiers.

*Note. Please note that [www.basket-reference.com](http://www.basket-reference.com/robots.txt) robots.txt file disallows to issue the /player\_search.cgi command. The above example is therefore provided for educational and clarification purposes only.*

---

## 7.3 How to use XPATH expressions

The VAR, LINK, BLOB, CUT constructs use different approaches for selecting text in a HTML form. The STARTPATTERN-ENDPATTERN combination is most easy one to use.

The XPATH command however enables to unequivocally locate a text block in the DOM of an HTML page. The following approach can be used to find the exact access path of a text block.

Define a syntactically correct XPATH statement and run webStraktor once. Whenever the XPATH command is used, a file containing the Document Object Tree (DOM) will be created on \$WebRootDir/Temp. The file name comprises the script name and ends with “-report.txt”.

Use this DOM report to determine the XPATH access path.

This is example content of such a -report.txt file

---

```
000751 (1) html[xmlns=http://www.w3.org/1999/xhtml][lang=en]->(1)
body[class=extra-large-keyvisual no-navigation]->(4)
script[type=text/javascript][CONTENT=try {                                var
pageTracker = _gat._getTracker("UA-6612234-1");
pageTracker._trackPageview();                                } catch(err) {}]
000752 (1) html[xmlns=http://www.w3.org/1999/xhtml][lang=en]->(1)
body[class=extra-large-keyvisual no-navigation]
000753 (1) html[xmlns=http://www.w3.org/1999/xhtml][lang=en]
```

---

The various nodes in the DOM and their level and sequence within a level are written into this file. The 751th line's XPath is /html/body/script

---

## 7.4 Web browser signatures

A web client transfers its signature to the web server via the “User-Agent” attribute on the HTTP header. Web servers use this information to create dedicated markup code for the client application.

Firefox running on Linux for example sends “User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:12.0) Gecko/20100101 Firefox/21.0”

Examples

Signature	Type of user agent
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0	Firefox on W7 64Bit
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/21.0	Firefox on Linux 64bit

Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)

Internet Explorer 10  
64Bit

A complete list can be found on <http://www.useragentstring.com/pages/Browserlist/>

The markup code provided might differ depending on the type of user agent. For debugging purposes it therefore advised to configure the webStraktor signature so that its matches that of your preferred web browser.

Use <http://www.useragentstring.com/> to display the User-Agent information of your browser and configure the BROWSER setting in the script's header accordingly.

The default User Agent will be used if the BROWSER pre-compiler is not set in a script. The default User Agent can be configured in the useragents.xml file by defining a browser named "default". The default User Agent in the configuration file will overrule webStraktor's internal default User Agent, which is a Firefox browser running on NT (Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.1.6) Gecko/20091201 Firefox/3.5.6 (.NET CLR 3.5.30729))

### 7.4.1 Bypassing the HTTP Client whilst debugging

By setting the SIMULATION pre-compiler directive webStraktor's HTTP client will no longer access the world wide web, instead it will read previously retrieved web pages which are stored in the \$WebStraktorRootDir\In folder. Use this features whilst coding the various constructs and fine-tuning the CODE instructions.

This is an example of the header section of a crawler script.

---

```
LOGLEVEL          9
SIMULATION        TRUE
PROXY             false
OUTPUTFILENAME    CurrencyExchagneRate
CONSOLIDATE       true
KEEPOUTPUTFILE    true
BROWSER           ipad
ROBOTS            false
```

---

## 7.5 How to use HTTP queries

In the World Wide Web, a query string is the part of a uniform resource locator (URL) that contains data to be passed to web applications. The main use of query strings is to refer to a HTML form. The query string is composed of a series of field-value pairs. Within each pair, the field name and value are separated by an equals sign, '='. The series of pairs is separated by the ampersand '&' (see Wikipedia and other sources).

In the event that a web from can be called via a GET method you might opt to use a query string instead of a webStraktor FORM construct and merely define a URL.

The following example illustrates this. Two command line parameters are pasted into the HTTP query string.

---

MAIN

URL [http://www.mysite.com/search?q=\\$1&category=\\$2&dt=results\\_page](http://www.mysite.com/search?q=$1&category=$2&dt=results_page)

PASSPHRASE "Your query returned »

CALL ProcOne

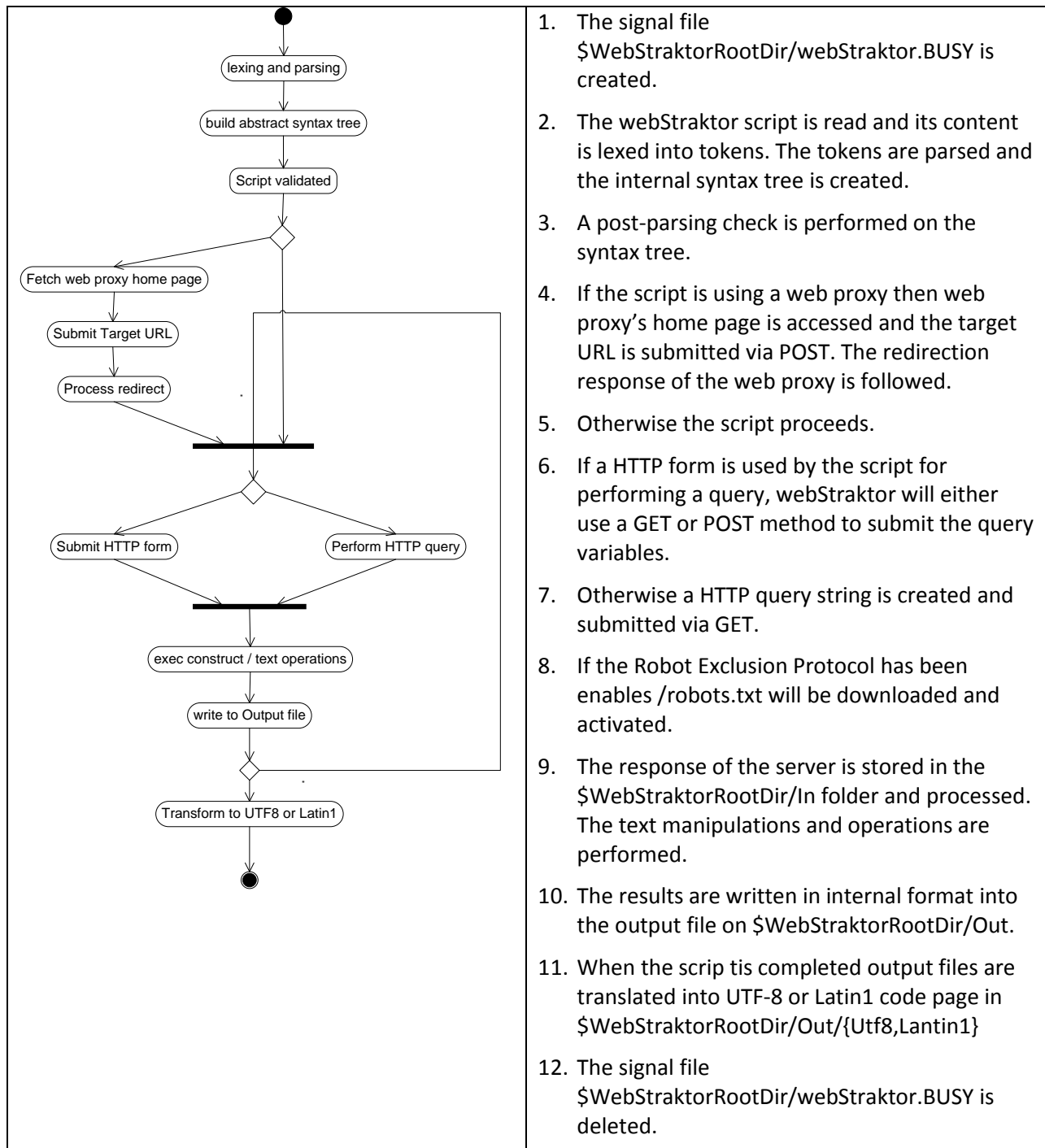
END MAIN

---

## 8. Developer notes

### 8.1 Activity diagram

This section comments briefly on the tasks that are typically performed by webStraktor whilst stepping through a single script.





---

## 8.2 The Robot Exclusion Protocol

Detailed information on the Robot Exclusion Protocol can be found at [http://en.wikipedia.org/wiki/Robots\\_exclusion\\_standard](http://en.wikipedia.org/wiki/Robots_exclusion_standard), webStraktor implements this protocol to a limited extend.

The Robot Exclusion Protocol can be enabled in main webStraktor.xml configuration file by

- Setting the “AssessRobotsTxt” toggle to YES, this will download the /robots.txt file prior to any other operation on a web site. The contents of the /robots.txt file will be stored in \$WebStraktorRootDir/In/<URL-Of-The-website>-robots.txt for further inspection
- Setting the “OverrideRobotSpecification” switch to NO, this will ensure that the webStraktor HTTPClient module will adhere to the instructions defined in the /Robots.txt file (*in so far the webStraktor client correctly implements those*).

You can override the global “AssessRobotsTxt” setting in the script’s header via the pre-compiler directive ROBOTS {true,false,default}. Some web sites (freedb.org for example) do not have a /robots.txt file, so when the global settings is enabled webStraktor will encounter an error when trying to open the first page.

webStraktor also assesses the parameter “Crawl-Delay” parameter in the /robots.txt file. This defines the delay period between consecutive web page retrievals. This parameter is also defined in the CrawlerDelay setting in the main webStraktor.xml configuration file. By default it is set to 4 second but will be overruled by the “Crawl-Delay” settings.

The following extract from the log file shows how webStraktor will adhere to the robots.txt instructions on [www.basket-reference.com](http://www.basket-reference.com).

---

```
11-Apr 20:12:08-610 CLN - ====> FETCHING : http://www.basketball-
reference.com/robots.txt
11-Apr 20:12:09-320 CLN - Content Encoding [null] Content MimeType
[text/plain; charset=utf-8]
11-Apr 20:12:09-320 CLN - Dumped 309 bytes onto d:\webStraktor\In\www-
basketball-reference-com-robots.txt] in Charset [utf-8]
11-Apr 20:12:09-320 CLN - ROBOT SPECS
11-Apr 20:12:09-320 CLN -   - Disallow File : /player_search.cgi
11-Apr 20:12:09-320 CLN -   - Disallow File : /boxscores/index.cgi
11-Apr 20:12:09-330 CLN -   - Disallow Folder : /blazers/
11-Apr 20:12:09-330 CLN -   - Disallow Folder : /dump/
11-Apr 20:12:09-330 CLN -   - Disallow Folder : /fc/
11-Apr 20:12:09-330 CLN -   - Disallow Folder : /my/
11-Apr 20:12:09-330 CLN -   - Disallow File : /7103
11-Apr 20:12:09-330 CLN - PATH [/] is allowed to be accessed by host
www.basketball-reference.com
```

---

In the above example webStraktor is not allowed to issue a /player\_search.cgi command.

This is the content of the [www.basket-reference.com/robots.txt](http://www.basket-reference.com/robots.txt) file as it is transferred into \$WebStraktorRootDir/Log/ www-basketball-reference-com-robots.txt file.

---

```
User-agent: *
Disallow: /player_search.cgi
Disallow: /boxscores/index.cgi
Disallow: /blazers/
Disallow: /dump/
Disallow: /fc/
Disallow: /my/
Disallow: /7103

# Disallow the plagiarism.org robot, www.slysearch.com
User-agent: SlySearch
Disallow: /                #Will disallow or robot from all urls on your site
```

---

In the above example the SlySearch bot is denied complete access to the site.

---

### 8.3 Internal character set

webStraktor uses an internal format to store any HTML content. This encoding mechanism can be observed on the files in the WebStraktorRootDir/In directory.

The webStraktor encoding schema is straightforward, as soon as the byte value exceeds 128( 0xF0) data is transformed into the following pattern : **&#HHHH;**

- &# is the marker to identify the beginning of a character encoded webStraktor format
- HHHH is the hex formatted value of the character (double byte)
- ; is the character concluding the internal format.

For example &#9642; is a square caret.

---

### 8.4 Syntax tree and object model

webStraktor scripts are deconstructed into a sequence of tokens by the webStraktorParseModel class. These tokens are parsed and translated into a hierarchy of programming language independent objects, functions and instructions. This is known as the “abstract syntax tree”. The structure of the object model hierarchy and the attributes of each object can be inspected in the log file.

This is the abstract syntax tree of the Random House crawler script.

---

```
MAIN URL [http://www.randomhouse.com]
FOLDER      = [d:\webStraktor]
OUTPUTFILE   = [random-books]
SOURCEFILE   = [d:\webStraktor\Src\PROG-RandomHouse.txt]
```

---

---

```

LOGLEVEL          = 9
PROXY             = false
isSIMULATION      = false
KEEPOUTPUTFILE    = true
CONSOLIDATE       = true
INIT OK           = true
FUNCTION (main) Start=null End=null Passphrase=Book Results URL=null MAXITERATIONS=-1
TRACE=0
  INSTRUCTION (FORM-94) Label=FORM-94 Tipe=FORM FormName=1
  FormURL=http://www.randomhouse.com Fields=[title_subtitle_auth=$1] TRC=0
  INSTRUCTION (pageTwo) Label=pageTwo Tipe=FOREACH Start=null End=null Xpath=null
  Regex=null Call=null CMD=null TRC=0 RfrncdFE=null
  FUNCTION (pageTwo) Start=null End=null Passphrase=null URL=null MAXITERATIONS=-1
  TRACE=0
  INSTRUCTION (CUT-60) Label=CUT-60 Tipe=CUT Start=<div class="list_container">
  End=<!--end list_container--> Xpath=null Regex=null Call=null CMD=null TRC=0
  RfrncdFE=null
  INSTRUCTION (FOREACH-66) Label=FOREACH-66 Tipe=FOREACH Start=null End=null
  Xpath=null Regex=null Call=null CMD=null TRC=0 RfrncdFE=null
  FUNCTION (FOREACH-66) Start=<div class="book"> End=</div><!--end book-->
  Passphrase=null URL=null MAXITERATIONS=-1 TRACE=0
  INSTRUCTION (BookTitle) Label=BookTitle Tipe=VAR Start=<h3> End=</h3> Xpath=null
  Regex=null Call=null CMD=GetHTMLContent();Trim() TRC=0 RfrncdFE=null
  COMMAND : GetHTMLContent
  COMMAND : TRIM
  INSTRUCTION (BookDetailInk) Label=BookDetailInk Tipe=LINK Start=<h3> End=</h3>
  Xpath=null Regex=null Call=BookDetail CMD=Get("href");Trim();GetField(2,"\"") TRC=0
  RfrncdFE=BookDetail
  COMMAND : GET[href]
  COMMAND : TRIM
  COMMAND : GetFIELD[2,""]
  FUNCTION (BookDetail) Start=null End=null Passphrase=<!--end commerce_box_header--
  > URL=null MAXITERATIONS=-1 TRACE=0
  INSTRUCTION (CUT-21) Label=CUT-21 Tipe=CUT Start=<!--end commerce_box_header-->
  End=</ul> Xpath=null Regex=null Call=null CMD=null TRC=0 RfrncdFE=null
  INSTRUCTION (TitleTwo) Label=TitleTwo Tipe=VAR Start=<li><strong>
  End=</strong></li> Xpath=null Regex=null Call=null CMD=null TRC=0 RfrncdFE=null
  INSTRUCTION (Author) Label=Author Tipe=VAR Start=Written by End=</a> Xpath=null
  Regex=null Call=null CMD=GetHTMLContent();TRIM() TRC=0 RfrncdFE=null
  COMMAND : GetHTMLContent
  COMMAND : TRIM
  INSTRUCTION (Format) Label=Format Tipe=VAR Start=Format: End=| Xpath=null
  Regex=null Call=null CMD=TRIM() TRC=0 RfrncdFE=null
  COMMAND : TRIM
  INSTRUCTION (ISBNNumber) Label=ISBNNumber Tipe=VAR Start=ISBN: End=</li>
  Xpath=null Regex=null Call=null CMD=KEEPNUMBER();TRIM() TRC=0 RfrncdFE=null
  COMMAND : KeepNumber
  COMMAND : TRIM
  INSTRUCTION (Price) Label=Price Tipe=VAR Start=<strong>Our Price:
  End=</strong></li> Xpath=null Regex=null Call=null CMD=KEEPDECIMALS();TRIM() TRC=0
  RfrncdFE=null
  COMMAND : UNKNOWN
  COMMAND : TRIM

```

---

## 8.5 Logging

Log information is written in the \$WebStraktorRootDir/Log folder. Each script creates its own log file. The name of the script followed by an enumeration of the command line parameters.

The log is self-pruning. When the number of loglines exceeds a predefined threshold the log will get truncated to 50% of the threshold.

The following extract of log shows two consecutive runs.

---

```
13-Apr 18:44:17-365 PRS - WRITER->          </BookDetail>
13-Apr 18:44:17-365 PRS - KEPT TEMP FILE [d:\webStraktor\Temp\random-books.txt-
4.txt]
13-Apr 18:44:17-365 PRS - WRITER->          </booklist>
13-Apr 18:44:17-365 PRS - KEPT TEMP FILE [d:\webStraktor\Temp\random-books.txt-
2.txt]
13-Apr 18:44:17-365 PRS - WRITER-> </main>
13-Apr 18:44:17-365 PRS - WRITER-></random-books>
13-Apr 18:44:17-365 CLN - Shutdown httpclient
13-Apr 18:44:17-396 iCV - -->    Read [1685] bytes on
d:\webStraktor\Out\Ascii\random-books-cons.txt
13-Apr 18:44:17-396 iCV - --> Written [1685] bytes on
d:\webStraktor\Out\utf8\random-books-cons.xml
13-Apr 18:44:17-396 iCV - Latin1 Conversion
13-Apr 18:44:17-412 iCV - -->    Read [1685] bytes on
d:\webStraktor\Out\utf8\random-books-cons.xml
13-Apr 18:44:17-521 ===== Logger stopped =====
13-Apr 18:47:30-219 ===== Logger started =====
13-Apr 18:47:30-235 SRC - LOGLEVEL 9
13-Apr 18:47:30-235 SRC - SIMULATION false
13-Apr 18:47:30-235 SRC - PROXY false
13-Apr 18:47:30-235 SRC - OUTPUTFILENAME random-books
13-Apr 18:47:30-235 SRC - CONSOLIDATE true
13-Apr 18:47:30-235 SRC - KEEPOUTPUTFILE true
```

---

The log level can be set between 0 and 9. When configured to run in level 0 webStraktor will only create loglines for errors encountered. Loglevel 9 provides the most detailed logging information. It is recommended to use level 9 for debugging and level 5 for normal operation.

A logline comprises

- A timestamp. The format of the logline timestamp (DATEFORMAT) can be set in the main configuration file
- An abbreviation of the component that created the logline, e.g. SRC is webStraktorParseModel, PRS is webStraktorParseURL, iCV is the code page convertor, etc.
- The payload of the logline

Java exceptions and exception stack traces are assumed to be captured and written into the script's log. Occasionally some exceptions might not be caught, so keep an eye on stdout when webStraktor is running.

---

## 8.6 Trace

webStraktor generates detailed trace information whilst stepping through the various instructions of a script.

The trace information is written to the \$WebStraktorRootDir/Log/<script-name>-trace.txt file.

The trace lines are also written onto a TCP/IP queue upon completion of each step in the script. The webstraktor monitor application has a TCT/IP listener embedded . The monitor application pops the

trace lines from the TCP/IP queue and as the script progresses the monitor will display in real-time statistical information.

This is an excerpt of the trace information produced when running the “PROG-RandomHouse” script.

The trace module starts by dumping the abstract syntax tree of the script and then continues by providing timestamped information per model item.

---

```

START
--HEADER  webStraktor
--@FLEX-PID      5012
--@FLEX-VERSION  1.0
--@FLEX-BUILD    2014-04-11
--@FLEX-EXEETIME 13-Apr 18:47:30-297
--@FLEX-SCRIPTNAME d:\webStraktor\Src\PROG-RandomHouse.txt
--@FLEX-CMDLINEPARAMS 1=franzen
--@FLEX-LOGFILE    d:\webStraktor\Log\random-books-log.txt
--@FLEX-LOGLEVEL   9
--@FLEX-PROXY
--@FLEX-KEEPFILE   d:\webStraktor\Out\Ascii\random-books-franzen.txt
--@FLEX-MODELITEMINFO      Trace ID      Parent Trace ID Component Name
Component Type
--@FLEX-MODELITEM 100      -1 main main
--@FLEX-MODELITEM 101      100 main-PassPhrase  passphrase
--@FLEX-MODELITEM 102      100 main-URL          url
--@FLEX-MODELITEM 103      100 FORM-94         form
--@FLEX-MODELITEM 104      100 pageTwo        call
--@FLEX-MODELITEM 105      100 pageTwo        procedure
--@FLEX-MODELITEM 106      105 pageTwo-PassPhrase passphrase
--@FLEX-MODELITEM 107      105 pageTwo-URL     url
--@FLEX-MODELITEM 108      105 CUT-60         cut
--@FLEX-MODELITEM 109      105 FOREACH-66      foreach
--@FLEX-MODELITEM 110      105 booklist        foreach
--@FLEX-MODELITEM 111      110 booklist-PassPhrase  passphrase
--@FLEX-MODELITEM 112      110 booklist-URL      url
--@FLEX-MODELITEM 113      110 BookTitle        variable
--@FLEX-MODELITEM 114      110 BookDetailInk     link
--@FLEX-MODELITEM 115      110 BookDetail        procedure
--@FLEX-MODELITEM 116      115 BookDetail-PassPhrase  passphrase
--@FLEX-MODELITEM 117      115 BookDetail-URL     url
--@FLEX-MODELITEM 118      115 CUT-21          cut
--@FLEX-MODELITEM 119      115 TitleTwo         variable
--@FLEX-MODELITEM 120      115 Author           variable
--@FLEX-MODELITEM 121      115 Format            variable
--@FLEX-MODELITEM 122      115 ISBNNumber       variable
--@FLEX-MODELITEM 123      115 Pricevariable
--END HEADER
--@FLEX-SYNCHRO milli 1397411250141 micro 45567905240
      main|C| 100|      Start| 45568075306|
      FORM-94|C| 103|      Start| 45568075926|
      FORM-94|H| 103|      Ok| 45569360921| 45568077053|
45569360650| 59219|http://www.randomhouse.com|d:\webStraktor\In\random-
books.txt-1.txt
      FORM-94|C| 103|      Stop| 45569361544|
      FORM-94|H| 103|      Ok| 45569665497| 45569362226|
45569664726| 22766|http://www.randomhouse.com|d:\webStraktor\In\random-
books.txt-1.txt
      FORM-94|C| 103|      Stop| 45569666086|FORM fetched

```

---

---

```

    FORM-94|C| 103| Start| 45569667024|
      main|C| 100| Start| 45569670077|
    main-PassPhrase|C| 101| Start| 45569670385|
    main-PassPhrase|C| 101| Stop| 45569670991|Book Results
    pageTwo|C| 104| Start| 45569673145|
    pageTwo|F| 104| Ok| 45569673434|d:\webStraktor\Temp\random-
books.txt-0.txt
    pageTwo|C| 105| Start| 45569673762|
    CUT-60|C| 108| Start| 45569674029|
    CUT-60|F| 108| Ok| 45569677218|d:\webStraktor\Temp\random-
books.txt-1.txt
    CUT-60|C| 108| Stop| 45569677631|
    FOREACH-66|C| 109| Start| 45569679030|
    FOREACH-66|F| 109| Ok| 45569679410|d:\webStraktor\Temp\random-
books.txt-2.txt
    booklist|C| 110| Start| 45569679813|
    BookTitle|C| 113| Start| 45569681249|

```

---

**Note.** The TCP/IP client-server modules are based on Java NIO. New I/O is a collection of APIs that provide features for intensive I/O operations.

**Note.** WebStraktor NIO communication occurs per default on port TCP/IP 8001. In the event that other applications are also using this port, you will need to manually update the “IPCPort” settings in the major configuration file.

---

## 9. Step by Step tutorial

---

### 9.1 Introduction

This is a step by step tutorial of the webStraktor application and its scripting language. The tutorial is conceived as case study.

Random House is the largest general-interest trade book publisher in the world. It has been owned since 1998 by the German private media corporation Bertelsmann and has become the umbrella brand for Bertelsmann book publishing (Wikipedia).

In our case study we will connect to the “Random House” web site and search for books written by a specific author.

---

### 9.2 Pre-requisites

Foremost bear in mind to be gentle on the web sites of which you are planning to extract data, i.e. adhere to the robot exclusion protocol, observe the crawler delay period and do not massively unload data. It is good etiquette to use webStraktor as if a human user performed the data extraction.

The minimal skill that you should master is to be able to inspect the HTML code. In FireFox perform a right-click and in the pop-up menu select “View Source Code” for inspecting HTML code.

You will need to have a basic understanding of the HTML syntax. You should at least be able to relate the appearance of components displayed on your web browser to the underlying HTML code.

---

### 9.3 Fetching a web page

The following script will fetch the home page of [www.randomhouse.com](http://www.randomhouse.com) without using a web proxy server (PROXY false) and will store the results in \$WebStraktorRootDir/out/Utf8/random-books.out. The various output files will be consolidated and will not be deleted from the Out folder. The loglevel is set to the most detailed level. The script performs a HTTP GET of the main page of [www.randomhouse.com](http://www.randomhouse.com).

Create this script in \$WebStraktorRootdir/Src and name it PROG-RandomBooks.txt.

---

```
LOGLEVEL          9
SIMULATION        false
PROXY             false
OUTPUTFILENAME    random-books
CONSOLIDATE       true
KEEPOUTPUTFILE   true
ROBOTS            default

MAIN
  URL www.randomhouse.com
  PASSPHRASE justtesting
END MAIN
```

---

Open the queue.txt file in the \$WebStraktorRootdir/Src directory and adapt it as follows.

---

```
--  
-- queue file  
--  
PROG-RandomHouse.txt
```

---

Save the queue.txt file and run webStraktor by executing the \$WebStraktorRootDir/Java/webStraktor/webStraktorRun.bat batch file. This batch file uses the Ant build file and runs webStraktor in Q mode.

Inspect the log file created in \$WebStraktorRootDir/log. This file is named “random-book-log.txt.” In particular inspect the section on the Robot Exclusion protocol. In the event that the “AssessRobotTxt” settings are enabled in webStraktor.xml you should be able to detect the following loglines.

---

```
13-Apr 12:36:56-743 CLN - ROBOT SPECS  
13-Apr 12:36:56-743 CLN - - Disallow Folder : /cgi-bin/  
13-Apr 12:36:56-743 CLN - - Disallow Folder : /mt32/  
13-Apr 12:36:56-743 CLN - - Disallow Folder : /delrey/starwars/catalog/  
13-Apr 12:36:56-743 CLN - - Disallow Folder :  
/livinglanguage/registration  
.. etc ..
```

---


The script verifies whether the page contains the passphrase “justtesting”. Obviously this will not be case and webStraktor will terminate. Look for the abort message at the end of the log file.

---

## 9.4 Adding a FORM

In this section we will add a FORM construct to the script in order to initiate a Search operation. We want to provide the name of an author via the command line (or queue file) and have webStraktor substituting the value on the “rhw\_search\_field” HTML input field by the first command line parameter.

This is the input form as it appears on the web site

A screenshot of a web search interface. It features a light blue rectangular container. Inside, there is a white text input field on the left. To its right are two buttons: a small orange button with the text 'GO' and a larger orange button with the text 'ADV. SEARCH'.

This is the HTML code for the above FORM



```

<form action="/book/search/search.php" method="get">
  <div id="rhw_search_left"></div>
  <div class="rhw_rfloat">
    
  </div>
  <div id="rhw_search_buttons">
    <a class="rollover"><input type="image" src="/art/bw06/base/search_go.png"
alt="go" /></a><a
href="/catalog/" class="rollover"></a>
  </div>
  <div id="rhw_search_field_div">
    <input type="text" name="title_subtitle_auth" id="rhw_search_field" />
  </div>
</form>

```

These are the form attributes that webStraktor detects.

```

FORM Name=2 Id= Method=POST
Action=/browse.php?u=dnRa3NlCPSC6hB5ln4xC1VyY3IInkwGRKF19pyh7P96h0%2F2IjAz
BtAkVUj2%2F&b=5
13-Apr 12:48:17-504 FRM - INPUT Name=convertGET Id=rhw_search_left
Type=hidden Value=1
13-Apr 12:48:17-504 FRM - INPUT Name= Id=rhw_search_field_div
Type=image Value=
13-Apr 12:48:17-504 FRM - INPUT Name=title_subtitle_auth
Id=rhw_search_field Type=text Value=

```

There is a mismatch between the action `/browse.php?u` in the log and `/book/search/search.php` on the form. This is because a web proxy was used. Web proxies substitute the action statements on web forms.

Adapt the script so that it looks like the following code.

---

```

---
-- webStraktor Demo script
-- Random House crawler script
-- 13 April 2014
--
LOGLEVEL          9
SIMULATION        false
PROXY             false
OUTPUTFILENAME    random-books
CONSOLIDATE       true
KEEPOUTPUTFILE   true
BROWSER          default
ROBOTS           true
--
--
MAIN
  FORM
    FORMNAME 1
    URL www.randomhouse.com
    INPUT title_subtitle_auth=$1
  END FORM

  PASSPHRASE "Book Results"
END MAIN

```

---

*Just in case you might wonder who Mr. Franzen might be. According to Wikipedia. Jonathan Earl Franzen (born August 17, 1959) is an American novelist and essayist. His 2001 novel, *The Corrections*, a sprawling, satirical family drama, drew widespread critical acclaim, earned Franzen a National Book Award, was a Pulitzer Prize for Fiction finalist, earned a James Tait Black Memorial Prize. His most recent novel is *Freedom* (2010).*

You also need to adapt the queue file in order to provide the name of an author as a command line parameter. Open the queue.txt file in the \$WebStraktorRootdir/Src directory and modify it as follows.

---

```

--
-- queue file
--
PROG-RandomHouse.txt franzen

```

---

The script should now run without errors and the PASSPHRASE should correctly resolve. Browse through the random-books-log.txt logfile to familiarize yourself with the logging format.

There should also be a random-books-franzen.xml file in the three \$WebStraktorRootDir/Output directories, apart from a standard XML headerline these files should all be empty.

---

## 9.5 Creating an Output file

In our previous script webStraktor was instructed to perform a Search operation on Jonathan Franzen. In most cases the result is list of books that match the query's criterion. In this section we will use a FOR-EACH to step through each of the books returned by the web site and store its title in the output file.

The following script will do the trick.

---

```
LOGLEVEL          9
SIMULATION        false
PROXY             false
OUTPUTFILENAME    random-books
CONSOLIDATE       true
KEEPOUTPUTFILE   true
ROBOTS            true
BROWSER           default
--
--
PROCEDURE pageTwo
  CUT
    STARTPATTERN "<div class=\"list_container\">"
    ENDPATTERN   "<!--end list_container-->"
  END CUT
  --
  FOREACH
    LABEL          booklist
    STARTPATTERN   "<div class=\"book\">"
    ENDPATTERN     "</div><!--end book-->"
    --
    VAR BookTitle
      STARTPATTERN "<h3>"
      ENDPATTERN   "</h3>"
      CODE {GetHTMLContent();Trim()}
    END VAR
    --
  END FOREACH
END PROCEDURE

MAIN
  FORM
    FORMNAME 1
    URL www.randomhouse.com
    INPUT title_subtitle_auth=$1
  END FORM
  PASSPHRASE "Book Results"
  CALL pageTwo
END MAIN
```

---

In this script we CALL procedure “pageTwo” and start by cutting out a subsection of the HTML returned. This subsection starts by <div class="list\_container"> and ends with "<!--end list\_container-->".

In this subsection we iterate through the various HTML code blocks that are stored between "<div class="book"> and </div><!--end book--> and we extract the title which is defined between <h3> and </h3>. We remove the HTML tags via “GetHTMLContent() and TRIM() the result.

This is the output that you should get when you re-run webStraktor

```
<xml version="1.0" encoding="UTF-8" ?>

<!--  CMDLINEPARAMS  [1=frenzen]  -->
```

```
<!-- URL [http://www.randomhouse.com] -->
<!-- Charset [UTF-8] -->
<!-- Started at [13-Apr-14 18:47:31:905] -->
<random-books>
  <main>
    <pageTwo>
      <booklist>
        <BookTitle>Busy Kitties</BookTitle>
      </booklist>
      <booklist>
        <BookTitle>The Laughing Policeman</BookTitle>
      </booklist>
    </pageTwo>
  </main>
</random-books>
```

---

## 9.6 Web crawling

Oddly enough Mr. Franzen is not the author of the above books. He wrote the introduction or he contributed to the creation of these books published by Random House. At this stage we might want to consider fine-tuning our script to get relevant information on the author of the books.

Each element of the search results on the 2<sup>nd</sup> page comprises a URL that is referring to a page containing detailed book information. The following script will extract these links (via the LINK construct) and it will fetch the web page the link is pointing to. It will then call another procedure "Bookdetail" to extract the detailed information.

```
--
-- webStraktor Demo script
-- Random House crawler script
-- 13 April 2014

--
--
LOGLEVEL          9
SIMULATION        false
PROXY             false
OUTPUTFILENAME    random-books
CONSOLIDATE       true
KEEPOUTPUTFILE   true
BROWSER           default
--
--

PROCEDURE BookDetail

    PASSPHRASE "<!--end commerce_box_header-->"

    CUT
        STARTPATTERN "<!--end commerce_box_header-->"
        ENDPATTERN    "</ul>"
    END CUT

    VAR TitleTwo
        STARTPATTERN "<li><strong>"
        ENDPATTERN    "</strong></li>"
    END VAR

    VAR Author
        STARTPATTERN "Written by"
        ENDPATTERN    "</a>"
        CODE           {GetHTMLContent();TRIM()}
    END VAR

    VAR Format
        STARTPATTERN "Format:"
        ENDPATTERN    "|"
        CODE           {TRIM()}
    END VAR

    VAR ISBNNumber
        STARTPATTERN "ISBN:"
        ENDPATTERN    "</li>"
        CODE           {KEEPNUMBER();TRIM()}
    END VAR
```

```

VAR Price
  STARTPATTERN "<strong>Our Price:"
  ENDPATTERN   "</strong></li>"
  CODE         {KEEPDECIMALS();TRIM()}
END VAR

END PROCEDURE

PROCEDURE pageTwo

  CUT
    STARTPATTERN "<div class=\"list_container\">"
    ENDPATTERN   "<!--end list_container-->"
  END CUT
  --
  --
  FOREACH

    LABEL          booklist
    STARTPATTERN   "<div class=\"book\">"
    ENDPATTERN     "</div><!--end book-->"
    --
    VAR BookTitle
      STARTPATTERN "<h3>"
      ENDPATTERN   "</h3>"
      CODE {GetHTMLContent();Trim()}
    END VAR
    --
    LINK BookLink
      STARTPATTERN "<h3>"
      ENDPATTERN   "</h3>"
      CODE {Get("href");Trim();GetField(2,"\"") }
      CALL BookDetail
      STORE BookDetailInk
    END LINK
    --
  END FOREACH
END PROCEDURE

MAIN
  FORM
    FORMNAME 1
    URL www.randomhouse.com
    INPUT title_subtitle_auth=$1
  END FORM
  PASSPHRASE "Book Results"
  CALL pageTwo

END MAIN

```

The “BookLink” LINK construct performs the following logic. This is the HTML code that comprises the HREF element of the HTTP anchor element.

---

```
<h3><a href = "/book/168101/the-laughing-policeman-by-maj-sjowall-and-per-wahloo">The Laughing Policeman</a></h3>
```

---

The “BookLink” LINK construct cuts the text between <h3> and </h3> and then processes the CODE {Get("href");Trim();GetField(2,"\"") } statement, i.e. it gets the content of the value of the “href” HTML

attribute, trims the returned text and concludes by retrieving the 2nd part of string separated by a double quote. This results in the access path `"/ book/168101/the-laughing-policeman-by-maj-sjowall-and-per-wahloo"`, which is then appended to the main URL [www.randomhouse.com](http://www.randomhouse.com) to create a valid URL, which is subsequently accessed by a HTTP GET request.

If you run this script through webStraktor you should now obtain the following output file in the `$WebStraktorRootDir/Out/Utf8` folder.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- CMDLINEPARAMS [1=franken] -->
<!-- URL [http://www.randomhouse.com] -->
<!-- Charset [UTF-8] -->
<!-- Started at [13-Apr-14 18:47:31:905] -->
<random-books>
  <main>
    <pageTwo>
      <booklist>
        <BookTitle>Busy Kitties</BookTitle>
        <BookDetail>
          <TitleTwo>Busy Kitties</TitleTwo>
          <Author>John Schindel</Author>
          <Format>Board</Format>
          <ISBNNumber>9781582461304</ISBNNumber>
          <Price>6.99</Price>
        </BookDetail>
      </booklist>
      <booklist>
        <BookTitle>The Laughing Policeman</BookTitle>
        <BookDetail>
          <TitleTwo>The Laughing Policeman</TitleTwo>
          <Author>Maj Sjowall</Author>
          <Format>Trade Paperback</Format>
          <ISBNNumber>9780307390509</ISBNNumber>
          <Price>14.95</Price>
        </BookDetail>
      </booklist>
    </pageTwo>
  </main>
</random-books>
```

---

## 9.7 Downloading images

### 9.7.1 Concluding the script

The detailed information page also has thumbnail image of the book's cover. In this final scenario we will extend the webStraktor script to download these cover images .

This is the corresponding HTML code for displaying the thumbnail images on the overview page.

---

```
</a>
```

---

It therefore suffices to add the following BLOB construct to PROCEDURE pageTwo

---

```
PROCEDURE pageTwo
  CUT
    STARTPATTERN "<div class=\"list_container\">"
    ENDPATTERN   "<!--end list_container-->"
  END CUT
  --
  FOREACH
    LABEL          booklist
    STARTPATTERN   "<div class=\"book\">"
    ENDPATTERN     "</div><!--end book-->"
    --
    VAR BookTitle
      STARTPATTERN "<h3>"
      ENDPATTERN   "</h3>"
      CODE {GetHTMLContent();Trim()}
    END VAR
    --
    BLOB BookCover
      STARTPATTERN "<img class=\"cover\">"
      ENDPATTERN   "</a>"
      CODE         {Get("ref");Trim();GetField(2,"\"") }
    END BLOB
    --
    LINK BookLink
      STARTPATTERN "<h3>"
      ENDPATTERN   "</h3>"
      CODE {Get("href");Trim();GetField(2,"\"") }
      CALL BookDetail
      STORE BookDetailInk
    END LINK
    --
  END FOREACH
END PROCEDURE
```

---



There will be two image files on the \$WebStraktorRootDit/out/Blob folder once you have re-executed the script: random-books-franzen-1.jpeg and random-books-franzen-2.jpeg.

### 9.7.2 Fine-tuning the script

The following modification (MAXITERATIONS 5) should be applied to the script in the event that you might want to limit the number of books returned to 5.

---

```
PROCEDURE pageTwo
  CUT
    STARTPATTERN "<div class=\"list_container\">"
    ENDPATTERN   "<!--end list_container-->"
  END CUT
  --
  FOREACH
    LABEL          booklist
    STARTPATTERN   "<div class=\"book\">"
    ENDPATTERN     "</div><!--end book-->"
    MAXITERATIONS 5
    --
    VAR BookTitle
      STARTPATTERN "<h3>"
      ENDPATTERN   "</h3>"
      CODE {GetHTMLContent();Trim()}
    END VAR
    --
    BLOB BookCover
      STARTPATTERN "<img class=\"cover\">"
      ENDPATTERN   "</a>"
      CODE         {Get("ref");Trim();GetField(2,"\"") }
    END BLOB
    --
    LINK BookLink
      STARTPATTERN "<h3>"
      ENDPATTERN   "</h3>"
      CODE {Get("href");Trim();GetField(2,"\"") }
      CALL BookDetail
      STORE BookDetailInk
    END LINK
    --
  END FOREACH
END PROCEDURE
```

---

In order to test the above script you will need to modify the queue file so that it brings back abundant data. The American novelists E.L. Doctorow and John Updike have most of their oeuvre published by Random House, so when you query on their names there will be plenty of books.

The following queue file will twice invoke the PROG-RandomHouse.txt script but per run will it limit the number of books returned to 5.

---

```
--  
-- queue file  
--  
--PROG-RandomHouse.txt franzen  
PROG-RandomHouse.txt updike  
PROG-RandomHouse.txt doctorow
```

---

There should be two additional output files in `$WebStraktorRootDir/Out` once `webStraktor` finishes, i.e. `random-books-doctorow.xml` and `random-books-updike.xml`. Have a look at the consolidated output file `random-books-cons.xml` also. There should be 3 distinct sections in the consolidated output file, one for Franzen, one for Doctorow and one for Updike.

Additional modification. Change `MAXITERATION` 5 to

- `SKIP 1,2` should you want to skip the first 2 books in the result set
- `OCCURS 1,3,5` should you want to fetch only the first, third and fifth book in the result set

Once you have debugged the script you might consider to set `ROBOTS` to false.

*Just in case you might wonder who Mr. Doctorow and Updike might be. According to Wikipedia:*

*Edgar Lawrence "E. L." Doctorow (born January 6, 1931) is an American author. He is known internationally for his unique works of historical fiction.*

*John Hoyer Updike (March 18, 1932 – January 27, 2009) was an American novelist, poet, short story writer, art critic, and literary critic. Updike's most famous work is his "Rabbit" series.*

---

## 10. Java sources

The following table defines the Java sources constituting the webStraktor package

Filename	Description
nioClient.java	Inter Process Communication between webStraktor and the webStraktor monitor tool
parseCommand.java	parses the webStraktor commands (CMD)
parseFunction.java	parses the webStraktor functions
parseInstruction.java	Parses the webStraktor instructions
webStraktorDateTime.java	DateTime utility
webStraktorHTML.java	Performs the XPath queries
webStraktorHTTPClient.java	The Apache HTTPClient is used to fetch web content via HTTP. The client also processes HTTP redirects (302), submits data into HTTP forms and manages cookies.
webStraktorIConv.java	Code Page convertor webStraktor uses a proprietary internal code page format to store data fetched from the web (webStraktorHTPClient). This component translates the internal format into ASCII, Latin1 or UTF-8 when the Output files are created.
webStraktorLogger.java	Logging component
webStraktorMain.java	Main component
webStraktorParseController.java	Controller component
webStraktorParseForm.java	Parses a HTML form
webStraktorParseModel.java	Parses the webStraktor script and creates an internal model of the script.
webStraktorParseQueue.java	Reads and executes the webStraktor scripts on the queue
webStraktorParseURL.java	Parses an URL
webStraktorPrintStream.java	Utility for writing data onto files
webStraktorSAX.java	Light-weight SAX component (XML parser)
webStraktorSettings.java	Configuration component
webStraktorStringUtil.java	Utility for character strings
webStraktorTrace.java	Trace component, used in conjunction with the webStraktor

	monitor
webStraktorUtil.java	This is a tool-chest providing various utilities.

---

## 11. webStraktor GUI

---

### 11.1 Introduction

The webStraktor GUI application enables to graphically develop crawler scripts. At this stage the application is still being developed and therefore this component is released “as-is”.

---

### 11.2 Compilation

Execute the compileViaAnt file under \$WebRootDir/Java/webStraktorGUI. This will compile the application and run it. Before you run it you will need to update the environment variables JAVA\_HOME, ANT\_HOME and WEBSTRAKTOR\_HOME.

You might need to adapt the Build.xml file to set the working directory and the IPCPort.

---

```
<target name="webStraktorMonitorMain">
    <java classname="webStraktorMonitor.webStraktorMonitorMain"
failonerror="true" fork="yes">
        <arg line=" d:\webStraktor\Gui\webStraktorGui.xml "/>
        <classpath refid="webStraktorMonitor.classpath"/>
    </java>
</target>
```

---

The first command line parameter is the webStraktor GUI configuration file.



Click on PROG-RandomHouse.txt. This script will now be reverse engineered and displayed.

You will now be able to inspect the Rand House crawler script. Click on the components on the development canvas, the component tree or the source code.

Use the slider to zoom in or out.

Click on “Generate”. A source code file will now be generated.

### 11.4.2 Create a script

Click on the “File” menu and select “New project”. Enter a name in the dialog box.

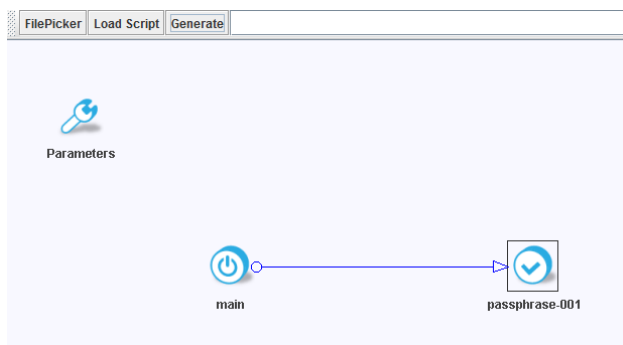
You will get a blank canvas.

Click on the “Main” component in the component pane, release the mouse button and click on the canvas. The Main component will now be added to the development canvas.

Click on the “Passphrase” component in the component window, release the mouse button and click on the development canvas. The Passphrase component will be added to the canvas.

Click on the Main component on the canvas, hold down the shift key and click on the Passphrase component. A connector will be created between both components.

Click on “Generate”, this will create the webStraktor script.



---

## 12. webStraktor monitor

---

### 12.1 Introduction

The webStraktor monitor application enables to monitor the progress of running webStraktor script. At this stage the application is still being developed and therefore this component is released “as-is”.

---

### 12.2 Compilation

Execute the compileViaAnt file under \$WebRootDir/Java/webStraktorMonitor. This will compile the application and run it. Before you run it you will need to update the environment variables JAVA\_HOME, ANT\_HOME and WEBSTRAKTOR\_HOME.

You might need to adapt the Build.xml file to set the working directory and the IPCPort.

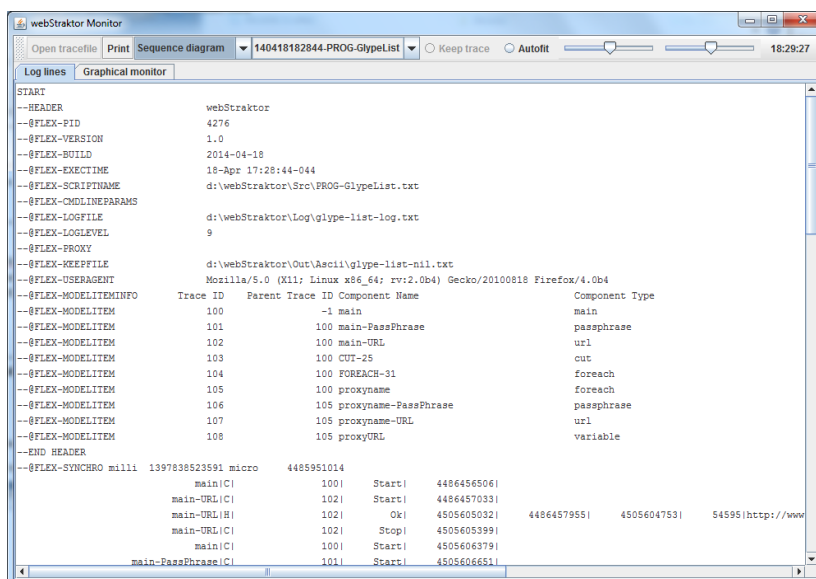
```
<target name="webStraktorMonitorMain">
    <java classname="webStraktorMonitor.webStraktorMonitorMain"
failonerror="true" fork="yes">
        <arg line="d:\webStraktor 8001"/>
        <classpath refid="webStraktorMonitor.classpath"/>
    </java>
</target>
```

The first command line parameter is the working directory. The TCP/IP IPC port can be set via the 2<sup>nd</sup> command line parameter.

---

### 12.3 Quick introduction

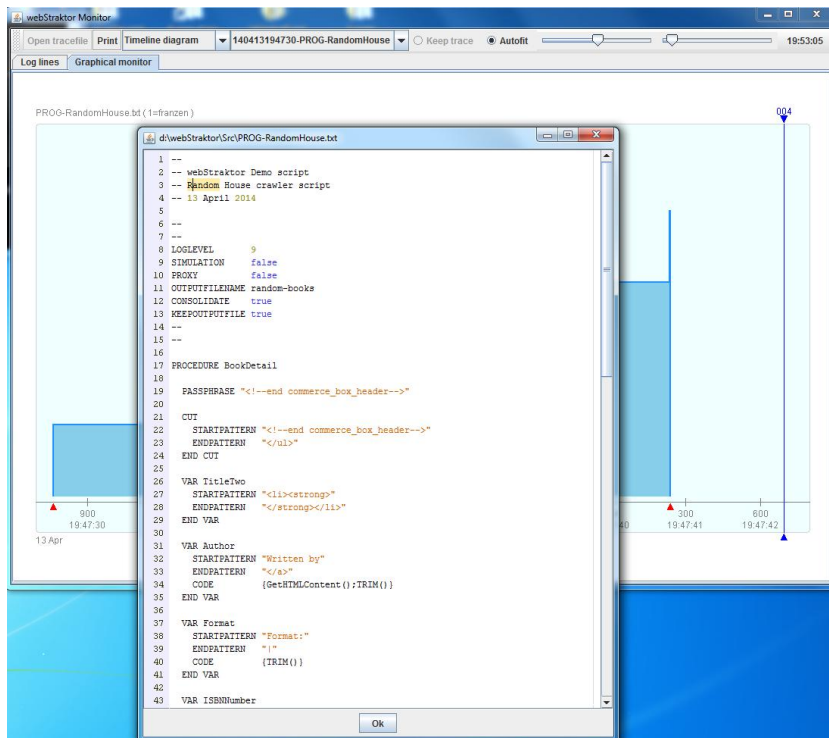
Open the monitor application and run a webStraktor script. This is a screenshot of the GlypeList script.



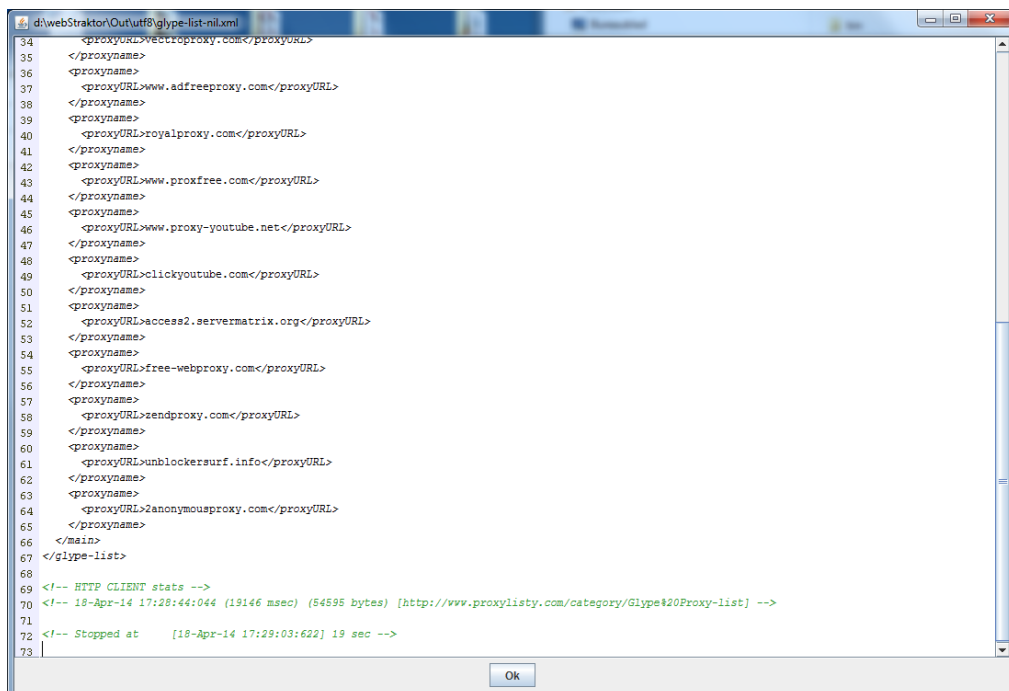


The screenshot displays the webStraktor Monitor application. The top menu bar includes 'Open tracefile', 'Print', 'Sequence diagram', and 'Log lines'. Below the menu bar, there is a status bar with 'Keep trace' and 'Autofit' options, and a timestamp '20:38:18'. The main area shows a graphical sequence diagram with vertical dashed lines representing components and horizontal solid lines representing messages. The components are labeled 'main PassPhrase', 'booklist', 'booklist-PassPhrase', and 'FOREACH-22'. The diagram illustrates a sequence of interactions between these components, with messages flowing from left to right and top to bottom.

webStraktor Manual



## GlypeProxy output screen



---

## 13. Known issues

These are the known bugs

Issue Number	Description	ETA	Resolved
BUG01	Robot Exclusion protocol often works incorrectly, sometimes disallowed access paths are not recognized.		
BUG02	FOREACH appears to malfunction on exceedingly long HTM markup lines.		
BUG03			
BUG04			

---

## 14. Further enhancements

---

### 14.1 Code enhancements

The following features will be implemented shortly

Request Number	Description	ETA	Resolved
CR001	Support for ajax and HTTP-XML protocols		
CR002	Create new function : RemoveInternalFormat()		
CR003	Automated regression test module		
CR004			
CR005			
CR006			

---

### 14.2 Outstanding items to be documented in this manual

These are the outstanding topics to be documented.

Fix	Description	ETA	Resolved
FX001			

---

## Appendix A - webStraktor in a larger application context

The webStraktor application was purposely created to fit into an overall application that is geared towards the management of a comic book collection.

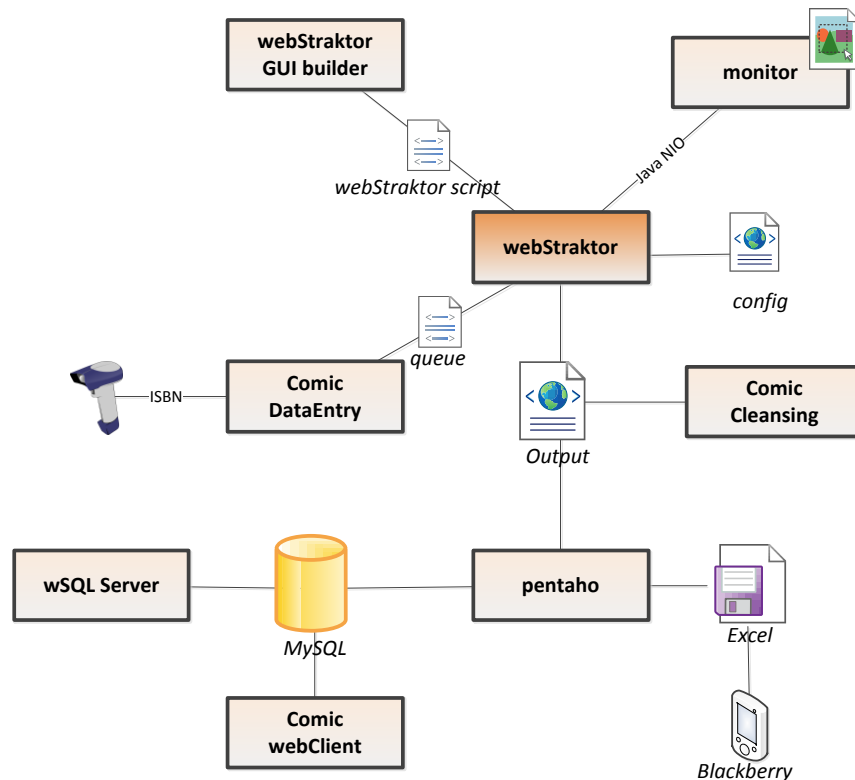


Figure 10 webStraktor application context

The Comic DataEntry application is a Java Swing application that uses a barcode scanner to read the ISBN number from a comic book's back cover. The related comic book information (series, editor, penciller, writer, etc) is extracted from the World Wide Web via a series of webStraktor scripts.

The comic book information extracted from the web is stored into webStraktor XML output files. These file are harmonized by the Comic Book Cleansing application prior to being loaded into a MySQL database by the Pentaho Data Integration component (<http://www.pentaho.com/product/data-integration>). A flexible Comic Book Data Model has been developed. A Web SQL Server application (<http://sourceforge.net/projects/websqlclient>) is used to manage the comic book data in the MySQL Comic Book Data Model. The Pentaho Data Integration tool is also used to extract MySQL Comic Book information into an Excel spreadsheet, which is occasionally synchronized onto a Blackberry smartphone.

The following picture is a screenshot of the Comic Book Data Entry application. One enters an ISBN number and the application will then use webStraktor to search for details of that comic book, including a picture of the front-cover of the book.



---

## Appendix B – Example scripts

### WORLDCAT

```
//
// worldcat parser script
// 4 November 2012
//
--
LOGLEVEL          9
SIMULATION        false
PROXY             true
OUTPUTFILENAME    WordCat
CONSOLIDATE       true
KEEPOUTPUTFILE   true
--
--
PROCEDURE booklist

    PASSPHRASE "<!DOCTYPE html PUBLIC"

    FOREACH

        LABEL book

        STARTPATTERN "class=\"menuElem\""
        ENDPATTERN   "</tr>"

        VAR bookname
        STARTPATTERN "class=\"name\""
        ENDPATTERN   "</div>"
        STORE        bookname
        CODE          {GET (strong),TRIM(),Lower()}
        END VAR

        LINK booklink
        STARTPATTERN "class=\"name\""
        ENDPATTERN   "</div>"
        STORE        booklink
        CODE          {GET ("href"),TRIM()}
        CALL          two
        IGNORE
        END LINK

    END FOREACH

END PROCEDURE

PROCEDURE two

    FOREACH

        LABEL bookdetail
        MAXITERATIONS 100
        OCCURS        1,2,3

        STARTPATTERN "<div id=\"bibdata\""
        ENDPATTERN   "</div>"

        VAR AlbumAuthor
        STARTPATTERN "bib-author-cell\""
```

```

    ENDPATTERN    </td>
    STORE         AlbumAuthor
    CODE          {getHTMLcontent(),TRIM()}
END VAR

VAR AlbumPublisher
    STARTPATTERN  bib-publisher-row">
    ENDPATTERN    </td>
    STORE         AlbumPublisher
    CODE          {remove("<th>Uitgever:</th>"),remove("<td id=\"bib-publisher-
cell\">"),remove("<th>Publisher:</th>"),trim()}
    END VAR

VAR AlbumType
    STARTPATTERN  "<span class='itemType'>"
    ENDPATTERN    </span>
    STORE         AlbumType
    CODE          {trim() }
    END VAR

VAR AlbumAuthorRegex
    REGEX         "bib-author-cell\">.*<a href"
    STORE         AlbumAuthorRegex
    CODE          {getHTMLcontent(),TRIM(),remove("bib-author-cell\"") }
    END VAR

END FOREACH
END PROCEDURE

MAIN
URL              http://www.worldcat.org/search?q=ti%3A$1+au%3A$2
PASSPHRASE      "<!DOCTYPE html PUBLIC"
CALL            booklist
END MAIN

```



## WORLDCAT – ExtractGlype

```
--
-- webStraktor Demo script
-- extraxts Glype web proxy servers
-- 18 April 2014

--
--
LOGLEVEL          9
SIMULATION        false
PROXY             true
OUTPUTFILENAME    glype-list
CONSOLIDATE       true
KEEPOUTPUTFILE    true
BROWSER           firefoxLinux
--
--

MAIN

URL http://www.proxylisty.com/category/Glype%20Proxy-list

PASSPHRASE "Glype Proxy list for proxies using Glype script"

CUT
  STARTPATTERN "title='Glype Proxy List'>Glype Proxy</a></td>"
  ENDPATTERN   "</table>"
END CUT

FOREACH
  LABEL proxynome
  STARTPATTERN "title='Not working? Report it'></a></td>"
  ENDPATTERN   "</tr>"
  MAXITERATIONS 20

  VAR proxyURL
  STARTPATTERN "Visit"
  ENDPATTERN   "</a>"
  CODE
  {getField(2,">");remove("http://");remove("https://");TRIM();}
  ENDVAR

END FOREACH

END MAIN
```

## List of acronyms

Acronym	Comment
API	Application programming Interface
ASCII	American Standard Code for Information Interchange
BLOB	Binary Large Object
CMD	Command
DOM	Document Object Model
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
HTTP	Hyper Tekst Transfer Protocol
IDE	Integrated Development Environment
ISBN	International Standard Book Number
JAR	Java Archive
Latin1	ISO-8859-1 character set
NIO	New Input Output
PID	Process Identifier
REGEX	Regular Expression
REM	Remark
RFC	Request for Comment
SAX	Simple API for XML
SRC	Source
TCP/IP	Transfer Control Protocol – Internet Protocol
TGT	Target
URL	Uniform Resource Locator
UTF8	Universal Character Set Transformation Format
VAR	Variable
XML	Extended Markup Language
XPATH	XML Path

In loving memory of Jacqueline F .