
instascrape

Release 0.0.7

Jan 20, 2021

Contents:

1	instascrape.scrapers package	1
1.1	Submodules	1
1.2	instascrape.scrapers.hashtag module	1
1.3	instascrape.scrapers.post module	2
1.4	instascrape.scrapers.profile module	4
1.5	instascrape.scrapers.reel module	5
1.6	instascrape.scrapers.location module	6
1.7	instascrape.scrapers.igtv module	7
1.8	instascrape.scrapers.scrape_tools module	9
1.9	Module contents	10
2	instascrape.exceptions package	11
2.1	Submodules	12
2.2	instascrape.exceptions.exceptions module	12
2.3	Module contents	13
3	Indices and tables	15
	Python Module Index	17
	Index	19

1.1 Submodules

1.2 instascrape.scrapers.hashtag module

1.2.1 Hashtag

Scrape data from a Hashtag page

class instascrape.scrapers.hashtag.**Hashtag** (*source: Union[str, bs4.BeautifulSoup, Dict[str, Any]]*)

Bases: instascrape.core._static_scraper._StaticHtmlScraper

Scraper for an Instagram hashtag page

get_recent_posts (*amt: int = 71*) → List[instascrape.scrapers.post.Post]

Return a list of recent posts to the hashtag

Parameters **amt** (*int*) – Amount of recent posts to return

Returns **posts** – List containing the recent 12 posts and their available data

Return type List[*Post*]

scrape (*mapping=None, keys: Optional[List[str]] = None, exclude: Optional[List[str]] = None, headers={'user-agent': 'Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Mobile Safari/537.36 Edg/87.0.664.57'}, inplace=True, session=None, webdriver=None*) → None

Scrape data from the source

Parameters

- **mapping** (*Dict[str, deque]*) – Dictionary of parsing queue's that tell the JSON engine how to process the JSON data
- **keys** (*List[str]*) – List of strings that correspond to desired attributes for scraping

- **exclude** (*List[str]*) – List of strings that correspond to which attributes to exclude from being scraped
- **headers** (*Dict[str, str]*) – Dictionary of request headers to be passed on the GET request
- **inplace** (*bool*) – Determines if data modified inplace or return a new object with the scraped data
- **session** (*requests.Session*) – Session for making the GET request
- **webdriver** (*selenium.webdriver.chrome.webdriver.WebDriver*) – Webdriver for scraping the page, overrides any default or passed session

Returns Optionally returns a scraped instance instead of modifying inplace if inplace arg is True

Return type return_instance

session = <requests.sessions.Session object>

to_csv (*fp: str*) → None

Write scraped data to .csv at the given filepath

Parameters **fp** (*str*) – Filepath to write data to

to_dict (*metadata: bool = False*) → Dict[str, Any]

Return a dictionary containing all of the data that has been scraped

Parameters **metadata** (*bool*) – Boolean value that determines if metadata specified in self._METADATA_KEYS will be included in the dictionary.

Returns **data_dict** – Dictionary containing the scraped data

Return type Dict[str, Any]

to_json (*fp: str*) → None

Write scraped data to .json file at the given filepath

Parameters **fp** (*str*) – Filepath to write data to

1.3 instascape.scrapers.post module

1.3.1 Post

Scrape data from a Post page

class instascape.scrapers.post.**Post** (*source: Union[str, bs4.BeautifulSoup, Dict[str, Any]]*)
Bases: instascape.core._static_scraper._StaticHtmlScraper

Scraper for an Instagram post page

SUPPORTED_DOWNLOAD_EXTENSIONS = ['.mp3', '.mp4', '.png', '.jpg']

download (*fp: str*) → None

Download an image or video from a post to your local machine at the given filepath

Parameters **fp** (*str*) – Filepath to download the image to

embed () → str

Return embeddable HTML str for this post

Returns **html_template** – HTML string with embed markup for this Post

Return type str

get_recent_comments () → List[instascape.scrapers.comment.Comment]

Returns a list of Comment objects that contain data regarding some of the posts comments

Returns **comments_arr** – List of Comment objects

Return type List[Comment]

scrape (*mapping=None, keys: Optional[List[str]] = None, exclude: Optional[List[str]] = None, headers={'user-agent': 'Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Mobile Safari/537.36 Edg/87.0.664.57'}, inplace=True, session=None, webdriver=None*) → None

Scrape data from the source

Parameters

- **mapping** (*Dict[str, deque]*) – Dictionary of parsing queue's that tell the JSON engine how to process the JSON data
- **keys** (*List[str]*) – List of strings that correspond to desired attributes for scraping
- **exclude** (*List[str]*) – List of strings that correspond to which attributes to exclude from being scraped
- **headers** (*Dict[str, str]*) – Dictionary of request headers to be passed on the GET request
- **inplace** (*bool*) – Determines if data modified inplace or return a new object with the scraped data
- **session** (*requests.Session*) – Session for making the GET request
- **webdriver** (*selenium.webdriver.chrome.webdriver.WebDriver*) – Webdriver for scraping the page, overrides any default or passed session

Returns Optionally returns a scraped instance instead of modifying inplace if inplace arg is True

Return type return_instance

session = <requests.sessions.Session object>

to_csv (*fp: str*) → None

Write scraped data to .csv at the given filepath

Parameters **fp** (*str*) – Filepath to write data to

to_dict (*metadata: bool = False*) → Dict[str, Any]

Return a dictionary containing all of the data that has been scraped

Parameters **metadata** (*bool*) – Boolean value that determines if metadata specified in self._METADATA_KEYS will be included in the dictionary.

Returns **data_dict** – Dictionary containing the scraped data

Return type Dict[str, Any]

to_json (*fp: str*) → None

Write scraped data to .json file at the given filepath

Parameters **fp** (*str*) – Filepath to write data to

1.4 instascape.scrapers.profile module

class instascape.scrapers.profile.**Profile** (*source: Union[str, bs4.BeautifulSoup, Dict[str, Any]]*)

Bases: instascape.core._static_scraper._StaticHtmlScraper

Scraper for an Instagram profile page

get_posts (*webdriver, amount=None, login_first=False, login_pause=60, max_failed_scroll=300, scrape=False, scrape_pause=5*)

Return Post objects from profile scraped using a webdriver (not included)

Parameters

- **webdriver** (*selenium.webdriver.chrome.webdriver.WebDriver*) – Selenium webdriver for rendering JavaScript and loading dynamic content
- **amount** (*int*) – Amount of posts to return, default is all of them
- **login_first** (*bool*) – Start on login page to allow user to manually login to Instagram
- **login_pause** (*int*) – Length of time in seconds to pause before starting scrape
- **max_failed_scroll** (*int*) – Maximum amount of scroll attempts before stopping if scroll is stuck
- **scrape** (*bool*) – Scrape posts with the webdriver prior to returning
- **scrape_pause** (*int*) – Time in seconds between each scrape

Returns **posts** – Post objects gathered from the profile page

Return type List[[Post](#)]

get_recent_posts (*amt: int = 12*) → List[instascape.scrapers.post.Post]

Return a list of the profiles recent posts. Max available for return is 12.

Parameters **amt** (*int*) – Amount of recent posts to return

Returns **posts** – List containing the recent 12 posts and their available data

Return type List[[Post](#)]

scrape (*mapping=None, keys: Optional[List[str]] = None, exclude: Optional[List[str]] = None, headers={'user-agent': 'Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Mobile Safari/537.36 Edg/87.0.664.57'}, inplace=True, session=None, webdriver=None*) → None

Scrape data from the source

Parameters

- **mapping** (*Dict[str, deque]*) – Dictionary of parsing queue's that tell the JSON engine how to process the JSON data
- **keys** (*List[str]*) – List of strings that correspond to desired attributes for scraping
- **exclude** (*List[str]*) – List of strings that correspond to which attributes to exclude from being scraped
- **headers** (*Dict[str, str]*) – Dictionary of request headers to be passed on the GET request
- **inplace** (*bool*) – Determines if data modified inplace or return a new object with the scraped data
- **session** (*requests.Session*) – Session for making the GET request

- **webdriver** (*selenium.webdriver.chrome.webdriver.WebDriver*) – Webdriver for scraping the page, overrides any default or passed session

Returns Optionally returns a scraped instance instead of modifying inplace if inplace arg is True

Return type return_instance

session = <requests.sessions.Session object>

to_csv (*fp: str*) → None

Write scraped data to .csv at the given filepath

Parameters **fp** (*str*) – Filepath to write data to

to_dict (*metadata: bool = False*) → Dict[str, Any]

Return a dictionary containing all of the data that has been scraped

Parameters **metadata** (*bool*) – Boolean value that determines if metadata specified in self._METADATA_KEYS will be included in the dictionary.

Returns **data_dict** – Dictionary containing the scraped data

Return type Dict[str, Any]

to_json (*fp: str*) → None

Write scraped data to .json file at the given filepath

Parameters **fp** (*str*) – Filepath to write data to

1.5 instascape.scrapers.reel module

class instascape.scrapers.reel.**Reel** (*source: Union[str, bs4.BeautifulSoup, Dict[str, Any]]*)

Bases: *instascape.scrapers.post.Post*

Scraper for an Instagram reel

SUPPORTED_DOWNLOAD_EXTENSIONS = ['.mp3', '.mp4', '.png', '.jpg']

download (*fp: str*) → None

Download an image or video from a post to your local machine at the given filepath

Parameters **fp** (*str*) – Filepath to download the image to

embed () → str

Return embeddable HTML str for this post

Returns **html_template** – HTML string with embed markup for this Post

Return type str

get_recent_comments () → List[instascape.scrapers.comment.Comment]

Returns a list of Comment objects that contain data regarding some of the posts comments

Returns **comments_arr** – List of Comment objects

Return type List[Comment]

scrape (*mapping=None, keys: Optional[List[str]] = None, exclude: Optional[List[str]] = None, headers={'user-agent': 'Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Mobile Safari/537.36 Edg/87.0.664.57'}, inplace=True, session=None, webdriver=None*) → None

Scrape data from the source

Parameters

- **mapping** (*Dict[str, deque]*) – Dictionary of parsing queue’s that tell the JSON engine how to process the JSON data
- **keys** (*List[str]*) – List of strings that correspond to desired attributes for scraping
- **exclude** (*List[str]*) – List of strings that correspond to which attributes to exclude from being scraped
- **headers** (*Dict[str, str]*) – Dictionary of request headers to be passed on the GET request
- **inplace** (*bool*) – Determines if data modified inplace or return a new object with the scraped data
- **session** (*requests.Session*) – Session for making the GET request
- **webdriver** (*selenium.webdriver.chrome.webdriver.WebDriver*) – Webdriver for scraping the page, overrides any default or passed session

Returns Optionally returns a scraped instance instead of modifying inplace if inplace arg is True

Return type return_instance

session = *<requests.sessions.Session object>*

to_csv (*fp: str*) → None

Write scraped data to .csv at the given filepath

Parameters **fp** (*str*) – Filepath to write data to

to_dict (*metadata: bool = False*) → *Dict[str, Any]*

Return a dictionary containing all of the data that has been scraped

Parameters **metadata** (*bool*) – Boolean value that determines if metadata specified in self._METADATA_KEYS will be included in the dictionary.

Returns **data_dict** – Dictionary containing the scraped data

Return type *Dict[str, Any]*

to_json (*fp: str*) → None

Write scraped data to .json file at the given filepath

Parameters **fp** (*str*) – Filepath to write data to

1.6 instascrape.scrapers.location module

class instascrape.scrapers.location.**Location** (*source: Union[str, bs4.BeautifulSoup, Dict[str, Any]]*)

Bases: instascrape.core._static_scraper._StaticHtmlScraper

Scraper for an Instagram profile page

get_recent_posts (*amt: int = 24*) → *List[instascrape.scrapers.post.Post]*

Return a list of recent posts to the location

Parameters **amt** (*int*) – Amount of recent posts to return

Returns **posts** – List containing the recent 24 posts and their available data

Return type *List[Post]*

scrape (*mapping=None, keys: Optional[List[str]] = None, exclude: Optional[List[str]] = None, headers={'user-agent': 'Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Mobile Safari/537.36 Edg/87.0.664.57'}, inplace=True, session=None, webdriver=None*) → None
 Scrape data from the source

Parameters

- **mapping** (*Dict[str, deque]*) – Dictionary of parsing queue's that tell the JSON engine how to process the JSON data
- **keys** (*List[str]*) – List of strings that correspond to desired attributes for scraping
- **exclude** (*List[str]*) – List of strings that correspond to which attributes to exclude from being scraped
- **headers** (*Dict[str, str]*) – Dictionary of request headers to be passed on the GET request
- **inplace** (*bool*) – Determines if data modified inplace or return a new object with the scraped data
- **session** (*requests.Session*) – Session for making the GET request
- **webdriver** (*selenium.webdriver.chrome.webdriver.WebDriver*) – Webdriver for scraping the page, overrides any default or passed session

Returns Optionally returns a scraped instance instead of modifying inplace if inplace arg is True

Return type return_instance

session = <requests.sessions.Session object>

to_csv (*fp: str*) → None

Write scraped data to .csv at the given filepath

Parameters **fp** (*str*) – Filepath to write data to

to_dict (*metadata: bool = False*) → Dict[str, Any]

Return a dictionary containing all of the data that has been scraped

Parameters **metadata** (*bool*) – Boolean value that determines if metadata specified in self._METADATA_KEYS will be included in the dictionary.

Returns **data_dict** – Dictionary containing the scraped data

Return type Dict[str, Any]

to_json (*fp: str*) → None

Write scraped data to .json file at the given filepath

Parameters **fp** (*str*) – Filepath to write data to

1.7 instascape.scrapers.igtv module

class instascape.scrapers.igtv.**IGTV** (*source: Union[str, bs4.BeautifulSoup, Dict[str, Any]]*)

Bases: *instascape.scrapers.post.Post*

Scraper for an IGTV post

SUPPORTED_DOWNLOAD_EXTENSIONS = ['.mp3', '.mp4', '.png', '.jpg']

download (*fp: str*) → None

Download an image or video from a post to your local machine at the given filepath

Parameters **fp** (*str*) – Filepath to download the image to

embed () → str

Return embeddable HTML str for this post

Returns **html_template** – HTML string with embed markup for this Post

Return type str

get_recent_comments () → List[instascrape.scrapers.comment.Comment]

Returns a list of Comment objects that contain data regarding some of the posts comments

Returns **comments_arr** – List of Comment objects

Return type List[Comment]

scrape (*mapping=None, keys: Optional[List[str]] = None, exclude: Optional[List[str]] = None, headers={'user-agent': 'Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Mobile Safari/537.36 Edg/87.0.664.57'}, inplace=True, session=None, webdriver=None*) → None

Scrape data from the source

Parameters

- **mapping** (*Dict[str, deque]*) – Dictionary of parsing queue's that tell the JSON engine how to process the JSON data
- **keys** (*List[str]*) – List of strings that correspond to desired attributes for scraping
- **exclude** (*List[str]*) – List of strings that correspond to which attributes to exclude from being scraped
- **headers** (*Dict[str, str]*) – Dictionary of request headers to be passed on the GET request
- **inplace** (*bool*) – Determines if data modified inplace or return a new object with the scraped data
- **session** (*requests.Session*) – Session for making the GET request
- **webdriver** (*selenium.webdriver.chrome.webdriver.WebDriver*) – Webdriver for scraping the page, overrides any default or passed session

Returns Optionally returns a scraped instance instead of modifying inplace if inplace arg is True

Return type return_instance

session = <requests.sessions.Session object>

to_csv (*fp: str*) → None

Write scraped data to .csv at the given filepath

Parameters **fp** (*str*) – Filepath to write data to

to_dict (*metadata: bool = False*) → Dict[str, Any]

Return a dictionary containing all of the data that has been scraped

Parameters **metadata** (*bool*) – Boolean value that determines if metadata specified in self._METADATA_KEYS will be included in the dictionary.

Returns **data_dict** – Dictionary containing the scraped data

Return type Dict[str, Any]

to_json (*fp: str*) → None

Write scraped data to .json file at the given filepath

Parameters **fp** (*str*) – Filepath to write data to

1.8 instascape.scrapers.scrape_tools module

instascape.scrapers.scrape_tools.determine_json_type (*json_data: Union[Dict[str, Any], str]*) → str

Return the type of Instagram page based on the JSON data parsed from source

Parameters **json_data** (*Union[JSONDict, str]*) – JSON data that will be checked and parsed to determine what type of page the program is looking at (Profile, Post, Hashtag, etc)

Returns **instagram_type** – Name of the type of page the program is currently parsing or looking at

Return type str

instascape.scrapers.scrape_tools.flatten_dict (*json_dict: Dict[str, Any]*) → Dict[str, Any]

Returns a flattened dictionary of data

Parameters **json_dict** (*dict*) – Input dictionary for flattening

Returns **flattened_dict** – Flattened dictionary

Return type dict

instascape.scrapers.scrape_tools.json_from_html (*source: Union[str, BeautifulSoup], as_dict: bool = True, flatten=False*) → Union[Dict[str, Any], str]

Return JSON data parsed from Instagram source HTML

Parameters

- **source** (*Union[str, BeautifulSoup]*) – Instagram HTML source code to parse the JSON from
- **as_dict** (*bool = True*) – Return JSON as dict if True else return JSON as string
- **flatten** (*bool*) – Flatten the dictionary prior to returning it

Returns **json_data** – Parsed JSON data from the HTML source as either a JSON-like dictionary or just the string serialization

Return type Union[JSONDict, str]

instascape.scrapers.scrape_tools.json_from_soup (*source, as_dict: bool = True, flatten=False*)

instascape.scrapers.scrape_tools.json_from_url (*url: str, as_dict: bool = True, headers={'user-agent': 'Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Mobile Safari/537.36 Edg/87.0.664.57'}, flatten=False*) → Union[Dict[str, Any], str]

Return JSON data parsed from a provided Instagram URL

Parameters

- **url** (*str*) – URL of the page to get the JSON data from
- **as_dict** (*bool = True*) – Return JSON as dict if True else return JSON as string
- **headers** (*Dict[str, str]*) – Dictionary of request headers to be passed on the GET request
- **flatten** (*bool*) – Flatten the dictionary prior to returning it

Returns json_data – Parsed JSON data from the URL as either a JSON-like dictionary or just the string serialization

Return type Union[JSONDict, str]

`instascape.scrapers.scrape_tools.parse_data_from_json(json_dict, map_dict, default_value=nan)`

Parse data from a JSON dictionary using a mapping dictionary that tells the program how to parse the data

`instascape.scrapers.scrape_tools.scrape_posts(posts: List[Post], session: Optional[requests.sessions.Session] = None, webdriver: Optional[selenium.webdriver.chrome.webdriver.WebDriver] = None, limit: Union[int, datetime.datetime, None] = None, headers: dict = {'user-agent': 'Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Mobile Safari/537.36 Edg/87.0.664.57'}, pause: int = 5, on_exception: str = 'raise', silent: bool = True, inplace: bool = False)`

1.9 Module contents

Primary API scraper tools

CHAPTER 2

instascape.exceptions package

2.1 Submodules

2.2 instascape.exceptions.exceptions module

exception `instascape.exceptions.exceptions.InstagramLoginRedirectError` (*message='Instagram is redirecting you to the login page instead of the page you are trying to scrape. This could be occurring because you made too*

Exception that indicates Instagram is redirecting away from the page that should be getting scraped. Can be remedied by logging into Instagram.

exception `instascape.exceptions.exceptions.MissingCookiesWarning`

Bases: `UserWarning`

exception `instascape.exceptions.exceptions.MissingSessionIDWarning`

Bases: `UserWarning`

exception `instascape.exceptions.exceptions.WrongSourceError` (*message='Wrong
input source, use
the correct class'*)

Bases: `Exception`

Exception that indicates user passed the wrong source type to the scraper. An example is passing a URL for a hashtag page to a Profile.

2.3 Module contents

CHAPTER 3

Indices and tables

- `genindex`
- `modindex`
- `search`

i

- `instascape.exceptions`, [13](#)
- `instascape.exceptions.exceptions`, [12](#)
- `instascape.scrapers`, [10](#)
- `instascape.scrapers.hashtag`, [1](#)
- `instascape.scrapers.igtv`, [7](#)
- `instascape.scrapers.location`, [6](#)
- `instascape.scrapers.post`, [2](#)
- `instascape.scrapers.profile`, [4](#)
- `instascape.scrapers.reel`, [5](#)
- `instascape.scrapers.scrape_tools`, [9](#)

D

`determine_json_type()` (in module `instascape.scrapers.scrape_tools`), 9
`download()` (*instascape.scrapers.igtv.IGTV method*), 7
`download()` (*instascape.scrapers.post.Post method*), 2
`download()` (*instascape.scrapers.reel.Reel method*), 5

E

`embed()` (*instascape.scrapers.igtv.IGTV method*), 8
`embed()` (*instascape.scrapers.post.Post method*), 2
`embed()` (*instascape.scrapers.reel.Reel method*), 5

F

`flatten_dict()` (in module `instascape.scrapers.scrape_tools`), 9

G

`get_posts()` (*instascape.scrapers.profile.Profile method*), 4
`get_recent_comments()` (*instascape.scrapers.igtv.IGTV method*), 8
`get_recent_comments()` (*instascape.scrapers.post.Post method*), 3
`get_recent_comments()` (*instascape.scrapers.reel.Reel method*), 5
`get_recent_posts()` (*instascape.scrapers.hashtag.Hashtag method*), 1
`get_recent_posts()` (*instascape.scrapers.location.Location method*), 6
`get_recent_posts()` (*instascape.scrapers.profile.Profile method*), 4

H

`Hashtag` (class in *instascape.scrapers.hashtag*), 1

I

`IGTV` (class in *instascape.scrapers.igtv*), 7
`InstagramLoginRedirectError`, 12
`instascape.exceptions` (module), 13
`instascape.exceptions.exceptions` (module), 12
`instascape.scrapers` (module), 10
`instascape.scrapers.hashtag` (module), 1
`instascape.scrapers.igtv` (module), 7
`instascape.scrapers.location` (module), 6
`instascape.scrapers.post` (module), 2
`instascape.scrapers.profile` (module), 4
`instascape.scrapers.reel` (module), 5
`instascape.scrapers.scrape_tools` (module), 9

J

`json_from_html()` (in module `instascape.scrapers.scrape_tools`), 9
`json_from_soup()` (in module `instascape.scrapers.scrape_tools`), 9
`json_from_url()` (in module `instascape.scrapers.scrape_tools`), 9

L

`Location` (class in *instascape.scrapers.location*), 6

M

`MissingCookiesWarning`, 13
`MissingSessionIDWarning`, 13

P

`parse_data_from_json()` (in module `instascape.scrapers.scrape_tools`), 10
`Post` (class in *instascape.scrapers.post*), 2
`Profile` (class in *instascape.scrapers.profile*), 4

R

`Reel` (class in *instascape.scrapers.reel*), 5

S

`scrape()` (*instascrape.scrapers.hashtag.Hashtag method*), 1
`scrape()` (*instascrape.scrapers.igtv.IGTV method*), 8
`scrape()` (*instascrape.scrapers.location.Location method*), 6
`scrape()` (*instascrape.scrapers.post.Post method*), 3
`scrape()` (*instascrape.scrapers.profile.Profile method*), 4
`scrape()` (*instascrape.scrapers.reel.Reel method*), 5
`scrape_posts()` (*in module instascrape.scrapers.scrape_tools*), 10
`session` (*instascrape.scrapers.hashtag.Hashtag attribute*), 2
`session` (*instascrape.scrapers.igtv.IGTV attribute*), 8
`session` (*instascrape.scrapers.location.Location attribute*), 7
`session` (*instascrape.scrapers.post.Post attribute*), 3
`session` (*instascrape.scrapers.profile.Profile attribute*), 5
`session` (*instascrape.scrapers.reel.Reel attribute*), 6
`SUPPORTED_DOWNLOAD_EXTENSIONS` (*instascrape.scrapers.igtv.IGTV attribute*), 7
`SUPPORTED_DOWNLOAD_EXTENSIONS` (*instascrape.scrapers.post.Post attribute*), 2
`SUPPORTED_DOWNLOAD_EXTENSIONS` (*instascrape.scrapers.reel.Reel attribute*), 5

T

`to_csv()` (*instascrape.scrapers.hashtag.Hashtag method*), 2
`to_csv()` (*instascrape.scrapers.igtv.IGTV method*), 8
`to_csv()` (*instascrape.scrapers.location.Location method*), 7
`to_csv()` (*instascrape.scrapers.post.Post method*), 3
`to_csv()` (*instascrape.scrapers.profile.Profile method*), 5
`to_csv()` (*instascrape.scrapers.reel.Reel method*), 6
`to_dict()` (*instascrape.scrapers.hashtag.Hashtag method*), 2
`to_dict()` (*instascrape.scrapers.igtv.IGTV method*), 8
`to_dict()` (*instascrape.scrapers.location.Location method*), 7
`to_dict()` (*instascrape.scrapers.post.Post method*), 3
`to_dict()` (*instascrape.scrapers.profile.Profile method*), 5
`to_dict()` (*instascrape.scrapers.reel.Reel method*), 6
`to_json()` (*instascrape.scrapers.hashtag.Hashtag method*), 2
`to_json()` (*instascrape.scrapers.igtv.IGTV method*), 8
`to_json()` (*instascrape.scrapers.location.Location method*), 7
`to_json()` (*instascrape.scrapers.post.Post method*), 3

`to_json()` (*instascrape.scrapers.profile.Profile method*), 5
`to_json()` (*instascrape.scrapers.reel.Reel method*), 6

W

`WrongSourceError`, 13