ServerFiles Documentation

Release 0.2

Biolab

Sep 27, 2017

Contents:

1 ServerFiles **1**
- 1.1 Server with files . . . 1
- 1.2 Info files . . . 2
- 1.3 Server query optimization . . . 2
- 1.4 Remote files . . . 2
- 1.5 Local files . . . 3

2 Indices and tables **5**

Python Module Index **7**

CHAPTER 1

ServerFiles

Access and store files when needed.

Server with files

Server provides files through HTTP. Any HTTP server that can serve static files can work, including Apache, Nginx and Python's HTTP server.

Files can be organized in subfolders. Each file can have a corresponding info file (with .info extension).

A test server could be made by just creating a new empty folder and creating a subfolder "additional-data" there with the following files:

```
additional-data/a-very-big-file.txt
additional-data/a-very-big-file.txt.info
```

Our .info file should contain the following:

```
{"tags": [ "huge file", "example" ], "datetime": "2016-10-10 11:39:07"}
```

Then we can start a test server with:

```
python -m http.server
```

To access the server and download the file we could use:

```
>>> import serverfiles
>>> sf = serverfiles.ServerFiles(server="http://localhost:8000/")
>>> sf.listfiles()
[('additional-data', 'a-very-big-file.txt')]
>>> lf = serverfiles.LocalFiles("sftest", serverfiles=sf)
>>> lf.download('additional-data', 'a-very-big-file.txt')
```

Info files

Info files, which have an additional .info extension, must be SON dictionaries. Keys that are read by this module are:

- datetime ("%Y-%m-%d %H:%M:%S"),
- compression (if set, the file is uncompressed automatically, can be one of .bz2, .gz, .tar.gz, .tar.bz2),
- and tags (a list of strings).

Server query optimization

A server can contain a __INFO__ file in its root folder. This file is a JSON list, whose elements are lists of [list-of-path, info dictionary]. If such file exists its contents will be used instead of server queries for file listing and info lookup, which is critical for high latency connections. Such file can be prepared as:

```
>>> sf = ServerFiles(server="yourserver")
>>> json.dump(list(sf.allinfo().items()), open("__INFO__", "wt"))
```

If your server already has an __INFO__ file, the above code will just get its contents.

Remote files

class serverfiles.**ServerFiles**(*server*, *username=None*, *password=None*)

A class for listing or downloading files from the server.

allinfo(**path*, ***kwargs*)

Return all info files in a dictionary, where keys are paths.

download(**path*, ***kwargs*)

Download a file and name it with target name. Callback is called once for each downloaded percentage.

info(**path*)

Return a dictionary containing repository file info.

listfiles(**args*, ***kwargs*)

Return a list of files on the server. Do not list .info files.

password = None

Password for authenticated HTTP queried.

search(*sstrings*, ***kwargs*)

Search for files on the repository where all substrings in a list are contained in at least one choosen field (tag, title, name). Return a list of tuples: first tuple element is the file's domain, second its name. As for now the search is performed locally, therefore information on files in repository is transfered on first call of this function.

server = None

Server URL.

username = None

Username for authenticated HTTP queried.

Local files

class serverfiles.**LocalFiles**(*path*, *serverfiles=None*)

Manage local files.

allinfo(**path*)

Return all local info files in a dictionary, where keys are paths.

download(**path*, ***kwargs*)

Download file from the repository. Callback can be a function without arguments and will be called once for each downloaded percent of file: 100 times for the whole file. If extract is True, files marked as compressed will be uncompressed after download.

info(**path*)

Return .info file for a file in a local repository.

listfiles(**path*)

List files (or folders) in local repository that have corresponding .info files. Do not list .info files.

localpath(**args*)

Return the local location for a file.

localpath_download(**path*, ***kwargs*)

Return local path for the given domain and file. If file does not exist, download it. Additional arguments are passed to the *download* function.

needs_update(**path*)

Return True if a file does not exist in the local repository, if there is a newer version on the server or if either version can not be determined.

remove(**path*, ***kwargs*)

Remove a file of a path from local repository.

search(*sstrings*, ***kwargs*)

Search for files in the local repository where all substrings in a list are contained in at least one chosen field (tag, title, name). Return a list of tuples: first tuple element is the domain of the file, second its name.

serverfiles = None

A ServerFiles instance.

serverfiles_dir = None

A folder downloaded files are stored in.

update(**path*, ***kwargs*)

Download the corresponding file from the server if server copy was updated.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

serverfiles, 1

Index

A

allinfo() (serverfiles.LocalFiles method), 3
allinfo() (serverfiles.ServerFiles method), 2

D

download() (serverfiles.LocalFiles method), 3
download() (serverfiles.ServerFiles method), 2

I

info() (serverfiles.LocalFiles method), 3
info() (serverfiles.ServerFiles method), 2

L

listfiles() (serverfiles.LocalFiles method), 3
listfiles() (serverfiles.ServerFiles method), 2
LocalFiles (class in serverfiles), 3
localpath() (serverfiles.LocalFiles method), 3
localpath_download() (serverfiles.LocalFiles method), 3

N

needs_update() (serverfiles.LocalFiles method), 3

P

password (serverfiles.ServerFiles attribute), 2

R

remove() (serverfiles.LocalFiles method), 3

S

search() (serverfiles.LocalFiles method), 3
search() (serverfiles.ServerFiles method), 2
server (serverfiles.ServerFiles attribute), 2
ServerFiles (class in serverfiles), 2
serverfiles (module), 1
serverfiles (serverfiles.LocalFiles attribute), 3
serverfiles_dir (serverfiles.LocalFiles attribute), 3

U

update() (serverfiles.LocalFiles method), 3
username (serverfiles.ServerFiles attribute), 2