

Reference Manual

Generated by Doxygen 1.6.1

Sat Apr 17 18:07:08 2010

Contents

1	Blog Link Crawler	1
2	Todo List	3
3	Bug List	5
4	Class Index	7
4.1	Class List	7
5	Class Documentation	9
5.1	crwlib::Node Class Reference	9
5.2	tempcawl::Node Class Reference	10
5.2.1	Detailed Description	11
5.2.2	Member Function Documentation	11
5.2.2.1	__init__	11
5.2.2.2	cat_links	11
5.2.2.3	count	11
5.2.2.4	crawl	11
5.2.2.5	dupCheck	12
5.2.2.6	export_csv	12
5.2.2.7	exportMatrix	13
5.2.2.8	insert	13
5.2.2.9	printMatrix	13
5.2.2.10	walkCrawl	14
5.2.2.11	walkPrint	14
5.2.3	Member Data Documentation	14
5.2.3.1	links	14
5.2.3.2	site	14
5.4	node Class Reference	16
5.4.1	Detailed Description	16

5.4	node Class Reference	16
5.4.1	Detailed Description	16
5.5	v2-blog-link-crawler::Node Class Reference	17
5.5.1	Detailed Description	18
5.5.2	Member Function Documentation	18
5.5.2.1	__init__	18
5.5.2.2	cat_links	18
5.5.2.3	count	18
5.5.2.4	crawl	18
5.5.2.5	dupCheck	19
5.5.2.6	export_csv	19
5.5.2.7	exportMatrix	20
5.5.2.8	insert	20
5.5.2.9	printMatrix	20
5.5.2.10	walkCrawl	21
5.5.2.11	walkPrint	21
5.5.3	Member Data Documentation	21
5.5.3.1	links	21
5.5.3.2	site	21
5.6	tempcrawl::parseLinks Class Reference	22
5.6.1	Member Function Documentation	22
5.6.1.1	handle_starttag	22
5.9	parseLinks Class Reference	25
5.9.1	Detailed Description	25
5.8	v2-blog-link-crawler::parseLinks Class Reference	24
5.8.1	Member Function Documentation	24
5.8.1.1	handle_starttag	24
5.9	parseLinks Class Reference	25
5.9.1	Detailed Description	25
5.10	crwlib::parseLinks Class Reference	26

Chapter 1

Blog Link Crawler

This program crawls web pages and keeps track of hyperlinks. Main class is [Node](#).

--->[Node](#)<---

Author:

Daniel Snider

Date:

2010

Note:

work in progress

most functions work best when the root [node](#) is the object in question

Global Variables:

crawled_sites = list of unique crawled sites. I have no idea if this is efficient but it is very required. This is used to ensure a site is not crawled twice. This is also used for the rows of the adjacency matrix.

unique_sites = numebr of unique sites is usually different than the number of crawled sites because not all collected sites have been crawled. This is used as the columns of the adjacency matrix.

blacklist = sites that should be ignored. There are a bunch of pesky wordpress sites that I need to block.

Todo

create a root global variable to reduce confusion of what object should be called

Chapter 2

Todo List

page **Blog Link Crawler** create a root global variable to reduce confusion of what object should be called
create a root global variable to reduce confusion of what object should be called

Member **tempcawl::Node::crawl** remove useless variables nu and ll.

Member **tempcawl::Node::dupCheck** instead of taking just the first link to a domain found, we could possibly always use the plain domain
when determining the domain of a full URL change algorithm to determine the end of the domain with ".com" or ".net" and a "/". every domain seems to follow the rule of ".something/" for the root of the site, i.e. the domain.

Member **tempcawl::Node::export_csv** each row should begin with the parent site but does not currently I don't believe. Priority: low

Member **tempcawl::Node::exportMatrix** check logic. Priority: high

Member **tempcawl::Node::printMatrix** check logic. Priority: low

Member **tempcawl::Node::walkCrawl** multithread

Member **v2-blog-link-crawler::Node::crawl** remove useless variables nu and ll.

Member **v2-blog-link-crawler::Node::dupCheck** instead of taking just the first link to a domain found, we could possibly always use the plain domain
when determining the domain of a full URL change algorithm to determine the end of the domain with ".com" or ".net" and a "/". every domain seems to follow the rule of ".something/" for the root of the site, i.e. the domain.

Member **v2-blog-link-crawler::Node::export_csv** each row should begin with the parent site but does not currently I don't believe. Priority: low

Member **v2-blog-link-crawler::Node::exportMatrix** check logic. Priority: high

Member **v2-blog-link-crawler::Node::printMatrix** check logic. Priority: low

Member **v2-blog-link-crawler::Node::walkCrawl** multithread

Member **v2-blog-link-crawler::parseLinks::handle_starttag** add "typepad" to the list of acceptable domains

Chapter 3

Bug List

Member [v2-blog-link-crawler::Node::export_csv](#) breaks when nuber of nodes gets over like 30

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

crwlib::Node	9
tempcawl::Node	10
node (This class is a graph data structure and is used to build a graph of links between websites)	16
node (This class is a graph data structure and is used to build a graph of links between websites)	16
v2-blog-link-crawler::Node	17
tempcawl::parseLinks	22
parseLinks (This class is for parsing html pages)	25
v2-blog-link-crawler::parseLinks	24
parseLinks (This class is for parsing html pages)	25
crwlib::parseLinks	26

Chapter 5

Class Documentation

5.1 crwlib::Node Class Reference

Public Member Functions

- def `__init__`
- def `insert`
- def `crawl`
- def `walkCrawl`
- def `walk_print`
- def `cat_links`
- def `count`
- def `dupCheck`
- def `export_csv`

Public Attributes

- `links`
- `site`

The documentation for this class was generated from the following file:

- `crwlib.py`

5.2 tempcawl::Node Class Reference

Public Member Functions

- def [__init__](#)
The constructor.
- def [insert](#)
Inserts a site into the connection graph.
- def [crawl](#)
Downloads a webpage and collects hyperlinks.
- def [walkCrawl](#)
Crawls every link in a node's links.
- def [walkPrint](#)
Prints every [node](#).
- def [cat_links](#)
Special formating used to export sites to .csv.
- def [count](#)
Counts the number of nodes.
- def [dupCheck](#)
Checks if a website is already linked by a site.
- def [export_csv](#)
Creates a spreadsheet of the graph.
- def [printMatrix](#)
Prints an adjecency matrix to console.
- def [exportMatrix](#)
Creates a file containing an adjacency matrix in Matlab format.

Public Attributes

- [links](#)
A list of nodes.
- [site](#)
Contains the website as a string.

5.2.1 Detailed Description

Author:

Daniel Snider

5.2.2 Member Function Documentation

5.2.2.1 def tempcawl::Node::__init__ (*self*, *newsite*)

The constructor. When a new [node](#) is created the links array is set to empty and the website string is set to be the newsite string that was passed in.

Parameters:

self The object pointer.

newsite The string URL of the new website.

Return values:

None

5.2.2.2 def tempcawl::Node::cat_links (*self*)

Special formating used to export sites to .csv.

Parameters:

self The object pointer.

Return values:

str Returns all the links that are linked by a site in one string

5.2.2.3 def tempcawl::Node::count (*self*)

Counts the number of nodes.

Parameters:

self The object pointer.

Return values:

int The number of [node](#) objects in the graph.

5.2.2.4 def tempcawl::Node::crawl (*self*, *nu*, *ll*)

Downloads a webpage and collects hyperlinks. Also checks for duplicates.

Note:

HTMLParser is used and is very strict, probably too strict and we are missing some sites. Documentation on this:
<http://www.python-forum.org/pythonforum/viewtopic.php?f=19&t=9348>

Parameters:

self The object pointer.

nu,ll Useless variables. Once needed for attempting to multithread.

Return values:

None

Todo

remove useless variables nu and ll.

5.2.2.5 def tempcawl::Node::dupCheck (*self*, *newsite*)

Checks if a website is already linked by a site. i.e. disallows duplicates in a node's "links" list. and disallows the new site to be it's self

Parameters:

self The object pointer.

newsite The string URL of the new website.

Return values:

bool true if website is not already present, false if website is already present

Todo

instead of taking just the first link to a domain found, we could possibly always use the plain domain when determining the domain of a full URL change algorithm to determine the end of the domain with ".com" or ".net" and a "/". every domain seems to follow the rule of ".something/" for the root of the site, i.e. the domain.

5.2.2.6 def tempcawl::Node::export_csv (*self*)

Creates a spreadsheet of the graph. Creates the file links.csv. It is a spreadsheet. Each row will contain the list of sites linked by a site.

Parameters:

self The object pointer. Should be the root [node](#) if you want the whole graph created

Return values:

None

Todo

each row should begin with the parent site but does not currently I don't believe. Priority: low

5.2.2.7 def tempcawl::Node::exportMatrix (*self*)

Creates a file containing an adjacency matrix in Matlab format. The adjacency matrix has a row for each crawled site and a column for each unique site found (which includes crawled sites and child nodes, i.e. non-crawled sites).

Parameters:

self The object pointer. Should be the root [node](#) if you want the whole graph created

Return values:

None

Todo

check logic. Priority: high

5.2.2.8 def tempcawl::Node::insert (*self*, *newsite*)

Inserts a site into the connection graph. Inserts the site into the links[] array for the current (self) site Also checks for duplicates. i.e. won't crawl if the input site has already been crawled

Parameters:

self The object pointer.

newsite The string URL of the new website.

Return values:

None

5.2.2.9 def tempcawl::Node::printMatrix (*self*)

Prints an adjacency matrix to console. Prints out 1s and 0s in a grid with each unique site on the horizontal and vertical and where a website intersects with a website that it links the to, the link is represented by 1. The matrix should be read horizontally

Parameters:

self The object pointer. Should be the root [node](#) if you want the whole graph created

Return values:

None

Todo

check logic. Priority: low

5.2.2.10 `def tempcawl::Node::walkCrawl (self)`

Crawls every link in a node's links. Prints a "." for each site that is crawled

Parameters:

self The object pointer.

Return values:

None

Todo

multithread

5.2.2.11 `def tempcawl::Node::walkPrint (self, root)`

Prints every [node](#). Prints to console each site and the site that linked to it in the form: www.Parent site --> www.Child site

Parameters:

self The object pointer.

root Parent site so to keep track of who the parent is for more informative print

Return values:

None

5.2.3 Member Data Documentation

5.2.3.1 `tempcawl::Node::links`

A list of nodes. Each [node](#) represents a website.

5.2.3.2 `tempcawl::Node::site`

Contains the website as a string.

The documentation for this class was generated from the following file:

- tempcawl.py

5.3 node Class Reference

This class is a graph data structure and is used to build a graph of links between websites.

5.3.1 Detailed Description

This class is a graph data structure and is used to build a graph of links between websites.

The documentation for this class was generated from the following file:

- v2-blog-link-crawler.py

5.4 node Class Reference

This class is a graph data structure and is used to build a graph of links between websites.

5.4.1 Detailed Description

This class is a graph data structure and is used to build a graph of links between websites.

The documentation for this class was generated from the following file:

- v2-blog-link-crawler.py

5.5 v2-blog-link-crawler::Node Class Reference

Public Member Functions

- def `__init__`
The constructor.
- def `insert`
Inserts a site into the connection graph.
- def `crawl`
Downloads a webpage and collects hyperlinks.
- def `walkCrawl`
Crawls every link in a node's links.
- def `walkPrint`
*Prints every *node*.*
- def `cat_links`
Special formating used to export sites to .csv.
- def `count`
Counts the number of nodes.
- def `dupCheck`
Checks if a website is already linked by a site.
- def `export_csv`
Creates a spreadsheet of the graph.
- def `printMatrix`
Prints an adjecency matrix to console.
- def `exportMatrix`
Creates a file containing an adjacency matrix in Matlab format.
- def `exportMultiLineMatrix`
- def `exportCrawled_sites`
- def `exportUnique_sites`

Public Attributes

- `links`
A list of nodes.
- `site`
Contains the website as a string.

5.5.1 Detailed Description

Author:

Daniel Snider

5.5.2 Member Function Documentation

5.5.2.1 `def v2-blog-link-crawler::Node::__init__ (self, newsite)`

The constructor. When a new [node](#) is created the links array is set to empty and the website string is set to be the newsite string that was passed in.

Parameters:

self The object pointer.

newsite The string URL of the new website.

Return values:

None

5.5.2.2 `def v2-blog-link-crawler::Node::cat_links (self)`

Special formating used to export sites to .csv.

Parameters:

self The object pointer.

Return values:

str Returns all the links that are linked by a site in one string

5.5.2.3 `def v2-blog-link-crawler::Node::count (self)`

Counts the number of nodes.

Parameters:

self The object pointer.

Return values:

int The number of [node](#) objects in the graph.

5.5.2.4 `def v2-blog-link-crawler::Node::crawl (self, nu, ll)`

Downloads a webpage and collects hyperlinks. Also checks for duplicates.

Note:

HTMLParser is used and is very strict, probably too strict and we are missing some sites. Documentation on this:
<http://www.python-forum.org/pythonforum/viewtopic.php?f=19&t=9348>

Parameters:

self The object pointer.

nu,ll Useless variables. Once needed for attempting to multithread.

Return values:

None

Todo

remove useless variables nu and ll.

5.5.2.5 def v2-blog-link-crawler::Node::dupCheck (self, newsite)

Checks if a website is already linked by a site. i.e. disallows duplicates in a node's "links" list. and disallows the new site to be it's self

Parameters:

self The object pointer.

newsite The string URL of the new website.

Return values:

bool true if website is not already present, false if website is already present

Todo

instead of taking just the first link to a domain found, we could possibly always use the plain domain when determining the domain of a full URL change algorithm to determine the end of the domain with ".com" or ".net" and a "/". every domain seems to follow the rule of ".something/" for the root of the site, i.e. the domain.

5.5.2.6 def v2-blog-link-crawler::Node::export_csv (self)

Creates a spreadsheet of the graph. [DEPRECIATED] Creates the file links.csv. It is a spreadsheet. Each row will contain the list of sites linked by a site.

Parameters:

self The object pointer. Should be the root [node](#) if you want the whole graph created

Return values:

None

Todo

each row should begin with the parent site but does not currently I don't believe. Priority: low

Bug

breaks when number of nodes gets over like 30

5.5.2.7 def v2-blog-link-crawler::Node::exportMatrix (*self*)

Creates a file containing an adjacency matrix in Matlab format. The adjacency matrix has a row for each crawled site and a column for each unique site found (which includes crawled sites and child nodes, i.e. non-crawled sites).

Parameters:

self The object pointer. Should be the root [node](#) if you want the whole graph created

Return values:

None

Todo

check logic. Priority: high

5.5.2.8 def v2-blog-link-crawler::Node::insert (*self*, *newsite*)

Inserts a site into the connection graph. Inserts the site into the links[] array for the current (self) site Also checks for duplicates. i.e. won't crawl if the input site has already been crawled

Parameters:

self The object pointer.

newsite The string URL of the new website.

Return values:

None

5.5.2.9 def v2-blog-link-crawler::Node::printMatrix (*self*)

Prints an adjacency matrix to console. Prints out 1s and 0s in a grid with each unique site on the horizontal and vertical and where a website intersects with a website that it links the to, the link is represented by 1. The matrix should be read horizontally

Parameters:

self The object pointer. Should be the root [node](#) if you want the whole graph created

Return values:

None

Todo

check logic. Priority: low

5.5.2.10 def v2-blog-link-crawler::Node::walkCrawl (*self*)

Crawls every link in a node's links. Prints a "." for each site that is crawled

Parameters:

self The object pointer.

Return values:

None

Todo

multithread

5.5.2.11 def v2-blog-link-crawler::Node::walkPrint (*self*, *root*)

Prints every [node](#). Prints to console each site and the site that linked to it in the form: www.Parent site --> www.Child site

Parameters:

self The object pointer.

root Parent site so to keep track of who the parent is for more informative print

Return values:

None

5.5.3 Member Data Documentation

5.5.3.1 v2-blog-link-crawler::Node::links

A list of nodes. Each [node](#) represents a website.

5.5.3.2 v2-blog-link-crawler::Node::site

Contains the website as a string.

The documentation for this class was generated from the following file:

- v2-blog-link-crawler.py

5.6 tempcawl::parseLinks Class Reference

Public Member Functions

- `def __init__`
The constructor.
- `def handle_starttag`
This function parses the tags of the website and creates new nodes for each site found.

Public Attributes

- `node`
I added this so that when crawling is happening and a new site is found, the `node` object being crawled can be accessed and the new site can be added in the appropriate place.

5.6.1 Member Function Documentation

5.6.1.1 `def tempcawl::parseLinks::handle_starttag (self, tag, attrs)`

This function parses the tags of the website and creates new nodes for each site found. The code was found from page 170 of Python Phrasebook

The documentation for this class was generated from the following file:

- `tempcawl.py`

5.7 parseLinks Class Reference

This class is for parsing html pages.

5.7.1 Detailed Description

This class is for parsing html pages.

The documentation for this class was generated from the following file:

- tempcawl.py

5.8 v2-blog-link-crawler::parseLinks Class Reference

Public Member Functions

- def [__init__](#)
The constructor.
- def [handle_starttag](#)
This function parses the tags of the website and creates new nodes for each site found.

Public Attributes

- [node](#)
I added this so that when crawling is happening and a new site is found, the [node](#) object being crawled can be accessed and the new site can be added in the appropriate place.

5.8.1 Member Function Documentation

5.8.1.1 def v2-blog-link-crawler::parseLinks::handle_starttag (*self*, *tag*, *attrs*)

This function parses the tags of the website and creates new nodes for each site found. The code was found from page 170 of Python Phrasebook

[Todo](#)

add "typepad" to the list of acceptable domains

The documentation for this class was generated from the following file:

- v2-blog-link-crawler.py

5.9 parseLinks Class Reference

This class is for parsing html pages.

5.9.1 Detailed Description

This class is for parsing html pages.

The documentation for this class was generated from the following file:

- tempcawl.py

5.10 crwlib::parseLinks Class Reference

Public Member Functions

- def `__init__`
- def `handle_starttag`

Public Attributes

- `node`

The documentation for this class was generated from the following file:

- `crwlib.py`

Index

- `__init__`
 - `tempcawl::Node`, [11](#)
 - `v2-blog-link-crawler::Node`, [18](#)
- `cat_links`
 - `tempcawl::Node`, [11](#)
 - `v2-blog-link-crawler::Node`, [18](#)
- `count`
 - `tempcawl::Node`, [11](#)
 - `v2-blog-link-crawler::Node`, [18](#)
- `crawl`
 - `tempcawl::Node`, [11](#)
 - `v2-blog-link-crawler::Node`, [18](#)
- `crwlib::Node`, [9](#)
- `crwlib::parseLinks`, [26](#)
- `dupCheck`
 - `tempcawl::Node`, [12](#)
 - `v2-blog-link-crawler::Node`, [19](#)
- `export_csv`
 - `tempcawl::Node`, [12](#)
 - `v2-blog-link-crawler::Node`, [19](#)
- `exportMatrix`
 - `tempcawl::Node`, [12](#)
 - `v2-blog-link-crawler::Node`, [19](#)
- `handle_starttag`
 - `tempcawl::parseLinks`, [22](#)
 - `v2-blog-link-crawler::parseLinks`, [24](#)
- `insert`
 - `tempcawl::Node`, [13](#)
 - `v2-blog-link-crawler::Node`, [20](#)
- `links`
 - `tempcawl::Node`, [14](#)
 - `v2-blog-link-crawler::Node`, [21](#)
- `node`, [15](#), [16](#)
- `parseLinks`, [23](#), [25](#)
- `printMatrix`
 - `tempcawl::Node`, [13](#)
 - `v2-blog-link-crawler::Node`, [20](#)
- `site`
 - `tempcawl::Node`, [14](#)
 - `v2-blog-link-crawler::Node`, [21](#)
- `tempcawl::Node`, [10](#)
 - `__init__`, [11](#)
 - `cat_links`, [11](#)
 - `count`, [11](#)
 - `crawl`, [11](#)
 - `dupCheck`, [12](#)
 - `export_csv`, [12](#)
 - `exportMatrix`, [12](#)
 - `insert`, [13](#)
 - `links`, [14](#)
 - `printMatrix`, [13](#)
 - `site`, [14](#)
 - `walkCrawl`, [13](#)
 - `walkPrint`, [14](#)
- `tempcawl::parseLinks`, [22](#)
 - `handle_starttag`, [22](#)
- `v2-blog-link-crawler::Node`, [17](#)
 - `__init__`, [18](#)
 - `cat_links`, [18](#)
 - `count`, [18](#)
 - `crawl`, [18](#)
 - `dupCheck`, [19](#)
 - `export_csv`, [19](#)
 - `exportMatrix`, [19](#)
 - `insert`, [20](#)
 - `links`, [21](#)
 - `printMatrix`, [20](#)
 - `site`, [21](#)
 - `walkCrawl`, [20](#)
 - `walkPrint`, [21](#)
- `v2-blog-link-crawler::parseLinks`, [24](#)
 - `handle_starttag`, [24](#)
- `walkCrawl`
 - `tempcawl::Node`, [13](#)
 - `v2-blog-link-crawler::Node`, [20](#)
- `walkPrint`
 - `tempcawl::Node`, [14](#)
 - `v2-blog-link-crawler::Node`, [21](#)