"Crawling web sites & Extracting Structured data using Scrapy framework"

A

Project Report

submitted in partial fulfillment of the

requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

With Specialization in
Cloud Computing and Virtualization Technology

by

Name	Roll No.
Karan Veer Singh Udawat	R110211019
Prakhar Agrawal	R110211024
Prashant	R110211026
Sarthak Saxena	R110211036

under the guidance of

Mr. M.V. Kamal

Department of Computer Science & Engineering

Centre for Information Technology

University of Petroleum & Energy Studies

Bidholi, Via Prem Nagar, Dehradun, UK

April – 2015

The innovation driven
E-School

CANDIDATE'S DECLARATION

We hereby certify that the project work entitled **"Crawling web sites & Extracting Structured data using Scrapy framework"** in partial fulfilment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING with specialization in **Cloud Computing and Virtualization Technology** and submitted to the Department of Computer Science & Engineering at Center for Information Technology, University of Petroleum & Energy Studies, Dehradun, is an authentic record of our work carried out during a period from **Jan 2015 – March 2015** under the supervision of **Mr. M.V. Kamal, Asst. Professor Selection Grade CIT(UPES)**.

The matter presented in this project has not been submitted by us for the award of any other degree of this or any other University.

Karan Veer Singh Udawat	**R100211019**
Prakhar Agrawal	**R100211024**
Prashant	**R100211026**
Sarthak Saxena	**R110211036**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 10/04/2015

Mr. M.V. Kamal
Project Guide

Dr. Amit Agarwal
Program Head – B.Tech -Computer Science-CCVT
Center for Information Technology
University of Petroleum & Energy Studies
Dehradun – 248 001 (Uttarakhand)

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide **Mr. M.V. Kamal**, for all advice, encouragement and constant support he has given us through out our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our respected Program Head of the Department, **Dr. Amit Agarwal**, for his great support in doing our project in **"Crawling web sites & Extracting Structured data using Scrapy framework"** at **CIT**.

We are also grateful to **Dr. Manish Prateek, Associate Dean** and **Dr. Kamal Bansal**DeanCoES, UPES for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

Name	**Karan Veer Singh Udawat**	**Prakhar Agrawal**	**Prashant**	**Sarthak Saxena**
Roll No.	**R110211019**	**R110211024**	**R110211026**	**R110211036**

ABSTRACT

“Web Crawling” refers to an application that processes a Web page or the HTML of a Web page to extract data for manipulation as converting the Web page to another format for example HTML to WML. Web Crawling scripts and applications will simulate a person viewing a Web site with a browser by these scripts you can connect to a website and request a page, just like a browser would. The web server will send back the page which you can then manipulate or extract specific information.

A "**Web crawler**" is a Program that automated searching the World Wide Web and that usually use for the purpose of Web indexing. A Web crawler also called as a spider, an ant an automatic indexer or Web scutter.

The project objective is to design and build a Web Crawler tool that is able to track certain attributes of different websites. In this project we are building a crawler for "imdb website " by using "**Scrapy Framework**".

Scrapy is an open source and web crawling collaborative framework for the extraction of data Doing so requires websites in a fast, simple but extensible way. It is an web crawling framework for writing program that crawl websites and extract data from them.

TABLE OF CONTENTS

S.No. Contents Page No

LIST OF FIGURES

S.No. Figure Page No

1. INTRODUCTION

1.1. History

Web Crawling (web harvesting or web tracking) is a software technique for extracting information from websites. Usually, this type of software programs simulate human exploration of the World Wide Web, either implementing low-level Hypertext Transfer Protocol (HTTP), or the addition of a full-fledged web browser, such as Internet Explorer or Mozilla Firefox.

Web crawling is related to the Web indexing that indexes information on the web using a web crawler and it is a universal technique adopted by most search engines. web scraping is more focused on the transformation of unstructured web data, generally in HTML format and structured that can be analyzed and stored in a central database or spreadsheet data calculating location data. Web scraping is also related to web automation that simulates human navigation by using computer software. Some Applications include web scraping used to compare prices online, contact scraping, weather monitoring site data change detection, mash up web, investigation, and integration of web data.

Web scraping software: There are many software tools available on web that can be used for web scraping solutions. This type of software attempts to automatically recognize the data structure of a page or provide a recording interface that eliminates the need to manually write code web scraping or some functions of scripts that can be used to extract and process the content and database interfaces that can store the scraped data into local databases.

Scrapy is an **open source and web crawling** collaborative framework for the extraction of data. Scrapy is written in Python. Scrapy is controlled using command line tools .It is an web crawling framework for writing program that crawl websites and extract data from them.

1.2. Requirement Analysis

Software Tools Used:

- Python 2.7
- Sublime Text
- MySql

Language Used:

- Python
- HTML
- XML

1.3.Main Objective

We use software known as "web crawlers" to discover publicly available websites. The crawler best known is called "Googlebot." Trackers looking web pages and follow the links on these pages, as you would if you were browsing web content. They go from link to link and bring data on those pages back to Google servers

The crawling process begins with a list of web addresses traces of the past and site maps provided by website owners. As our crawlers visit these websites, seeking links to other pages to visit. This type of software give special attention to new sites that changes to existing sites and dead links.

Computer programs that use for crawling the websites determine which sites to search, how many pages and how often to fetch at each site. Google does not accept payment to crawl a web site more frequently. What we think more about having the best possible crawling results, because it's best for users and their business.

Our project objective is that to create a tool "**Web Crawler**" to extract the necessary data from websites and store it in a structured way so that the data can be used for various analytical purposes.

1.4.Sub Objective

Our project objective is that to create a **"Web Crawler"** tool to extract the required data from the websites and store that data in a structured way so that the data can be used for various analytical purposes.

For our project we are using "Scrapy framework" which is an application framework for crawling web sites and extracting structured data which can be used for a wide range of useful applications for data mining, information processing or historical archival.

As we know that **each website has its own unique web crawler or Web spider** and for our project we are creating a **Web Crawler For "Imdb website**". That crawler will be able to get the different elements of the Imdb website and store it in the form of structured data in our database.

2. System Analysis

2.1 Existing System

There are lot of existing crawlers whose list is displayed below:

Crawler	User-agents	HTTP(S) requests user-agent
Googlebot (Google Web search)	Googlebot	Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html) or (rarely used): Googlebot/2.1 (+http://www.google.com/bot.html)
Googlebot News	Googlebot-News (Googlebot)	Googlebot-News
Googlebot Images	Googlebot-Image (Googlebot)	Googlebot-Image/1.0
Googlebot Video	Googlebot-Video (Googlebot)	Googlebot-Video/1.0
Google Mobile	Googlebot-Mobile	[various mobile device types] (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
Google Smartphone	Googlebot	Mozilla/5.0 (iPhone; CPU iPhone OS 6_0 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A5376e Safari/8536.25 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
Google Mobile AdSense	Mediapartners-Google or Mediapartners (Googlebot)	[various mobile device types] (compatible; Mediapartners-Google/2.1; +http://www.google.com/bot.html)
Google AdSense	Mediapartners-Google Mediapartners (Googlebot)	Mediapartners-Google
Google AdsBot landing page quality check	AdsBot-Google	AdsBot-Google (+http://www.google.com/adsbot.html)

2.2Motivation

The main motivation that drew us to make a crawler is because there is no such crawler that crawls IMDB to list down all the data that could be further used for analysis.

IMDB is the largest website containing the highest ranked data related to each and every film out there. There are many companies and analyst who need these details to get used to your own personal or commercial purpose. Similarly, a startup based in Delhi on behalf INC42 gave this project for us to build a tracker and get the data stored in the database for more anaylis as no such system.

There are system that exist, for example. Googlebot but not extract data from the website IMDB well in order to make this work we need a separate crawler to extract data.

The reason we chose scrapy because it provides the best support for writing spider and to load the elements to be extracted. They can be easily defined and then extraction could be easily processed.

Scrapy being open source platform for building crawlers to crawl python based website that is highly preferred and used today.

Again there is a huge market for analysis and data starts making it the best projects that could held to a successful outcome.

2.3Proposed System

The proposed system is based on Scrapy framework that works on spider and item loader which further help the system to extract data from the website.

Scrapy is a collaborative framework for crawling websites and extraction of structured data that can be used for a wide range of useful applications such as data mining, processing information or archive.

Scrapy Although originally designed for screen scraping (more precisely, web scraping), also can be used to extract data using API (like Amazon Associates Web Services) or as a general purpose web crawler.

Steps on how the system will work are:

1 Pick a website

2 Define the data you want to scrape

3 Write a Spider to extract the data

4 Run the spider to extract the data

5 Review scraped data

6 Storing the scraped data

3. Technologies Used

3.1Python

Python is an easy to learn, powerful programming language. It features high data structures efficient level and a simple but effective approach to object-oriented programming(such as Java). Python syntax and dynamic typing, together and with its interpreted feature make it good for scripting and rapid application development in many areas on most platforms language.

The Python interpreter and the extensive standard library are freely available in source or binary form for all platforms from the Python web site that is "**https: //www.python.org/**" and it also freely distributed. This same web site also contains distributions of and pointers to many third party Python modules, programs and free tools and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C ++ (or other languages). Python is also suitable as an extension language for customizable applications.

3.2HTML

Hyper Text Markup Language commonly known as HTML, is the standard markup language used to create web pages is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>). HTML tags most commonly come like this in pair <h1> and </h1>, although some labels represent empty elements and so also unpaired for eg <img>. The first tag in a pair is the start tag, the second tag is the end (also called opening tags and closing tags).

Web browsers read HTML files and then compose them into visible or audible web pages. Browsers do not display HTML tags and scripts, but used to interpret the content of the page. HTML semantic structure of a web site together with the signals for presentation described which is a markup language not a programming language.

HTML elements form the building blocks of all websites. HTML allows images and objects that are embedded and can be used to create interactive forms Provides a means to create structured by denoting structural semantics for text such as heading, lists, links , paragraphs, quotes and other items documents. You can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

Web browsers can also refer to Cascading Style Sheets (CSS) to define the appearance and layout of text and other material. The World Wide Web Consortium (W3C), maintainer of both HTML and CSS standards. CSS defines how HTML elements are to be displayed.

.

3.3XML

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in format which is both readable and machine readable.XML is defined by the W3C XML 1.0 specification and by other several specifications, all are free and open standards.

The design goals of XML generality, simplicity and usability over the Internet. It is in a text data format with strong support via Unicode for human languages. the design of XML focuses on documents which are widely used for the representation of arbitrary data structures such as those used in web services.

There are some schema systems exist to aid in the definition of XML-based languages, while many other application programming interfaces(APIs) have been developed to aid the processing of XML data.

3.4MYSQL

MySQL is one of the most popular and Open Source SQL database management system that is developed distributed and supported by Oracle Corporation.

The MySQL Web site is "http://www.mysql.com/"that provides the latest information about MySQL software.

MySQL is a database management system.

A database is a structured collection of data. It can be anything from a simple shopping list to a picture gallery or the big amounts of information in an organization. To add, delete,access and process data stored in a database management system then MySQL database server is needed. Since computers are very good at handling large amounts of data management systems databases play a central role in computing, as standalone programs, or as part of other applications.

MySQL databases are relational.

A relational database is like Stores data into separate tables rather than putting all the data in a one room. The databases structures are organized into physical files that is optimized for speed. some logical model, with objects like databases, views, tables, rows and columns, gives a flexible programming environment. We establish rules governing relations between different data fields like one to one, one-to-many, unique, mandatory or optional. The database provide these rules so that a data base, application never sees inconsistent, out of date or missing data problems.

The SQL part of "MySQL" stands for "Structured Query Language". SQL is a Query language not a database and used to access databases.

3.4LXML

The lxml XML toolkit is a Pythonic binding for the C libraries libxml2 and libxslt. It is unique because it combines the speed and XML feature integrity of these libraries with the simplicity of a Python API, that is mostly compatible and superior to the well-known ElementTree API. The latest version of Lxml works with all C Python versions from 2.6 to 3.4 .

lxml.etree follows the ElementTree API as much as possible building it on top of native libxml2 tree and If you wanna learn to elementtree, start with the tutorial lxmletree and also look for the ElementTree and its compatibility overview and the ElementTree performance page comparing lxml to the original ElementTree and cElementTree implementations.

3.5OPENSSL

OpenSSL is an open-source implementation of the SSL and TLS protocols in computer networking.its core library,written in the C programming language and it implements basic cryptographic functions and its provides various utility functions. the OpenSSL library use in a variety of computer languages .

Versions are available for most Unix-like operating systems such as including Solaris, Linux, Mac OS X and the other open-source BSD, and operating systems, and Microsoft Windows. IBM provides a port for the System i (OS/400).

OpenSSL is basically based on SSLeay that is by Eric Andrew Young and Tim Hudson, that development of which unofficially ended on December 17, 1998, when Young and Hudson both started to work for RSA Security.

4.Design

4.1. Scrapy Architecture:

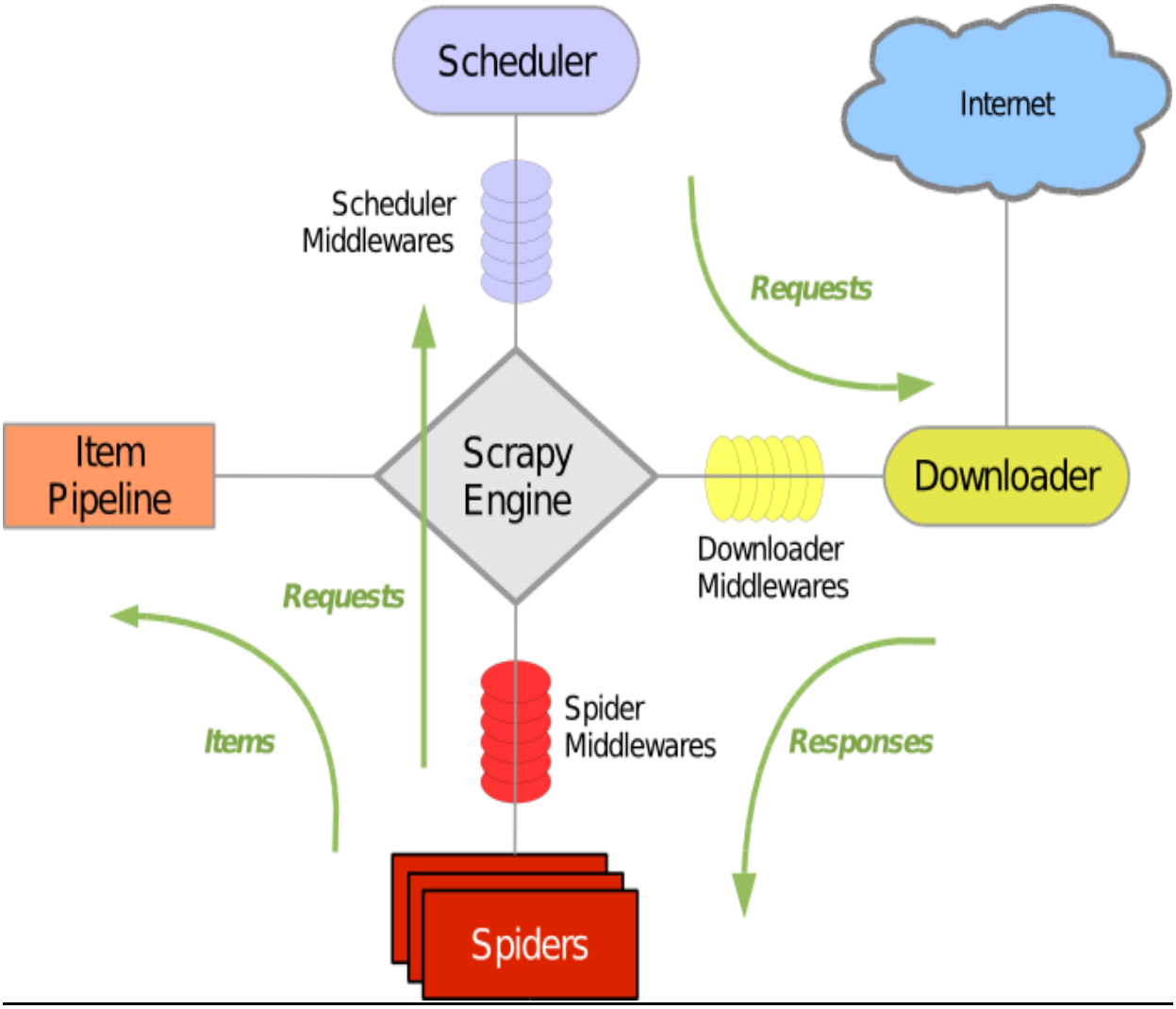

Fig. 1- Architecture of Scrapy

4.2. Components:

Scrapy Engine

Scrapy engine is responsible for controlling the data flow between all components of the scrapy system, and triggering events when certain actions occur.

Scheduler

Scrapy Scheduler receives requests from the engine and enqueues them for feeding them later (also to the engine) when the engine requests them.

Downloader

Scrapy Downloader is responsible for fetching web pages data and feeding them to the engine which, in turn, feeds them to the spiders.

Spiders

Spiders are custom classes written by Scrapy users to parse responses and extract scrapted items from them or additional URLs (requests) to follow. Each spider is able to handle a specific domain (or group of domains).

Item Pipeline

The Item Pipeline is responsible for processing the items once they have been extracted or scraped by the spiders . now tasks include cleansing, validation and persistence such as storing the item in a database.

Downloader middlewares

Downloader middlewares are specific hooks that sit between the Engine and the Downloader and process requests when they pass from the Engine to the Downloader, and responses that pass from Downloader to the Engine. They provide a convenient mechanism for extending Scrapy functionality by plugging custom code.

Spider middlewares

Spider middlewares are specific hooks that sit between the Engine and the Spiders and that are able to process spider input (responses) and output (items and requests). They provide a convenient mechanism for extending Scrapy functionality by plugging custom code.

4.3. Data Flow:

The data flow in Scrapy is controlled by the execution engine, and goes like this:

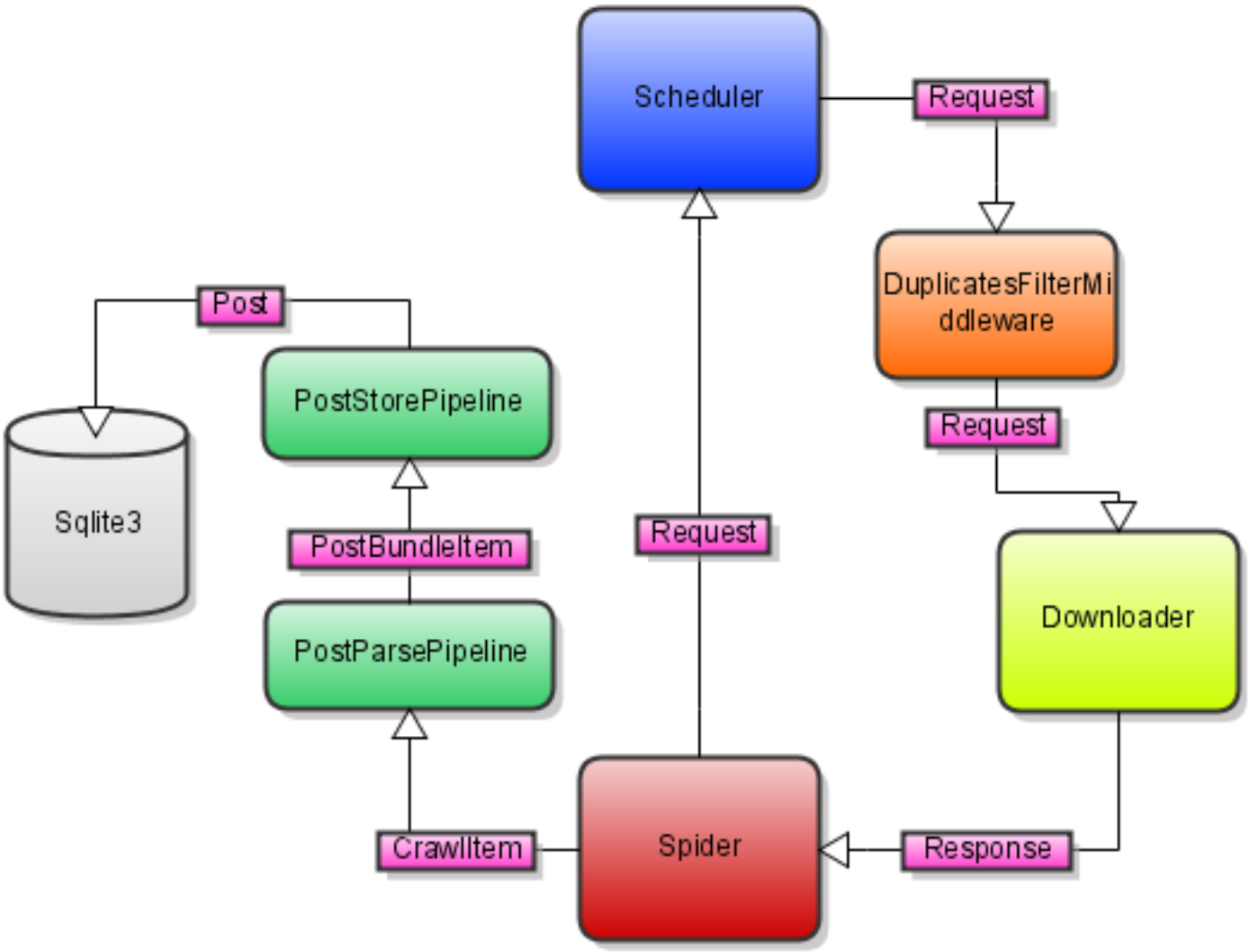

Fig. 2- Data Flow In Scrapy

STEPS:

1. First the Engine opens a domain and locates the Spider that handles the domain, and then asks the spider for the first URLs to crawl.

2. Now the Engine gets the first URLs to crawl from the Spider and schedules .

3. Now the Engine will asks the Scheduler for the next URLs to crawl.

4. The Scheduler will returns the next URLs to crawl to the Engine and then the Engine sends them to the Downloader, passing through the Downloader Middleware (request direction).

5. Once the page finishes downloading the data Downloader generates a Response with that page and sends response to the Engine, passing through the Downloader Middleware (response direction).

6. The Engine get the Response from the Downloader and then sends it to the Spider for processing, now passing through the Spider Middleware (input direction).

7. The Spider processes the Response and returns scraped Items and give new Requests (to follow) to the Engine.

8. The Engine sends scraped Items or data (returned by the Spider) to the Item Pipeline that store data and Requests (returned by spider) to the Scheduler.

9. That process repeats (from step 2) until there are no more requests from the Scheduler, and the Engine closes the domain.

5. IMPLEMENTATION

5.1. Installing Scrapy:

First we have to Install python

After installing Python in our systems, follow these steps before installing Scrapy:

- add the C:\python27\Scripts and C:\python27 folders to the system path by adding those directories to the PATH environment variable

- now Install OpenSSL by following these steps:

  1. gotoOpenSSL page

  2. download theVisual C++ 2008 redistributables

  3. now download OpenSSL

  4. and add the c:\openssl\bin (or similar) directory to your PATH, the same way you added python27 in the first step.

- some binary packages that Scrapy depends on (like lxml and pyOpenSSL) sometime require a compiler available to install, and it will fail if you don't have Visual Studio installed. You can find installers for those in the given links. Make sure your Python version should be supported

  – pywin32: http://sourceforge.net/projects/pywin32/files/

  – Twisted: http://twistedmatrix.com/trac/wiki/Downloads

  – zope.interface: download the egg from zope.interfacepypi page and install it by running easy_installfile.egg

  – lxml: http://pypi.python.org/pypi/lxml/

  – pyOpenSSL: https://launchpad.net/pyopenssl

5.2. Creating a project:

Before we start scraping, we will have set up a new Scrapy project. Enter a directory where you'd like to store your code and then run:

```
scrapystartprojectimdb
```

This will create a tutorial directory with the following contents:

```
imdb/
scrapy.cfg
imdb/
    __init__.py
    items.py
    pipelines.py
    settings.py
spiders/
      __init__.py
      ...
```

These are basically:

scrapy.cfg: the project configuration file

imdb/: the project's python module, you'll later import your code from here.

imdb/items.py: the project's items file.

Imdb/pipelines.py: the project's pipelines file.

imdb/settings.py: the project's settings file.

imdb/spiders/: a directory where you'll later put your spiders.

5.3. Defining Our Item:

Items work like containers that will be loaded with the scraped data, they work like simple python dints but they provide additional protection against populating undeclared fields to prevent typos.

Item are declared by creating a scrapy.Item class and then defining its attributes as scrapy.Field objects, like we will do in an ORM.

Now We begin by the item that we will use to hold the sites scrapted data obtained from dmoz.org, now we want to scrape the name, url and description of the sites, now we define fields for these attributes. we edit items.py, found in the tutorial directory. now Our Item class looks like this:

```
importscrapy

classIMDBItem(scrapy.Item):

movie = scrapy.Field()
rating = scrapy.Field()
desc = scrapy.Field()
cast = scrapy.Field()
budget = scrapy.Field()
location= scrapy.Field()
year = scrapy.Field()
```

This process may seem complicated at first, but defining the item allows us to use other components of Scrapy that need to know how your item looks.

5.4. Our Spider:

Spiders in our project are user-written classes that is used to scrape data from a domain or group of domains.
Spiders define a list of URLs to download and how to follow links, and how to parse the contents of those pages to extract items.

For creating a Spider, we must **subclass scrapy.Spider** and then define the three main mandatory attributes:

name: that identifies the Spider. that must be unique, we can't set the same name for different Spiders.

start_urls: is a list of URLs where the Spider will begin to crawl from. So, the first pages downloaded will be those listed here. The subsequent URLs will be generated successively from data contained in the start URLs.
Where **parse()** is a method for the spider,that will be called when the downloaded Response object of each start URL. That response is passed to the method such as the first and only argument.
This type of method is responsible for parsing the response data and extracting data (as scraped items) and more URLs to follow.
Now The parse() method is in charge of processing and the response and returning scraped data and more URLs to follow as Request objects.

This is the code for our first Spider; save it in a file named dmoz_spider.py under the tutorial/spidersdirectory:

```
importscrapy

classDmozSpider(scrapy.Spider):
name = "imdb"
allowed_domains = ["wikipeida.com"]
start_urls = [
    "http://www.wikipedia.com/TOP25MOVIES",
    "http://www.wikipedia.com/Resources/"

def parse(self, response):
filename = response.url.split("/")[-2]
with open(filename, 'wb') as f:
f.write(response.body)
```

5.5. Crawling:

for our spider to work we have to go to the project's top level directory and run:

```
scrapy crawl imdb
```

The crawl imdb command runs the spider for the dmoz.org domain. You will get an output similar to this:

```
2014-01-23 18:13:07-0400 [scrapy] INFO: Scrapy started (bot: tutorial)
2014-01-23 18:13:07-0400 [scrapy] INFO: Optional features available: ...
2014-01-23 18:13:07-0400 [scrapy] INFO: Overridden settings: {}
2014-01-23 18:13:07-0400 [scrapy] INFO: Enabled extensions: ...
2014-01-23 18:13:07-0400 [scrapy] INFO: Enabled downloader middlewares: ...
2014-01-23 18:13:07-0400 [scrapy] INFO: Enabled spider middlewares: ...
2014-01-23 18:13:07-0400 [scrapy] INFO: Enabled item pipelines: ...
2014-01-23 18:13:07-0400 [dmoz] INFO: Spider opened
2014-01-23 18:13:08-0400 [dmoz] DEBUG: Crawled (200) <GET
http://www.dmoz.org/Computers/Programming/Languages/Python/Resources/> (referer: None)
2014-01-23 18:13:09-0400 [dmoz] DEBUG: Crawled (200) <GET
http://www.dmoz.org/Computers/Programming/Languages/Python/Books/> (referer: None)
2014-01-23 18:13:09-0400 [dmoz] INFO: Closing spider (finished)
```

6.Output Screens

```
2015-04-12 13:49:02+0530 [tutorial] DEBUG: Retrying <GET http://www.imdb.com/title/tt2802144/?ref_=chttp_tt_245> (failed 1 times): User timeout caused connection failure: Getting http://www.imdb.com/title
/tt2802144/?ref_=chttp_tt_245 took longer than 180 seconds..
2015-04-12 13:49:02+0530 [tutorial] DEBUG: Retrying <GET http://www.imdb.com/title/tt0070511/?ref_=chttp_tt_244> (failed 1 times): User timeout caused connection failure: Getting http://www.imdb.com/title
/tt0070511/?ref_=chttp_tt_244 took longer than 180 seconds..
2015-04-12 13:49:03+0530 [tutorial] DEBUG: Crawled (200) <GET http://www.imdb.com/title/tt2802144/?ref_=chttp_tt_245> (referer: http://www.imdb.com/chart/top?ref_=nv_ch_250_4)
2015-04-12 13:49:03+0530 [tutorial] DEBUG: Scraped from <200 http://www.imdb.com/title/tt2802144/?ref_=chttp_tt_245>
       {'AspectRatio': u' 2.35 : 1\n    ',
        'Budget': u'\n     Budget:       $81,000,000       \n\n     (estimated)\n     ',
        'CastMembers': [{'ActorName': u'<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">\n<a href="/name/nm6244464/?ref_=tt_cl_t1" itemprop="url"> <span class="itempro
p" itemprop="name">Adrian Quinton</span>\n</a>         </td>',
        'CharacterName': u'<div>\n           <a href="/character/ch0503766/?ref_=tt_cl_t1">Terrorist</a> \n                \n            </div>',
        'Ranking': 1},
                        {'ActorName': u'<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">\n<a href="/name/nm0000147/?ref_=tt_cl_t2" itemprop="url"> <span class="itempro
p" itemprop="name">Colin Firth</span>\n</a>         </td>',
        'CharacterName': u'<div>\n           <a href="/character/ch0500183/?ref_=tt_cl_t2">Harry Hart</a> /  \n           Galahad \n                \n            </div>',
        'Ranking': 2},
                        {'ActorName': u'<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">\n<a href="/name/nm0835016/?ref_=tt_cl_t3" itemprop="url"> <span class="itempro
p" itemprop="name">Mark Strong</span>\n</a>         </td>',
        'CharacterName': u'<div>\n           <a href="/character/ch0500184/?ref_=tt_cl_t3">Merlin</a> \n                \n            </div>',
        'Ranking': 3},
                        {'ActorName': u'<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">\n<a href="/name/nm5956650/?ref_=tt_cl_t4" itemprop="url"> <span class="itempro
p" itemprop="name">Jonno Davies</span>\n</a>         </td>',
        'CharacterName': u'<div>\n           Lee \n                \n            </div>',
        'Ranking': 4},
                        {'ActorName': u'<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">\n<a href="/name/nm0202603/?ref_=tt_cl_t5" itemprop="url"> <span class="itempro
p" itemprop="name">Jack Davenport</span>\n</a>         </td>',
        'CharacterName': u'<div>\n           Lancelot \n                \n            </div>',
        'Ranking': 5},
                        {'ActorName': u'<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">\n<a href="/name/nm7014019/?ref_=tt_cl_t6" itemprop="url"> <span class="itempro
p" itemprop="name">Alex Nikolov</span>\n</a>         </td>',
        'CharacterName': u'<div>\n           <a href="/character/ch0500182/?ref_=tt_cl_t6">Little Eggsy</a> \n                \n            </div>',
        'Ranking': 6},
                        {'ActorName': u'<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">\n<a href="/name/nm0418379/?ref_=tt_cl_t7" itemprop="url"> <span class="itempro
p" itemprop="name">Samantha Womack</span>\n</a>         </td>',
        'CharacterName': u'<div>\n           <a href="/character/ch0503695/?ref_=tt_cl_t7">Michelle Unwin</a> \n                \n            </div>',
        'Ranking': 7},
                        {'ActorName': u'<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">\n<a href="/name/nm0000434/?ref_=tt_cl_t8" itemprop="url"> <span class="itempro
p" itemprop="name">Mark Hamill</span>\n</a>         </td>',
        'CharacterName': u'<div>\n           Professor Arnold \n                \n            </div>',
        'Ranking': 8},
                        {'ActorName': u'<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">\n<a href="/name/nm0867677/?ref_=tt_cl_t9" itemprop="url"> <span class="itempro
p" itemprop="name">Velibor Topic</span>\n</a>         </td>',
        'CharacterName': u'<div>\n           Big Goon \n                \n            </div>',
        'Ranking': 9},
                        {'ActorName': u'<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">\n<a href="/name/nm1154749/?ref_=tt_cl_t10" itemprop="url"> <span class="itempr
op" itemprop="name">Sofia Boutella</span>\n</a>         </td>',
        'CharacterName': u'<div>\n           Gazelle \n                \n            </div>',
        'Ranking': 10},
                        {'ActorName': u'<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">\n<a href="/name/nm0000168/?ref_=tt_cl_t11" itemprop="url"> <span class="itempr
op" itemprop="name">Samuel L. Jackson</span>\n</a>         </td>',
        'CharacterName': u'<div>\n           <a href="/character/ch0500185/?ref_=tt_cl_t11">Valentine</a> \n                \n            </div>',
        'Ranking': 11},
                        {'ActorName': u'<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">\n<a href="/name/nm0000323/?ref_=tt_cl_t12" itemprop="url"> <span class="itempr
op" itemprop="name">Michael Caine</span>\n</a>         </td>',
        'CharacterName': u'<div>\n           <a href="/character/ch0502806/?ref_=tt_cl_t12">Arthur</a> \n                \n            </div>',
        'Ranking': 12},
                        {'ActorName': u'<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">\n<a href="/name/nm5473782/?ref_=tt_cl_t13" itemprop="url"> <span class="itempr
op" itemprop="name">Taron Egerton</span>\n</a>         </td>',
        'CharacterName': u'<div>\n           <a href="/character/ch0500182/?ref_=tt_cl_t13">Gary \'Eggsy\' Unwin</a> \n                \n            </div>',
        'Ranking': 13},
                        {'ActorName': u'<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">\n<a href="/name/nm1052141/?ref_=tt_cl_t14" itemprop="url"> <span class="itempr
op" itemprop="name">Geoff Bell</span>\n</a>         </td>',
        'CharacterName': u'<div>\n           Dean \n                \n            </div>',
```

Fig. 3- Sample IMDB Crawl

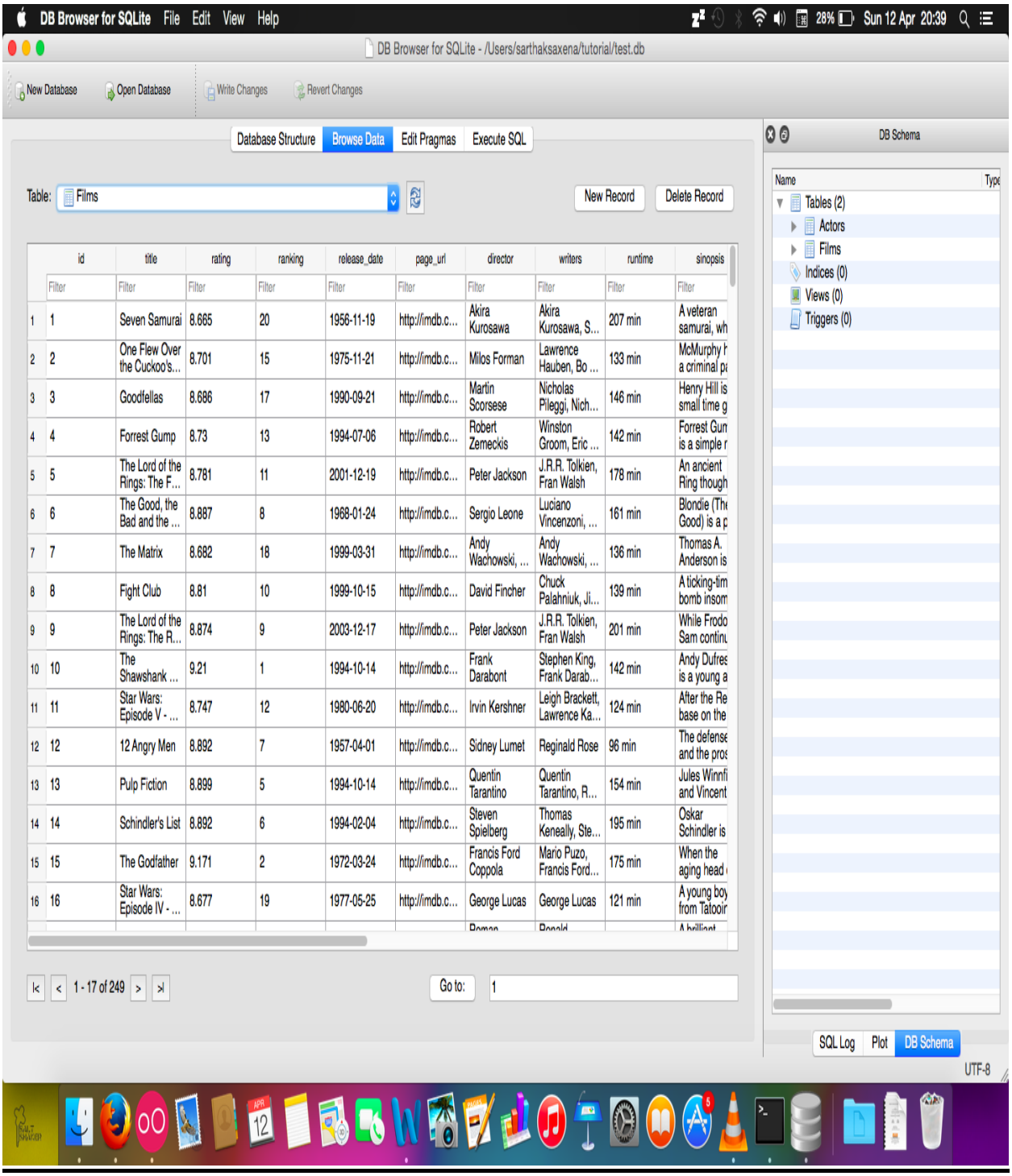

Fig. 4- Database Content

7. Limitations and Future Enhancements

There are a lot of limitations:

1. Xpath to be individually mentioned
2. Standalone spider for each items
3. Isolated spiders for each website
4. Can’t be combined to a universal spider
5. Each set of different website data has different spider to extract the data

In the future the spiders could be enhanced with:

1. Writing a universal spider
2. Xpath should work on aggregator items instead of the particular item

8. References

- https://pypi.python.org/pypi/Scrapy
- https://github.com/scrapy/scrapy
- http://doc.scrapy.org/en/latest/
- Burner, M. (1977), "Crawling Towards Eternity: Building an Archive of the World Wide Web," Web Techniques Magazine 2, 5.
- Cho, J., H. Garcia-Molina, and L. Page (1998), "Efficient Crawling Through URL Ordering," In Proceedings of the Seventh International World Wide Web Conference, pp. 161–172.
- Gray, M., "Internet Growth and Statistics: Credits and Background," www.mit.edu/people/mkgray/net/background.html.
- Miller, R.C. and K. Bharat (1998), "SPHINX: A Framework for Creating Personal, Site-Specific Web Crawlers," In Proceedings of the Seventh International World Wide Web Conference, pp. 119–130.
- Miller, R.C. and K. Bharat (1998), "SCRAPY: A Framework for Creating Personal, Site-Specific Web Crawlers," In Proceedings of the Seventh International World Wide Web Conference, pp. 119–130.
- Pinkerton, B. (1994), "Finding What People Want: Experiences with the WebCrawler," In Proceedings of the Second International World Wide Web Conference
- Smith, Z. (1997), "The Truth About the Web: Crawling Towards Eternity," Web Techniques Magazine 2, 5.
- Scrappy Project Management: The 12 Predictable and Avoidable Pitfalls That Every Project Faces By Kimberly Wiefling
- C. Castillo, "Effective web crawling," in ACM SIGIR Forum, vol. 39, no. 1. ACM, 2005, pp. 55–56
- http://aperture.sourceforge.net/tutorial/crawlers.html
- http://www.imdb.com/chart/top