Package 'scrapeR'

February 20, 2015

Type Package

Title Tools for Scraping Data from HTML and XML Documents

Version 0.1.6

Date 2009-10-12

Author Ryan M. Acton <rmacton@gmail.com>

Maintainer Ryan M. Acton <rmacton@gmail.com>

Depends XML,RCurl

Description Tools for Scraping Data from Web-Based Documents

License GPL (>= 2)

URL <http://www.ryanacton.com>

LazyLoad yes

Repository CRAN

Date/Publication 2010-02-03 10:09:28

NeedsCompilation no

R topics documented:

| scrapeR-package | *Tools for Scraping Data from Web-Based Documents* |

Description

Tools for Scraping Data from Web-Based Documents

Details

1

Package:	scrapeR
Type:	Package
Version:	0.1.6
Date:	2009-11-10
License:	GPL (>= 2)
LazyLoad:	yes

Author(s)

Ryan M. Acton <rmacton@gmail.com> <http://www.ryanacton.com>

References

Duncan Temple Lang. (2009). XML: Tools for parsing and generating XML within R and S-Plus. <http://CRAN.R-project.org/package=XML>.

Duncan Temple Lang. (2009). RCurl: General network (HTTP/FTP/...) client interface for R. <http://CRAN.R-project.org/package=RCurl>.

---

scrape	*A Tool For Scraping and Parsing HTML and XML Documents From the Web*

Description

This function assists the user with retrieving HTML and XML files, parsing their contents, and diagnosing potential errors that may have occurred along the way.

Usage

```
scrape(url=NULL,object=NULL,file=NULL,chunkSize=50,maxSleep=5,
userAgent=unlist(options("HTTPUserAgent")),follow=FALSE,
headers=TRUE,parse=TRUE,isXML=FALSE,.encoding=integer(),
verbose=FALSE)
```

Arguments

url	a vector of URLs, each as a character string. Either the url, object, or the file parameter must be provided.
object	character; the name of an R object that contains the raw source code of an HTML or XML. This parameter is likely useful when a previous call to scrape simply gathered document source code, followed redirects, and/or returned the headers, thus allowing the user to inspect the output first for potential problems before deciding to parse it into an R-friendly tree-like structure. Either the object, url, or the file parameter must be provided.

file	a vector of paths to local files, as a character string. Either the `file`, `url`, or the `object` parameter must be provided.
chunkSize	integer; if a vector of `urls` is supplied whose size is greater than the value of `chunkSize`, the `urls` will be split into chunks of size `chunkSize`. By splitting the `urls` into chunks, the number of simultaneous HTTP requests is reduced, thus placing less burden on the server. The default value of `chunkSize` is `50`. It is not recommended that one specifies a value of `chunkSize` larger than `100`.
maxSleep	integer; if the vector of `urls` is larger than the value of `chunkSize`, the function will "sleep" for `ceiling(runif(1,min=0,max=maxSleep))` seconds between chunks. It is often helpful to use a sleep parameter when making repeated HTTP requests so as to not overwhelm the servers with gapless sequential requests. The default value for this parameter is 5.
userAgent	the User-Agent HTTP header that is supplied with any HTTP requests made by this function. This header is used to identify your HTTP calls to the host server. It is strongly recommended that one uses an informative User-Agent header, perhaps with a link to one's email or web address. This information may prove helpful to system administrators when they are unsure of the legitimacy your HTTP requests, as it provides them a way of contacting you. See the URL reference for "User-Agent" headers below for more information. By default, the User-Agent header is assigned the value given by `unlist(options("HTTPUserAgent"))`, but the user is encouraged to construct a customized version.
follow	logical; should these HTTP requests follow URL redirects if they are encountered? Here, redirection will only occur with HTTP requests for which the status code is of the 3xx type (see the reference to HTTP status codes below). This parameter is only meaningful if the `url` parameter is supplied. The default value for this parameter is `FALSE`.
headers	logical; should these HTTP requests retrieve the resulting HTTP headers? This parameter is only meaningful if the `url` parameter is supplied. The default value for this parameter is `FALSE`.
parse	logical, should the `url` or `file` vectors be parsed into R-friendly tree-like structures? See [xmlTreeParse](#) for more information about this feature and how the object is returned. If parse==TRUE, this tree-like structure is easily navigable using the XPath language (see the corresponding url reference provided below and the help page for [xpathSApply](#)). The default value for this parameter is `TRUE`.
isXML	logical; do the `url` or `file` vectors point to well-formed XML files? See [xmlTreeParse](#) for the differences between parsing XML and HTML documents. The default value for this parameter is `FALSE`.
.encoding	integer or a string; identifies the encoding of the retrieved content. See [getURL](#) for more information.
verbose	logical; shall the function print extra information to the console? The default value for this parameter is `FALSE`.

Value

If `url` or `file` is supplied, then either the raw source code of the `urls` (`files`) is returned as a list of (potentially long) character vectors (when parse==FALSE), or a list of R-friendly tree-like

structures of the documents is returned (when parse==TRUE). If object is supplied, then either the raw source code contained within the object is returned as a list object of (potentially long) character strings (when parse==FALSE), or a list object of R-friendly tree-like structures for the documents is returned (when parse==TRUE). If url or object are supplied, the resulting object may have the following attributes:

redirect.URL   the destination URLs that resulted from a series of redirects, if they occurred; else NA. This is only returned if follow==TRUE.

headers   the HTTP headers resulting from these HTTP requests. These are only returned if headers==TRUE.

Author(s)

Ryan M. Acton <racton@uci.edu> <http://www.ryanacton.com>

References

Duncan Temple Lang. (2009). XML: Tools for parsing and generating XML within R and S-Plus. <http://CRAN.R-project.org/package=XML>.

Duncan Temple Lang. (2009). RCurl: General network (HTTP/FTP/...) client interface for R. <http://CRAN.R-project.org/package=RCurl>.

Information about HTTP status codes: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>.

Information about User-Agent headers: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.43>.

Information about the XPath language: <http://www.w3schools.com/XPath/default.asp>.

Examples

```
## Not run:
## Example 1.  Getting all of the package names available for download
##             from CRAN (http://cran.r-project.org/web/packages/)

# First, pull in the page's source code, check for (and follow) a page redirection,
# and retrieve the headers before deciding to parse the code.
pageSource<-scrape(url="http://cran.r-project.org/web/packages/",headers=TRUE,
parse=FALSE)

# Second, inspect the headers to ensure a status code of 200, which means the page
# was served properly.  If okay, then parse the object into an XML tree and retrieve
# all of the package names.
if(attributes(pageSource)$headers["statusCode"]==200) {
page<-scrape(object="pageSource")
xpathSApply(page,"//table//td/a",xmlValue)
} else {
cat("There was an error with the page. \n")
}

## End(Not run)
```

```
## Example 2. Parsing a local XML file, then pulling out information of interest

# First, locate and parse the demo recipe file supplied with this package
fileToLoad<-system.file("recipe.xml",package="scrapeR")
mmmCookies<-scrape(file=fileToLoad,isXML=TRUE)

# Next, retrieve the names of the dry ingredients that I'll need to buy
xpathSApply(mmmCookies[[1]],"//recipe/ingredient[@type='dry']/item",xmlValue)

# Next, remind myself how much flour is needed
paste(xpathSApply(mmmCookies[[1]],"//item[.='flour']/preceding-sibling::amount",
xmlValue),xpathSApply(mmmCookies[[1]],"//item[.='flour']/
preceding-sibling::unit",xmlValue))

# Finally, remind myself who the author of this recipe is
xpathSApply(mmmCookies[[1]],"//recipe",xmlGetAttr,"from")
```

Index