# Why Use Scrapy?

It is easier to build and scale large crawling projects.

It has a built-in mechanism called Selectors, for extracting the data from websites.

It handles the requests asynchronously and it is fast.

It automatically adjusts crawling speed using Auto-throttling mechanism.

Ensures developer accessibility.

# Features of Scrapy

Scrapy is an open source and free to use web crawling framework.

Scrapy generates feed exports in formats such as JSON, CSV, and XML.

Scrapy has built-in support for selecting and extracting data from sources either

by XPath or CSS expressions.

Scrapy based on crawler, allows extracting data from the web pages automatically.

# Advantages

Scrapy is easily extensible, fast, and powerful.

It is a cross-platform application framework (Windows, Linux, Mac OS and BSD).

Scrapy requests are scheduled and processed asynchronously.

Scrapy comes with built-in service called Scrapyd which allows to upload projects

and control spiders using JSON web service.

It is possible to scrap any website, though that website does not have API for raw

data access.

# Disadvantages

Scrapy is only for Python 2.7. +

Installation is different for different operating systems.

# Configuration Settings

Scrapy will find configuration settings in the scrapy.cfg file. Following are a few locations:

C:\scrapy(project folder)\scrapy.cfg in the system

scrapy.cfg Deploy the configuration file

You can find the scrapy.cfg inside the root of the project. Scrapy can also be configured using the following environment variables:

# Creating a Project

You can use the following command to create the project in Scrapy:

scrapy startproject project\_name

scrapy genspider mydomain mydomain.com

fetch: It fetches the URL using Scrapy downloader.

runspider: It is used to run self-contained spider without creating a project.

settings: It specifies the project setting value.

shell: It is an interactive scraping module for the given URL.

startproject: It creates a new Scrapy project.

version: It displays the Scrapy version.

view: It fetches the URL using Scrapy downloader and show the contents in a browser.

You can have some project related commands as listed:

crawl: It is used to crawl data using the spider.

check: It checks the items returned by the crawled command.

list: It displays the list of available spiders present in the project.

edit: You can edit the spiders by using the editor.

parse: It parses the given URL with the spider.

bench: It is used to run quick benchmark test (Benchmark tells how many number of pages can be crawled per minute by Scrapy).

Spider is a class responsible for defining how to follow the links through a website and extract the information from the pages.

scrapy. Spider It is a spider from which every other spiders must inherit. It has the following class

The following table shows the fields of scrapy.Spider class:

Field & Description

1, name = It is the name of your spider.

2, allowed\_domains = It is a list of domains on which the spider crawls.

3, start\_urls = It is a list of URLs, which will be the roots for later crawls, where the spider will begin to crawl from.

4, custom\_settings = These are the settings, when running the spider, will be overridden from project wide configuration.

5, crawler = It is an attribute that links to Crawler object to which the spider instance is bound.

6, settings = These are the settings for running a spider.

7, logger = It is a Python logger used to send log messages.

8, from\_crawler(crawler,\*args,\*\*kwargs)

It is a class method, which creates your spider. The parameters are:

crawler: A crawler to which the spider instance will be bound.

args(list): These arguments are passed to the method \_init\_().

kwargs(dict): These keyword arguments are passed to the method \_init\_().

9, start\_requests() = When no particular URLs are specified and the spider is opened for scrapping,

Scrapy calls start\_requests() method.

10, make\_requests\_from\_url(url) = It is a method used to convert urls to requests.

11, parse(response) = This method processes the response and returns scrapped data following more

URLs.

12, log(message[level,component]) = It is a method that sends a log message through spiders logger.

13, closed(reason) = This method is called when the spider closes

# Spider Arguments

Spider arguments are used to specify start URLs and are passed using crawl command

with -a option, shown as follows:

scrapy crawl first\_scrapy -a group = accessories

The following code demonstrates how a spider receives arguments:

import scrapy

class FirstSpider(scrapy.Spider):

name = "first"

def \_\_init\_\_(self, group=None, \*args, \*\*kwargs):

super(FirstSpider, self).\_\_init\_\_(\*args, \*\*kwargs)

self.start\_urls = ["http://www.example.com/group/%s" % group]