ht://Check user guide

Gabriele Bartolini <gabriele.bartolini@devise.it>

Table of Contents

Introduction 1
- How it works 2
- The information retrieval module 2
- The tables of a ht://Check database 3
- Getting the information stored 4

Installation 4
- System Requirements 4
- Download ht://Check 4
- Decompressing the tarball 4
- Quick Install 5
- The configure script 5
- Specifying the application directory 5
- Specifying a MySQL directory 5
- Setting the path to ht://Check's man page 5
- MySQL user's privileges for ht://Check 5
- MySQL connection settings 6

Getting started 6

The configuration file 8
- General syntax 8
- Attributes 8
- Inclusion and variable expansion 8
- Configuration attributes 8

FAQ 19
- Configuration and compilation 19
- The MySQL database of ht://Check 19
- Configuring the spider (htcheck) 19

Copyright 20

References 20

ht://Check, more than a link checker - User guide.

Abstract

ht://Check is a link checker that retrieves information through the *HTTP* protocol and stores it in a *MySQL database*. It is particularly suited for small Internet domains or Intranet.

It is written in ANSI C++, which makes it portable over POSIX systems and extremely fast.

ht://Check is free software, distributed under the GNU General Public License (GPL).

Introduction

ht://Check's main goal is to help webmasters managing one or more related sites: after a "crawl", ht://Check creates a rich *data source* made up of information based on the retrieved documents. Here follows a short list of the major insights that ht://Check is able to detect:

- *complete source code* for HTML documents retrieved;
- *single documents attributes* such as content-type, size, last modification time, etc.
- information regarding the *retrieval process of a resource* [1]
- information regarding the *structure of a document*, such as the HTML tags they are made up of
- information regarding the *structure of the website* that has been analysed (links between documents create the so-called *inter-documents relationships* between Internet resources); this feature allows users to get further information:
    - *link results*: check whether a link to a URL or a URL fragment (anchor) exists and is not *broken*; retrieves further information such as redirections, e-mail links and bad encoded links (according to RFC1738) [2]
    - *relationships between documents*, in terms of incoming links and outgoing ones (Web structure mining activity)
- *web content accessibility checks*: from version 1.2.3, ht://Check also performs accessibility checks in accordance with the principles of the University of Toronto's Open Accessibility Checks (OAC) project, allowing users to discover site-wide barriers like images without proper alternatives, missing titles, etc.

A skinny report is given by the `htcheck` application. Most of the available information can be analysed through the PHP interface which comes as a separate package.

How it works

ht://Check is essentially a web *spider*, or *robot* or *crawler*. As well as a search engine (like ht://Dig) indexes words from the Internet, ht://Check stores HTML statements such as tags and attributes, links, URL information, and more.

At the moment, ht://Check supports only *HTTP/1.1* (and HTTP/1.0 also): future plans regard enabling the FTP, NNTP, HTTPS and also local files checks.

Everything is stored in a MySQL database, created from scratch by the application itself. You don't need to create it before, just run `htcheck` and every needed table will be automatically built by the program.

For information regarding the connection to the MySQL database, please consult the MySQL connection settings using the option file section.

The *information retrieval* module

ht://Check is made up of two logical "modules", one corcerning the information retrieval, the other one the analysis of the performed crawl.

[1] for example the response when resources fully retrieved, showing the returned *HTTP status codes*
[2] some limitations apply: such as Javascript URLs, which cannot be parsed

The first step, which is the most important also, is completely performed by the `htcheck` program; depending on the values set in the configuration file, htcheck starts retrieving the URL defined in the *`start_url`* configuration attribute; the crawling process is limited in several ways, most of which regard the URL domain (like *`limit_urls_to`* , *`limit_normalized`*, *`exclude_urls`*) or the distance from the starting URL (*`max_hop_count`*), etcetera.

When `htcheck` retrieves the first document, it checks the answer that the server gave back; if the document exists (HTTP 200 *status code* is returned), and the `Content-Type` is `text/html`, `htcheck` starts parsing the document, and retrieves and stores at least all of the HTML tags and attributes that create a link (it can store all of them if you set *`store_only_links`* to false).

`htcheck` can also manage HTTP redirection (created by header "*Location*" sent by the remote HTTP server) and cookies (as defined by http://www.netscape.com/newsref/std/cookie_spec.html).

In a few words that's the main mechanism regarding the information retrieval module, but -believe me- it is not as easy as it seems! But, as far as you are concerned, I think that's enough for now.

The tables of a *ht://Check* database

First of all, you don't need to create a database for ht://Check; indeed `htcheck` will do it for you!

However, ht://Check creates a database which is made up of these tables:

- Schedule
- Url
- Server
- HtmlStatement
- HtmlAttribute
- Link
- htCheck
- Cookies (since version 1.1)
- Accessibility (since version 1.2.3)

The main task of the *Schedule* table is to manage the crawling system: by querying this table, `htcheck` knows which URLs need to be retrieved, or just checked if they exist.

The *Url* table contains info about those URLs that have been retrieved (either successfully or not): here you can find the HTTP status code returned and its reason phrase, its size, the last access time and modification time too, and more.

The *Server* table contains information about the HTTP servers that have been encountered during the crawling process.

The *HtmlStatement* table contains information about the HTML statements found in each URL; every one of

them contains one and only one HTML *tag*, but can also contain one or more HTML *attributes* inside. These ones are stored in the *HtmlAttribute* table.

The *Link* table let us find and locate every link instantiated by HTML statements (or by HTTP redirections too), so we can have a referencing as well as a referenced URL, and know precisely which HTML attribute created this link.

The *Cookies* table is handled since version 1.1 and stores all the cookies that have been retrieved during the crawl and their related information.

The *htCheck* table contains general info such as start and finish time, number of connections, etcetera.

Getting the information stored

Our starting point is that we now have a database full of information, because `htcheck` has already finished to crawl through the web.

The very first way to get reports from a *crawl*, is to run `htcheck` with the *-s* option, which let it produce summaries (see the Getting Started section).

The other way given by ht://Check is to use the PHP interface, which is really simple and easy to use. Since version 2.0.0, the interface is distributed separately from the ht://Check main package.

As the database is now a common MySQL database, you can use whatever you want in order to to retrieve the information stored in it (Perl, C/C++ programs, JSP). You can also get them on Windows systems, just download *MyODBC*. You got lots of choices, as you can see!

Installation

System Requirements

In order to install and run ht://Check you need a GNU/Linux system with:

- *GNU C/C++ compiler* and libstdc++ installed
- *MySQL* 5.1.x, 5.0.x, 4.1.x, 4.0.x, 3.23.x or 3.22.x

However, ht://Check compiles on other POSIX platforms: so please, if you try and successfully install it, please drop me a line with the characteristics of your system.

Download ht://Check

ht://Check can be downloaded from `http://htcheck.sourceforge.net/`.

Decompressing the tarball

Usually you download ht://Check sources in a `tar.gz` file. In order to decompress them with the following command:

`tar xzvf filename.tar.gz`

For `tar.bz2` files, use:

`tar xjvf filename.tar.bz2`

Quick Install

```
configure
make
make install
```

The `configure` script

For more info on the *configure* script, run: [code,bash]

```
configure --help
```

Specifying the application directory

By default, ht://Check is installed into the `/opt/htcheck` directory. And everything is under that directory. Nothing is put out of it. If you want to specify another directory of installation, just use the configuration option —prefix=DIR. For example, if you want to install it into the `/myapps/htcheck` dir, just run configure with this option too:

```
configure [other options] --prefix=/myapps/htcheck
```

Specifying a MySQL directory

ht://Check needs *MySQL* client library support. By default, ht://Check uses `mysql_config` to determine the compiler settings. In case the automatic detection of mysql_config fails (different location on the file system, or different name), please specify it using the `–with-mysql` option:

```
--with-mysql=/opt/local/bin/mysql_config5
```

Setting the path to ht://Check's man page

ht://Check comes with a simple man page, useful for reminding you the options of the application. Let's suppose you installed ht://Check in the `/opt/htcheck` directory, you can easily set the man application to read this page too, by adding in the user or system profile (i.e. `~/.bash_profile` or `/etc/profile`) these line:

```
export MANPATH=$MANPATH:/opt/htcheck/man
```

MySQL user's privileges for ht://Check

In order to run the `htcheck` program, you must connect to the MySQL server as a valid user, with enough permissions. As long as the spider needs to create and drop databases, tables and indexes too, perform insert, update and delete operations you must grant to it these rights (by altering the *user* table's contents of the

mysql database on the MySQL server). So, set to *Y* these fields values:

- Select_priv
- Insert_priv
- Update_priv
- Delete_priv
- Create_priv
- Drop_priv
- Index_priv

However, you are suggested to give a look at the following section.

MySQL connection settings

In order to access a MySQL server, you have 2 choices:

- doing nothing: the access is made by the current user to localhost with no password specified.
- create or use an existing option file for MySQL. See the ref following section.

MySQL connection settings using the option file

We were saying that you can create or use an existing option file for MySQL, where you can specify the host to be accessed, the user, the password, the port and the socket.

By default, ht://Check looks for the `~/.my.cnf` file and if this is not found the global option file for mysql is searched (`/etc/my.cnf`). You can change the prefix (*my*) with the `mysql_conf_file_prefix` configuration option (only for MySQL 3.23, 4.0, 4.1 and 5.0). The group searched is [client] but it can be customised with `mysql_conf_group`.

For example, you can write the `~/.my.cnf` file this way:

```
[client]
host=mysqlserver.mydomain.com
user=htcheck
password=ht12345
```

You can also specify a different `port` or `socket`. You are strongly recommended to change this file permissions to 600.

It goes without saying that in both cases you have to *grant permissions* to the user ht://Check is connecting as. See the previous section and *MySQL documentation* for more info on this subject.

Getting started

In order to perform the first crawl, you just need to edit the configuration file, which resides in the configuration directory with the name *htcheck.conf* (you may use another file as configuration file, but you gotta run htcheck it with the *-c* option).

Just change the *start_url* attribute to whatever you want, for example:

```
start_url:  http://www.foo.com
```

Remember that every URL must start with the service name, that is to say *http://*.

Then set the *limit_urls_to* attribute to $(start_url), in order to scan only the *http://www.foo.com* website.

You may change many other attributes (database name included), but for now, in order to test if it works or not, that's enough.

You can finally enter the bin directory inside the *htcheck* installation directory (by default /opt/htcheck) and run:

```
htcheck -vs
```

However, here are the available options (just run htcheck –help) and you will get this:

```
usage: htcheck  [-isvkhr] [-c configfile] [-D dbname] [--help] [--version]

Options:
        -v      Verbose mode (more 'v's increment verbosity)

        -s      Statistics (broken links, etc...) available

        -i      Initialize the database (drop a previous db)

        -k      Initialize the database (drop tables, keep the db)

        -c configfile
                Configuration file

        -D dbname
                Name of the database

        --help  Display this
        -h      Same as --help

        --version       Display version
        -r      Same as --version
```

Remember that htcheck always check if the database already exists in the MySQL server. If it does not exist, it is created from scratch. On the other hand, if htcheck is launched with the *-i* option, this database is initialized again (this means that a new crawl is performed), else the program just use a previous database, which is useful in order to get some reports like broken links and anchors, content-type summaries (in this case you gotta set the *-s* option).

Since version 1.2.0 it is possible not to drop a database, but keep it alive, and recreate the structure: in technical words, ht://Check tables are dropped and then recreated: this feature was proposed by Patrick Guillot (<pguillot@paanjaru.com>) and enables to use ht://Check within a database that can be used for other purposes as well.

The configuration file

General syntax

ht://Check uses a flexible configuration file. This configuration file is a plain ASCII text file. Each line in the file is either a comment or contains an attribute. Comment lines are blank lines or lines that start with a #.

Attributes

Attributes consist of a variable name and an associated value:

```
<name>:<whitespace><value><newline>
```

The `name` contains any alphanumeric character or underline (_).

The `value` can include any character except newline. It also cannot start with spaces or tabs since those are considered part of the whitespace after the colon. It is important to keep in mind that any trailing spaces or tabs will be included.

It is possible to split the `value` across several lines of the configuration file by ending each line with a backslash (\). The effect on the value is that a space is added where the line split occurs.

If ht://Check needs a particular attribute and it is not in the configuration file, it will use the default value which is defined in htcommon/defaults.cc of the source directory.

Inclusion and variable expansion

A configuration file can include another file, by using a special `name`, include. The `value` is taken as the file name of another configuration file to be read in at this point. If the given file name is not fully qualified, it is taken relative to the directory in which the current configuration file is found.

Variable expansion is permitted in the file name. Multiple include statements, and nested includes are also permitted. Example:

```
include: common.conf
```

Configuration attributes

Here you can find a brief explanation of ht://Check configuration attributes.

They've been grouped in these sections:

- setting the *spider*
- setting the database info
- setting HTTP connections
- setting what to store

- setting what to report
- accessibility checks

Setting the "spider"

`start_url`

This is the list of URLs that will be used to start a dig when there was no existing database. Note that multiple URLs can be given here.

Type: string

Default: `http://htcheck.sourceforge.net/`

Example:

```
start_url:      http://www.somewhere.org/alldata/index.html
```

`limit_urls_to`

This specifies a set of patterns that all URLs have to match against in order for them to be included in the search. Any number of strings can be specified, separated by spaces. If multiple patterns are given, at least one of the patterns has to match the URL. Matching is a case-insensitive string match on the URL to be used. The match will be performed *after* the relative references have been converted to a valid URL. This means that the URL will *always* start with `http://`. Granted, this is not the perfect way of doing this, but it is simple enough and it covers most cases.

Type: string

Example:

```
limit_urls_to:  .sdsu.edu kpbs
```

`limit_normalized`

This specifies a set of patterns that all URLs have to match against in order for them to be included in the search. Unlike the limit_urls_to directive, this is done after the URL is normalized.

Type: string

Default:

Example:

```
limit_normalized: http://www.mydomain.com
```

`exclude_urls`

If a URL contains any of the space separated patterns, it will be rejected. This is used to prevent `htcheck`

from performing infinite loops on poorly designed dynamic pages.

Type: string

Default:

Example:

```
exclude_urls: students.html cgi-bin
```

bad_extensions

This is a list of extensions on URLs which are considered non-parsable. This list is used mainly to supplement the MIME-types that the HTTP server provides with documents. Some HTTP servers do not have a correct list of MIME-types and so can advertise certain documents as text while they are some binary format.

Type: string

Default:

Example:

```
bad_extensions: .foo .bar .bad
```

bad_querystr

This is a list of CGI query strings to be excluded from indexing. This can be used in conjunction with CGI-generated portions of a website to control which pages are indexed.

Type: string

Default:

Example:

```
bad_querystr: forum=private section=topsecret&amp;passwd=required
```

max_hop_count

Instead of limiting the indexing process by URL pattern, it can also be limited by the number of hops or clicks a document is removed from the starting URL. The starting page will have hop count 0.

Type: number

Default: `999999`

Example:

```
max_hop_count: 4
```

max_urls_count

Maximum number of URLs to be parsed. When this number is reached, ht://Check stops parsing URLs and

performs a simple check for existance.

Type: number

Default: `-1`

Example:

```
max_urls_count: 100
```

`check_external`

If set to *true*, htcheck check if external Urls exist or not. An external Url is an Url which doesn't match limit configuration attributes. External URLs aren't parsed.

Type: boolean

Default: `true`

Example:

```
check_external: false
```

Setting the database info

`db_name`

Name of the MySQL database to be created or read.

Type: string

Default: `htcheck` (or as defined by the `–with-db-name` configure option)

Example:

```
db_name: test
```

`db_name_prepend`

String to be prepended to the MySQL database name specified. This allows to set a common string to identify all the database name used by ht://Check and to grant database privileges by using this string value. You can change the default value also by using the configure option: —with-db-name-prepend (default empty).

Type: string

Default: (or as defined by the `–with-db-name-prepend` configure option)

Example:

```
db_name_prepend: htcheck_
```

mysql_conf_file_prefix

Only for MySQL < 5.1. Prefix for the MySQL configuration file to be searched. Default is *my* and the file that is searched is usually `~/.my.cnf` (suggested). If it is not found the `/etc/.my.cnf` file is searched. For its syntax, look at the *Option File* contents inside the MySQL documentation.

Type: string

Default: `my`

Example:

```
mysql_conf_file_prefix: htcheck
```

mysql_conf_group

Group to be searched inside the .my.cnf file of MySQL for getting the settings for the connection to the server. In other words, it's the section marked with [<group>] inside the MySQL option file (default is [client]).

Type: string

Default: `client`

Example:

```
mysql_conf_group: htcheck
```

optimize_db

Optimize the database tables at the end of the crawl. Disable it if the database server doesn't support it.

Type: boolean

Default: `false`

Example:

```
optimize_db: true
```

sql_big_table_option

Enable or disable this option that is useful when performing huge queries. Otherwise, sometimes when it's not set, the MySQL db server may return a *table is full* error.

Type: boolean

Default: `true`

Example:

```
sql_big_table_option: false
```

`url_index_length`

This number specifies the length of the index of the Url field in the Schedule and Url tables of the database. You can set different values depending on the average length of the URLs that htcheck can find in your sites. If you don't want to set any limitation, just put a *-1* value. This now allows the user to control the length of the index for the Url field in the Schedule and Url tables. This attribute may affect the performance of the crawls, as long as the length of a index can either slow down or speed up the spidering process.

Type: number

Default: `64`

Example:

```
url_index_length: -1
```

Setting HTTP connections

`user_agent`

This allows customization of the user_agent: field sent when the digger requests a file from a server.

Type: string

Default: `ht://Check`

Example:

```
user_agent: htcheck-crawler
```

`persistent_connections`

If set to true, when servers make it possible, htdig can take advantage of persistent connections, as defined by HTTP/1.1 (*RFC2616*). This permits to reduce the number of open/close operations of connections, when retrieving a document with HTTP.

Type: boolean

Default: `true`

Example:

```
persistent_connections: false
```

`head_before_get`

This option works only if we take advantage of persistent connections (see persistent_connections attribute). If set to true an HTTP/1.1 *HEAD* call is made in order to retrieve header information about a document. If the status code and the content-type returned let the document be parsable, then a following *GET* call is made.

Type: boolean

Default: `true`

Example:

```
head_before_get: false
```

`timeout`

Specifies the time the digger will wait to complete a network read. This is just a safeguard against unforeseen things like the all too common transformation from a network to a notwork.

The timeout is specified in seconds.

Type: number

Default: `30`

Example:

```
timeout: 42
```

`authorization`

This tells htcheck to send the supplied *username*:*password* with each HTTP request. The credentials will be encoded using the "Basic" authentication scheme. There must be a colon (:) between the username and password.

Type: string

Default:

Example:

```
authorization: myusername:mypassword
```

`max_retries`

This option set the maximum number of retries when retrieving a document fails (mainly for reasons of connection).

Type: number

Default: `3`

Example:

```
max_retries: 6
```

`tcp_max_retries`

This option set the maximum number of attempts when a connection raises a xref:timeout. After all these retries, the connection attempt results *timed out*.

Type: number

Default: `1`

Example:

```
tcp_max_retries: 6
```

`tcp_wait_time`

This attribute sets the wait time after a connection fails and the xref:timeout is raised.

Type: number

Default: `5`

Example:

```
tcp_wait_time: 10
```

`http_proxy`

When this attribute is set, all HTTP document retrievals will be done using the HTTP-PROXY protocol. The URL specified in this attribute points to the host and port where the proxy server resides.

The use of a proxy server greatly improves performance of the indexing process.

Type: string

Default:

Example:

```
http_proxy: http://proxy.bigbucks.com:3128
```

`http_proxy_exclude`

When this is set, URLs matching this will not use the proxy. This is useful when you have a mixture of sites near to the digging server and far away.

Type: string

Default:

Example:

```
http_proxy_exclude: http://intranet.foo.com/
```

`http_proxy_authorization`

This tells htcheck to send the supplied *username*:*password* with each HTTP request, when using a proxy with authorization requested. The credentials will be encoded using the \"Basic\" authentication scheme. There *must* be a colon (:) between the username and password.

Type: string

Default:

Example:

```
http_proxy_authorization: myusername:mypassword
```

accept_language

This attribute allows to restrict the set of natural languages that are preferred as a response to an HTTP request performed by the digger. This can be done by putting one or more language tags (as defined by RFC 1766) in the preferred order, separated by spaces. By doing this, when the server performs a content negotiation based on the *accept-language* given by the HTTP user agent, a different content can be shown depending on the value of this attribute. If set empty, no language will be sent and the server default will be returned.

Type: string

Default:

Example:

```
accept_language:        en-us en it
```

remove_default_doc

Set this to the default documents in a directory used by the servers you are indexing. These document names will be stripped off of URLs when they are normalized, if one of these names appears after the final slash, to translate URLs like http://foo.com/index.html into http://foo.com/ Note that you can disable stripping of these names during normalization by setting the list to an empty string. The list should only contain names that all servers you index recognize as default documents for directory URLs, as defined by the DirectoryIndex setting in Apache's srm.conf, for example.

Type: string list

Default:

Example:

```
remove_default_doc: default.html default.htm index.html index.htm
```

disable_cookies

If set to *true*, htcheck will disable the HTTP cookies management.

Type: boolean

Default: `false`

Example:

```
disable_cookies: true
```

cookies_input_file

Set the input file to be used when importing cookies for the crawl; cookies must be specified according to Netscape's format. For more information, give a look at the example cookies file distributed with ht://Check. By default, no input file is read.

Type: string

Default:

Example:

```
cookies_input_file: /tmp/cookies.txt
```

url_reserved_chars

This string allows to customise the set of characters that can be considered as reserverd in a URL, avoiding their coding under the `RFC1738` standard. This string is used when checking whether a URL is well-encoded or not, issuing a BadEncoded state for the link which created it. The default value is slightly different from what the RFC says, giving more flexibility to the spider (it is suggested not to change it unless you are extremely sure of what you are doing).

Type: string

Default: `;/?:@&=$,._%-#x~+`

Example:

```
url_reserved_chars: \\;/?:@&=+\$,._%-#x~
```

Setting what to store

max_doc_size

This is the upper limit to the amount of data retrieved for documents. This is mainly used to prevent unreasonable memory consumption since each document will be read into memory by htcheck.

Type: number

Default: `100000`

Example:

```
max_doc_size: 5000000
```

store_only_links

If set to `false`, htcheck will store in the DB *every* tag he finds in every document it crawls. If set to `true`, htcheck stores only those Html attributes and statements that produce a link or set an anchor (identified by the pair tag: A, attribute: name).

Type: boolean

Default: `false`

Example:

```
store_only_links: true
```

`store_url_contents`

This attribute allows to store the contents of the parsed URLs. It is very *useful*, but can also be *dangerous*. You must know what you are doing, and if you enable this, your performances may slow down and your disk storage requirements can get extremely high. It is recommended to use this only for small crawls.

Type: boolean

Default: `false`

Example:

```
store_url_contents: true
```

`available_charsets`

This attribute specifies the set of possible *charsets* that htcheck recognises and stores into the database; other charsets will be marked as *other*.

Type: string list

Default:

```
windows-1250 iso-8859-1 iso-8859-10 iso-8859-13 iso-8859-14
iso-8859-15 iso-8859-2 iso-8859-3 iso-8859-4 iso-8859-5 iso-8859-6 iso-8859-7
iso-8859-8 iso-8859-9 koi8-r koi8-u utf-8 windows-1251 windows-1252 windows-1253
windows-1254 windows-1255 windows-1256 windows-1257 windows-1258 windows-874
```

Example:

```
available_charsets: iso-8859-1
```

Setting what to report

`summary_anchor_not_found`

Enable or disable the show of the summary of the HTML anchors that have not been found.

Type: boolean

Default: `true`

Example:

```
summary_anchor_not_found: false
```

Accessibility checks

`accessibility_checks`

Enable or disable the recognition of accessibility problems, using some of the checks proposed by the Open Accessibility Checks project by the Adaptive TechnologyResource Center at the University Of Toronto. From version 1.2.3, ht://Checks internally stores this kind of information in the *AccessibilityChecks* table using the code number specified in OAC (http://oac.atrc.utoronto.ca).

Type: boolean

Default: `true`

Example:

```
accessibility_checks: false
```

FAQ

Configuration and compilation

I'm compiling with gcc 3.2 and getting several warnings/errors regarding ostream

You should use the following command to configure ht://Check so it can be built with gcc 3.2:

```
CXXFLAGS=-Wno-deprecated CPPFLAGS=-Wno-deprecated ./configure
```

However, from version 1.2.2, sources have been updated in order to automatically detect the correct standard C++ library; backward compatibility C++ headers (such as fstream.h) are not used anymore in the main code, although pre-processing checks are performed for older libraries.

The MySQL database of ht://Check

What tables have to be created? What about the fields? and their format?

ht://Check does everything for you. It creates the database structure itself, so you don't need to create it before. You just need to grant the spider enough permissions in order to do that.

Configuring the *spider* (`htcheck`)

How do I change the URLs to check without going through the PHP interface?

No. There's no way to configure the spider through PHP for now. You just have to edit the configuration file (usually *htcheck.conf*).

If I run htcheck at the commandline, I don't see a way to change the URLs to check. I'm guessing that the Server table in the htcheck database is what I want to modify, right?

No .. you don't need to modify the MySQL database at all. Indeed it's for getting the results only. Every database is directly created by the application (from scratch). You must edit the parameters in the htcheck.conf file. You have to set one or more starting URL with the *start_url* attribute. Then you can limit the search to a set of URLs by setting the *limit_urls_to*, *limit_normalized* and *exclude_urls* options. These are the most used and important, though you can use the *bad_extension*, *max_hop_count*, *bad_query_string*. But in most of cases you only have to set the *limit_urls_to* parameter. For instance:

```
start_url: http://www.foo.com
limit_urls_to: $(start_url)
```

The *limit_normalized* parameter checks for every URL after it has been normalised (transformed into this format: `service://host:port/path`).

Copyright

References

[htdig] The ht://Dig Group. *ht://Dig Search Engine*. http://www.htdig.org/

[mysql] Sun Microsystems, Inc. *MySQL*. http://www.mysql.com/

[RFC1738] The Internet Society. *RFC 1738 - Uniform Resource Locators (URL)*.
http://tools.ietf.org/html/rfc1738

[RFC1766] The Internet Society. *RFC 1766 - Tags for the Identification of Languages*.
http://tools.ietf.org/html/rfc1766

[RFC2616] The Internet Society. *RFC 2616 - Hypertext Transfer Protocol 1.1 — HTTP/1.1*.
http://tools.ietf.org/html/rfc2616

Index

I
inter-documents relationships, 2

S
structure, 2

W

web content accessibility, 2
Web mining
  structure, 2
Web structure mining, 2