# Scraping

## Scraper class:

The Scraper class contains all the method useful in order to perform the scraping activity. In particular the server can call the main method *scrape()* for the purpose of start a new scraping session.

Our aim is to create first and then update a Database that store the information about the article published from the major Italian newspaper web-site (la Repubblica, il Corriere, Sky TG 24 and ANSA, even if the code was arranged in order to make adding new newspaper website easy). This is made in a structured way, distinguishing the authors, the title, the text, and etcetera.

## Scraping activity:

The scraping activity can be divided in two separated parts: retrieving the link of the new article and analyze the content of the article found.

### Twitter’s article links retrieval

In order to perform this first part, we choose to retrieve the link through Twitter for several reason the main ones are:

* The tweet’s structure is fixed, while the homepage of the newspaper websites are a bit messy and of course to analyze a fixed structure in order to fine a string that match your requirement is easier.
* Every tweet has an associated id, so it’s also easier search the find the link of the new article pushing aside the other ones.
* Twitter accounts of the major newspapers are constantly updated
* Finding the article link through Twitter, and more in particular through its specific API, reduces the request to the newspaper websites, and this is a good news because we have always the risk of being banned form the website because of an high number of request!

The already mentioned used API is *Twitter4j*;in particular the main class used is *Status* that represent one single status of a user.

Every time that a new scraping session start a new instance of *Twitter* class is created by the method *getTwitterFactory()* and for every newspapers’ Twitter account we try to retrieve the id associated to the last Tweet scraped (that is stored in a file called lastTweetId). If the file doesn’t exist, or we don’t have information about the account to scrape, we chose to take into consideration the last 1000 tweets posted by the user. Otherwise, if we have information about the last tweet scraped, we retrieve all the tweets up to this one.

After retrieving the new tweet, we store the id of the more recent tweet of each Newspaper in the file lastTweetId in order to guarantee that an article isn’t analyzed twice. At the end we have some tweets from which using the method *getArticleLink()* we extract the link of the article that eventually are related to that tweets.

### Retrieving information from websites

As regard the article analysis the code, organized in methods that are called using the convention *scrape+NewspaperName(),* could become messy because of the various possible structure of the web page. In order to retrieve much information as possible, during the design phase we used a log file that stores the part of the Article that our program wasn’t able to find with the associated link.

In order to perform a fair scrape, but trying to reduce as much as possible the time required at the same time, we decided to send a request to each website every 30 seconds.

### Workflow of scraping process

The behavior of the program is the following: assuming that we are analyzing 4 newspaper, and we have at least one article that is available and is waiting to be analyzed for each newspaper, we begin finding a link of an article associated to the newspaper number 1. Then, we make a request to that web page and we analyze it. We repeat these operations for each newspaper, and finally the program stops and wait for 30 seconds.

This operation is repeated while there are no more article left.