Feed exports

New in version 0.10.

One of the most frequently required features when implementing scrapers is being able to store the scraped data properly and, quite often, that means generating an “export file” with the scraped data (commonly called “export feed”) to be consumed by other systems.

Scrapy provides this functionality out of the box with the Feed Exports, which allows you to generate a feed with the scraped items, using multiple serialization formats and storage backends.

Serialization formats

For serializing the scraped data, the feed exports use the Item exporters. These formats are supported out of the box:

- JSON
- JSON lines
- CSV
- XML

But you can also extend the supported format through the `FEED_EXPORTERS` setting.

JSON

- `FEED_FORMAT`: `json`
- Exporter used: `JsonItemExporter`
- See this warning if you’re using JSON with large feeds.

JSON lines

- `FEED_FORMAT`: `jsonlines`
- Exporter used: `JsonLinesItemExporter`

CSV

- `FEED_FORMAT`: `csv`
- Exporter used: `CsvItemExporter`
- To specify columns to export and their order use `FEED_EXPORT_FIELDS`. Other feed exporters can also use this option, but it is important for CSV because unlike many other export formats CSV uses a fixed header.

XML

- `FEED_FORMAT`: `xml`
- Exporter used: `XmlItemExporter`

Pickle

- `FEED_FORMAT`: `pickle`
- Exporter used: `PickleItemExporter`

Marshal

- `FEED_FORMAT`: `marshal`
- Exporter used: `MarshalItemExporter`

Storages

When using the feed exports you define where to store the feed using a URI (through the `FEED_URI` setting). The feed exports supports multiple storage backend types which are defined by the URI scheme.

The storages backends supported out of the box are:

- Local filesystem
- FTP

- S3 (requires botocore or boto)
- Standard output

Some storage backends may be unavailable if the required external libraries are not available. For example, the S3 backend is only available if the botocore or boto library is installed (Scrapy supports boto only on Python 2).

Storage URI parameters

The storage URI can also contain parameters that get replaced when the feed is being created. These parameters are:

- `%(time)s` - gets replaced by a timestamp when the feed is being created
- `%(name)s` - gets replaced by the spider name

Any other named parameter gets replaced by the spider attribute of the same name. For example, `%(site_id)s` would get replaced by the `spider.site_id` attribute the moment the feed is being created.

Here are some examples to illustrate:

- Store in FTP using one directory per spider:
  - `ftp://user:password@ftp.example.com/scraping/feeds/%(name)s/%(time)s.json`
- Store in S3 using one directory per spider:
  - `s3://mybucket/scraping/feeds/%(name)s/%(time)s.json`

Storage backends

Local filesystem

The feeds are stored in the local filesystem.

- URI scheme: `file`
- Example URI: `file:///tmp/export.csv`
- Required external libraries: none

Note that for the local filesystem storage (only) you can omit the scheme if you specify an absolute path like `/tmp/export.csv`. This only works on Unix systems though.

FTP

The feeds are stored in a FTP server.

- URI scheme: `ftp`
- Example URI: `ftp://user:pass@ftp.example.com/path/to/export.csv`
- Required external libraries: none

S3

The feeds are stored on Amazon S3.

- URI scheme: `s3`
- Example URIs:
  - `s3://mybucket/path/to/export.csv`
  - `s3://aws_key:aws_secret@mybucket/path/to/export.csv`
- Required external libraries: botocore or boto

The AWS credentials can be passed as user/password in the URI, or they can be passed through the following settings:

- `AWS_ACCESS_KEY_ID`
- `AWS_SECRET_ACCESS_KEY`

Standard output

The feeds are written to the standard output of the Scrapy process.

- URI scheme: `stdout`
- Example URI: `stdout:`
- Required external libraries: none

Settings

These are the settings used for configuring the feed exports:

- `FEED_URI` (mandatory)
- `FEED_FORMAT`
- `FEED_STORAGES`
- `FEED_EXPORTERS`
- `FEED_STORE_EMPTY`
- `FEED_EXPORT_ENCODING`

- `FEED_EXPORT_FIELDS`
- `FEED_EXPORT_INDENT`

FEED_URI

Default: `None`

The URI of the export feed. See Storage backends for supported URI schemes.

This setting is required for enabling the feed exports.

FEED_FORMAT

The serialization format to be used for the feed. See Serialization formats for possible values.

FEED_EXPORT_ENCODING

Default: `None`

The encoding to be used for the feed.

If unset or set to `None` (default) it uses UTF-8 for everything except JSON output, which uses safe numeric encoding (`\uXXXX` sequences) for historic reasons.

Use `utf-8` if you want UTF-8 for JSON too.

FEED_EXPORT_FIELDS

Default: `None`

A list of fields to export, optional. Example: `FEED_EXPORT_FIELDS = ["foo", "bar", "baz"]`.

Use FEED_EXPORT_FIELDS option to define fields to export and their order.

When FEED_EXPORT_FIELDS is empty or None (default), Scrapy uses fields defined in dicts or `Item` subclasses a spider is yielding.

If an exporter requires a fixed set of fields (this is the case for CSV export format) and FEED_EXPORT_FIELDS is empty or None, then Scrapy tries to infer field names from the exported data

- currently it uses field names from the first item.

FEED_EXPORT_INDENT

Default: `0`

Amount of spaces used to indent the output on each level. If `FEED_EXPORT_INDENT` is a non-negative integer, then array elements and object members will be pretty-printed with that indent level. An indent level of `0` (the default), or negative, will put each item on a new line. `None` selects the most compact representation.

Currently implemented only by `JsonItemExporter` and `XmlItemExporter`, i.e. when you are exporting to `.json` or `.xml`.

FEED_STORE_EMPTY

Default: `False`

Whether to export empty feeds (ie. feeds with no items).

FEED_STORAGES

Default: `{}`

A dict containing additional feed storage backends supported by your project. The keys are URI schemes and the values are paths to storage classes.

FEED_STORAGES_BASE

Default:

```
{
    '': 'scrapy.extensions.feedexport.FileFeedStorage',
    'file': 'scrapy.extensions.feedexport.FileFeedStorage',
    'stdout': 'scrapy.extensions.feedexport.StdoutFeedStorage',
    's3': 'scrapy.extensions.feedexport.S3FeedStorage',
    'ftp': 'scrapy.extensions.feedexport.FTPFeedStorage',
}
```

A dict containing the built-in feed storage backends supported by Scrapy. You can disable any of these backends by assigning `None` to their URI scheme in `FEED_STORAGES`. E.g., to disable the built-in FTP

storage backend (without replacement), place this in your `settings.py`:

```
FEED_STORAGES = {
    'ftp': None,
}
```

FEED_EXPORTERS

Default: `{}`

A dict containing additional exporters supported by your project. The keys are serialization formats and the values are paths to Item exporter classes.

FEED_EXPORTERS_BASE

Default:

```
{
    'json': 'scrapy.exporters.JsonItemExporter',
    'jsonlines': 'scrapy.exporters.JsonLinesItemExporter',
    'jl': 'scrapy.exporters.JsonLinesItemExporter',
    'csv': 'scrapy.exporters.CsvItemExporter',
    'xml': 'scrapy.exporters.XmlItemExporter',
    'marshal': 'scrapy.exporters.MarshalItemExporter',
    'pickle': 'scrapy.exporters.PickleItemExporter',
}
```

A dict containing the built-in feed exporters supported by Scrapy. You can disable any of these exporters by assigning `None` to their serialization format in `FEED_EXPORTERS`. E.g., to disable the built-in CSV exporter (without replacement), place this in your `settings.py`:

```
FEED_EXPORTERS = {
    'csv': None,
}
```

Previous

Next