Ideas for Google Summer of Code 2014

Contents

Guidelines

Scrapy and Google Summer of Code

Scrapy is a very popular web crawling and scraping framework for Python (10th in Github most trending Python projects) used to write spiders for crawling and extracting data from websites. Scrapy has a healthy and active community, and it's applying for its first Google Summer of Code in 2014.

Information for Students

If you're interested in participating in GSoC 2014 as a student, you should join the scrapy-users mailing list and post your questions and ideas there. You can also join the **#scrapy** IRC channel at **Freenode** to chat with other Scrapy users & developers. All Scrapy development happens at GitHub Scrapy repo.

This page contains a list of curated ideas that have passed all requirements, please add your new ideas to GSoC-2014-Draft-Ideas.

Ideas

Simplified Scrapy add-ons

Brief explanation	Scrapy currently supports many hooks and mechanisms for extending its functionality, but no single entry point for enabling and configuring them. Enabling an extension often requires modifying many settings, often in a coordinated way, which is complex and error prone. This project is meant to provide a unified and simplified way to hook up functionality without dealing with middlewares, pipelines or individual components.
Expected results	Adding or removing extensions should be just a matter of adding or removing lines in a scrapy.cfg file. The implementation must be backward compatible with enabling extension the "old way" (ie. modifying settings directly).
Required skills	Python, general understanding of Scrapy extensions desirable but not required
Difficulty level	Easy
Mentor(s)	Pablo Hoffman

Scrapy Core API cleanup & per-spider settings

Brief explanation	Finish core API cleanup and native support for per-spider settings. This task will involve some refactoring, keeping backwards compatibility for the most popular features. It will require either knowledge of Scrapy internals or some time to get acquainted with them. It should still fit into a 3-month project either way, for a student with the right skill set and motivation.
Expected results	Core API implemented, documented and tested, as documented in SEP-019
Required skills	Python, API design
Difficulty level	Advanced

Mentor(s)	Pablo Hoffman, Nicolas Ramirez, Daniel GraÃ±a

Python 3 support

Brief explanation	Add Python 3.3 support to Scrapy, keeping 2.7 compatibility. The main challenge with this task is that Twisted (a library that Scrapy is built upon) does not yet support Python 3, and Twisted is quite large. However, Scrapy only uses a (very small) subset of Twisted. Students working on this should be prepared to port (or drop) certain parts of Twisted that do not yet support Python 3.
Expected results	Scrapy testing suite should pass most tests and basic spider should work under Python 3.3, at least on Linux (ideally also on Mac/Windows).
Required skills	Python 2 & 3, some Testing and Twisted background
Difficulty level	Intermediate
Mentor(s)	Mikhail Korobov

HTTP API for Scrapy spiders

Brief explanation	Write an HTTP API that wraps any Scrapy spider, it should accept Requests, execute them in Scrapy, and return data extracted by the spider. Scrapy supports crawling in batch mode very well, i.e. a long running process starting from seed requests, extracting and following links and exporting data. Sometimes users would like to reuse the same code to extract interactively. This project provides an API to support this usage and allows scrapy extraction code to be reused from other applications.
Expected results	Working server, Twisted API, docs and tests.
Required skills	Python, Twisted (desired), Scrapy basics
Difficulty level	Intermediate
Mentor(s)	Shane Evans

IPython IDE for Scrapy

Brief explanation	Develop a better IPython + Scrapy integration that would display the HTML page inline in the console, provide some interactive widgets and run Python code against the results. IPython 2.0 release is not too far away, and it should provide a standard protocol for this. Here is a scrapy-ipython proof of concept demo.
Expected results	It should become possible to develop Scrapy spiders interactively and visually inside IPython notebooks
Required skills	Python, JavaScript, HTML, Interface Design, Security
Difficulty level	Advanced
Mentor(s)	Mikhail Korobov, Shane Evans

Scrapy benchmarking suite

Brief explanation	Develop a more comprehensive benchmarking suite. Profile and address CPU bottlenecks found. Address both known memory inefficiencies (which will be provided) and new ones uncovered.
Expected results	Reusable benchmarks, measureable performance improvements.
Required skills	Python, Profiling, Algorithms and Data Structures
Difficulty level	Advanced
Mentor(s)	Mikhail Korobov, Daniel GraÃ±a, Shane Evans

Support for spiders in other languages

Brief explanation	A project that allows users to define a Scrapy spider by creating a stand alone script or executable
Expected results	Demo spiders in a programming languge other than Python, documented API and tests.
Required skills	Python and other programming language
Difficulty level	Intermediate
Mentor(s)	Shane Evans, Pablo Hoffman

Scrapy has a lot of useful functionality not available in frameworks for other programming languages. The goal of this project is to

allow developers to write spiders simply and easily in any programming language, while permitting Scrapy to manage concurrency, scheduling, item exporting, caching, etc. This project takes inspiration from hadoop streaming, a utility allowing hadoop mapreduce jobs to be written in any language.

This task will involve writing a Scrapy spider that forks a process and communicates with it using a protocol that needs to be defined and documented. It should also allow for crashed processes to be restarted without stopping the crawl.

Stretch goals:

- Library support in python and another language. This should make writing spiders similar to how it is currently done in Scrapy
- Recycle spiders periodically (e.g. to control memory usage)
- Use multiple cores by forking multiple processes and load balancing between them.

Modern Scrapyd Web UI

Brief explanation	Scrapyd is an application to deploy and run Scrapy spiders, with a HTTP API and web UI. The web UI is very poor, just static HTMLs with no styling at all. The goal of this task is to make the UI prettier and more powerful.
Expected results	A web UI that is pleasant and efficient to use. It should allow browsing scraped items and logs.
Required skills	Python, Javascript, HTML/CSS
Difficulty level	Easy
Mentor(s)	Pablo Hoffman

Scrapy integration tests

Brief explanation	Add integration tests for different networking scenarios
Expected results	Be able to tests from vertical to horizontal crawling against websites in same and different ips respecting throttling and handling timeouts, retries, dns failures. It must be simple to define new scenarios with predefined components (websites, proxies, routers, injected error rates)
Required skills	Python, Networking and Virtualization
Difficulty level	Intermediate
Mentor(s)	Daniel GraÃ±a

New HTTP/1.1 download handler

Brief explanation	Replace current HTTP1.1 downloader handler with a in-house solution easily customizable to crawling needs. Current HTTP1.1 download handler depends on code shipped with Twisted that is not easily extensible by us, we ship twisted code under *scrapy.xlib.tx* to support running Scrapy in older twisted versions for distributions that doesn't ship uptodate Twisted packages. But this is an ongoing cat-mouse game, the http download handler is an essential component of a crawling framework and having no control over its release cycle leaves us with code that is hard to support. The idea of this task is to depart from current Twisted code looking for a design that can cover current and future needs taking in count the goal is to deal with websites that doesn't follow standards to the letter.
Expected results	A HTTP parser that degrades nicely to parse invalid responses, filtering out the offending headers and cookies as browsers does. It must be able to avoid downloading responses bigger than a size limit, it can be configured to throttle bandwidth used per download, and if there is enough time it can lay out the interface to response streaming
Required skills	Python, Twisted and HTTP protocol
Difficulty level	Advanced
Mentor(s)	Daniel GraÃ±a

New Scrapy signal dispatching

Brief explanation	Profile and look for alternatives to the backend of our signal dispatcher based on pydispatcher lib. Django moved out of pydispatcher many years ago which simplified the API and improved its performance. We are looking to do the same with Scrapy. A major challenge of this task is to make the transition as seamless as possible, providing good documentation and guidelines, along with as much backwards compatibility as possible.
Expected results	The new signal dispatching implemented, documented and tested, with backwards compatibility support.
Required skills	Python

Difficulty level	Easy
Mentor(s)	Daniel GraÃ±a, Pablo Hoffman