How to Crawl the Web Politely with Scrapy

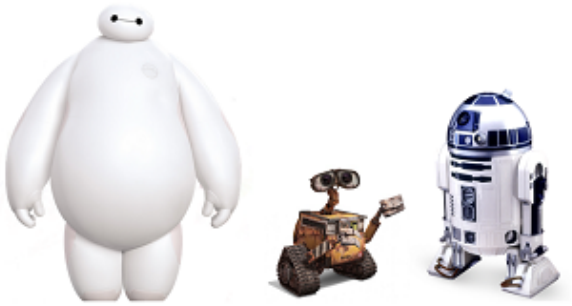
October 25th 2016

TWEET THIS

The first rule of web crawling is you do not harm the website. The second rule of web crawling is you do **NOT** harm the website. We're supporters of the democratization of web data, but not at the expense of the website's owners.

In this post we're sharing a few tips for Scrapy users (Scrapy is a 100% open source web crawling framework) who want polite and considerate web crawlers.

Whether you call them spiders, crawlers, or robots, let's work together to create a world of Baymaxs, WALL-Es, and R2-D2s rather than an apocalyptic wasteland of HAL 9000s, T-1000s, and Megatrons.



Embrace the lovable bots

What Makes a Crawler Polite?

> A polite crawler respects robots.txt
> A polite crawler never degrades a website's performance
> A polite crawler identifies its creator with contact information
> A polite crawler is not a pain in the buttocks of system administrators

robots.txt

Always make sure that your crawler follows the rules defined in the website's robots.txt file. This file is usually available at the root of a website (www.example.com/robots.txt) and it describes what a crawler should or shouldn't crawl according to the Robots Exclusion Standard. Some websites even use the crawlers' user agent to specify separate rules for different web crawlers:

```
User-agent: Some Annoying_Bot
Disallow: / User-Agent: *
Disallow: /*.json
Disallow: /api
Disallow: /post
Disallow: /submit
Allow: /
```

Crawl-Delay

Mission critical to having a polite crawler is making sure your crawler doesn't hit a website too hard. Respect the delay that crawlers should wait between requests by following the robots.txt Crawl-Delay directive.

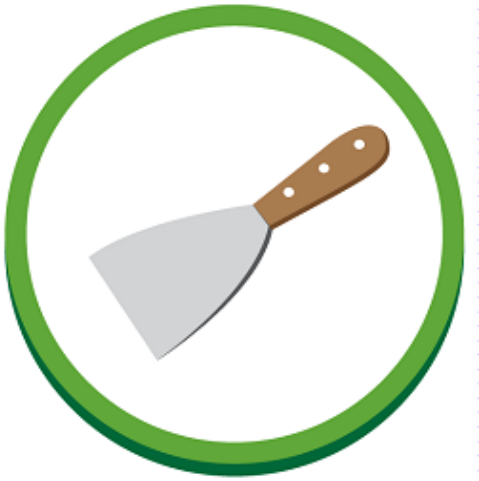
When a website gets overloaded with more requests that the web server can handle, they might become unresponsive. Don't be that guy or girl that causes a headache for the website administrators.

User-Agent

However, if you have ignored the cardinal rules above (or your crawler has achieved aggressive sentience), there needs to be a way for the website owners to contact you. You can do this by including your company name and an email address or website in the request's User-Agent header. For example, Google's crawler user agent is "Googlebot".

How to be Polite using Scrapy

Scrapy is a bit like Optimus Prime: friendly, fast, and capable of getting the job done no matter what. However, much like Optimus Prime and his fellow Autobots, Scrapy occasionally needs to be kept in check. So here's the nitty gritty for ensuring that Scrapy is as polite as can be.



Robots.txt

Crawlers created using Scrapy 1.1+ already respect robots.txt by default. If your crawlers have been generated using a previous version of Scrapy, you can enable this feature by adding this in the project's settings.py:

```
ROBOTSTXT_OBEY = True
```

Then, every time your crawler tries to download a page from a disallowed URL, you'll see a message like this:

```
2016-08-19 16:12:56 [scrapy] DEBUG: Forbidden by robots.txt: <GET http://website.com/login>
```

Identifying your Crawler

It's important to provide a way for sysadmins to easily contact you if they have any trouble with your crawler. If you don't, they'll have to dig into their logs and look for the offending IPs.

Be nice to the friendly sysadmins in your life and identify your crawler via the Scrapy USER_AGENT setting. Share your crawler name, company name and a contact email:

```
USER_AGENT = 'MyCompany-MyCrawler (bot@mycompany.com)'
```

Introducing Delays

Scrapy spiders are blazingly fast. They can handle many concurrent requests and they make the most of your bandwidth and computing power. However, with great power comes great responsibility.

To avoid hitting the web servers too frequently, you need to use the DOWNLOAD_DELAY setting in your project (or in your spiders). Scrapy will then introduce a random delay ranging from 0.5 * DOWNLOAD_DELAY to 1.5 * DOWNLOAD_DELAY seconds between consecutive requests to the same domain. If you want to stick to the exact DOWNLOAD_DELAY that you defined, you have to disable RANDOMIZE_DOWNLOAD_DELAY.

By default, DOWNLOAD_DELAY is set to 0. To introduce a 5 second delay between requests from your crawler, add this to your settings.py:

```
DOWNLOAD_DELAY = 5.0
```

If you have a multi-spider project crawling multiple sites, you can define a different delay for each spider with the download_delay (yes, it's lowercase) spider attribute:

```
class MySpider(scrapy.Spider):
    name = 'myspider'
    download_delay = 5.0
    ...
```

Concurrent Requests Per Domain

Another setting you might want to tweak to make your spider more polite is the number of concurrent requests it will do for each domain. By default, Scrapy will dispatch at most 8 requests simultaneously to any given domain, but you can change this value by updating the CONCURRENT_REQUESTS_PER_DOMAIN setting.

Heads up, the CONCURRENT_REQUESTS setting defines the maximum amount of simultaneous requests that Scrapy's downloader will do for all your spiders. Tweaking this setting is more about your own server performance / bandwidth than your target's when you're crawling multiple domains at the same time.

AutoThrottle to Save the Day

Websites vary drastically in the number of requests they can handle. Adjusting this manually for every website that you are crawling is about as much fun as watching paint dry. To save your sanity, Scrapy provides an extension called AutoThrottle.

AutoThrottle automatically adjusts the delays between requests according to the current web server load. It first calculates the latency from one request. Then it will adjust the delay between requests for the same domain in a way that no more than AUTOTHROTTLE_TARGET_CONCURRENCY requests will be simultaneously active. It also ensures that requests are evenly distributed in a given time span.

To enable AutoThrottle, just include this in your project's settings.py:

```
AUTOTHROTTLE_ENABLED = True
```

Scrapy Cloud users don't have to worry about enabling it because it's already enabled by default.

There's a wide range of settings to help you tweak the throttle mechanism, so have fun playing around!

Use an HTTP Cache for Development

Developing a web crawler is an iterative process. However, running a crawler to check if it's working means hitting the server multiple times for each test. To help you to avoid this impolite activity, Scrapy provides a built-in middleware called HttpCacheMiddleware. You can enable it by including this in your project's settings.py:

```
HTTPCACHE_ENABLED = True
```

Once enabled, it caches every request made by your spider along with the related response. So the next time you run your spider, it will not hit the server for requests already done. It's a win-win: your tests will run much faster and the website will save resources.

Don't Crawl, use the API

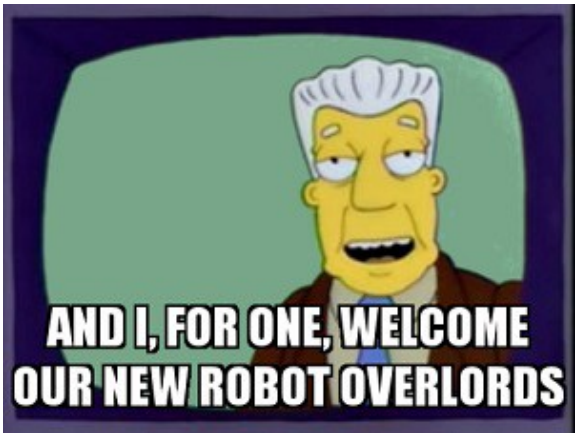
Many websites provide HTTP APIs so that third parties can consume their data without having to crawl their web pages. Before building a web scraper, check if the target website already provides an HTTP API that you can use. If it does, go with the API. Again, it's a win-win: you avoid digging into the page's HTML and your crawler gets more robust because it doesn't need to depend on the website's layout.

Scrapinghub Abuse Report Form

Hey folks using our Scrapy Cloud platform! We trust you will crawl responsibly, but to support website administrators, we provide an abuse report form where they can report any misbehaviour from crawlers running on our platform. We'll kindly pass the message along so that you can modify your crawls and avoid ruining a sysadmin's day. If your crawler's are turning into Skynet and running roughshod over human law, we reserve the right to halt their crawling activities and thus avert the robot apocalypse.

Wrap Up

Let’s all do our part to keep the peace between sysadmins, website owners, and developers by making sure that our web crawling projects are as noninvasive as possible. Remember, we need to band together to delay the rise of our robot overlords, so let’s keep our crawlers, spiders, and bots polite.



To all website owners, help a crawler out and ensure your site has an HTTP API.

Scrapy Cloud is forever free and is the peanut butter to Scrapy’s jelly. Hopefully you learned a few tips for how to both speed up your crawls and prevent abuse complaints.

This post was written by Valdir Stumm(@stummjr), a developer at Scrapinghub.

Please heart the “Recommend” so that others can learn more about how to use Scrapy politely.

Learn more about what web scraping and web data can do for you.

Originally published on the Scrapinghub blog.