Article
Lucas Enard · Aug 24, 2022 7m read

Open Exchange

Web Scraping in IRIS using only Python

This GitHub is the simplest way to scrap using IRIS and Python, all of that already incorporated in an IRIS PRODUCTION.

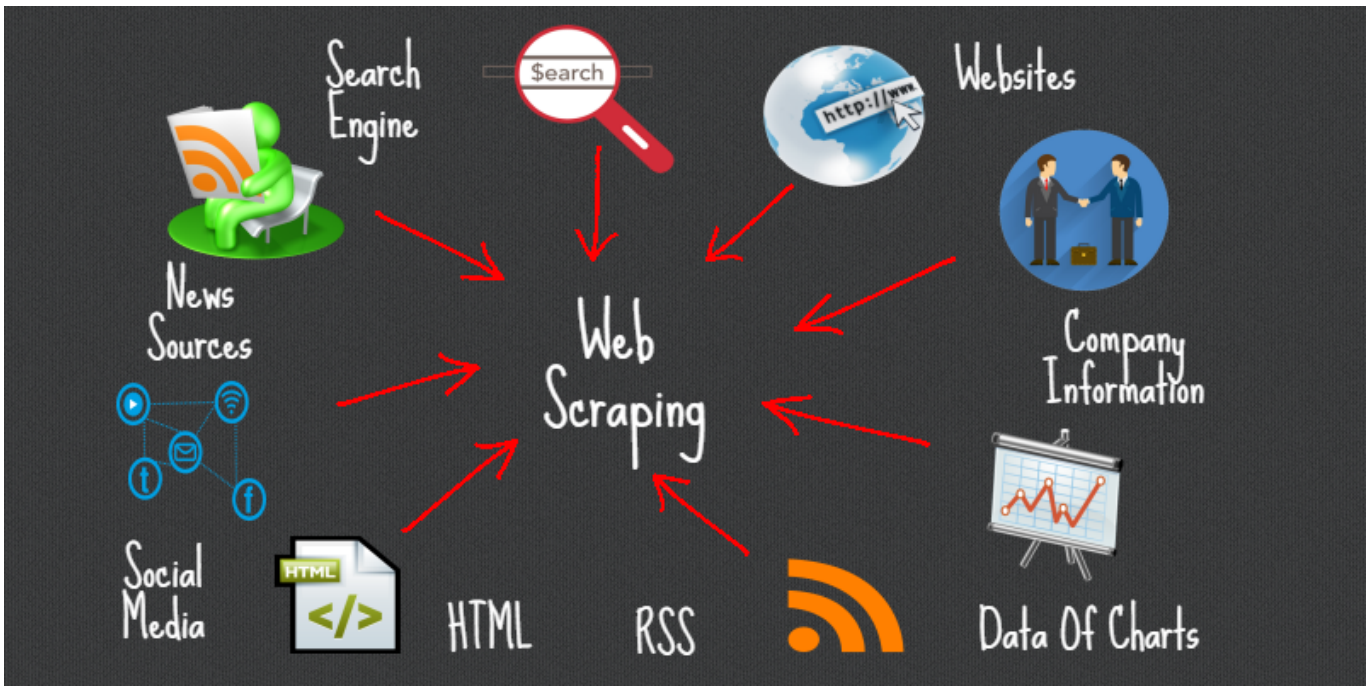
From here you can build any IRIS production in full Python or in ObjectScript as this module is interoperable.

See for more information

1. IRIS-WEB-SCRAPING

2. What is Web Scraping:

In simple terms, **Web scraping**, **web harvesting**, or **web data extraction** is an automated process of collecting large data(unstructured) from websites. The user can extract all the data on particular sites or the specific data as per the requirement. The data collected can be stored in a structured format for further analysis.



Steps involved in web scraping:

1. Find the URL of the webpage that you want to scrape
2. Select the particular elements by inspecting
3. Write the code to get the content of the selected elements
4. Store the data in the required format

It ' s that simple !!

2.1. The popular libraries/tools used for web scraping are:

- Selenium – a framework for testing web applications
- BeautifulSoup – Python library for getting data out of HTML, XML, and other markup languages
- Pandas – Python library for data manipulation and analysis

2.2. The BS4 tool

What is Beautiful Soup ?

Beautiful Soup is a pure Python library for extracting structured data from a website. It allows you to parse data from HTML and XML files. It acts as a helper module and interacts with HTML in a similar and better way as to how you would interact with a web page using other available developer tools.

- It usually saves programmers hours or days of work since it works with your favorite parsers like lxml and html5lib to provide organic Python ways of navigating, searching, and modifying the parse tree.
- Another powerful and useful feature of beautiful soup is its intelligence to convert the documents being fetched to Unicode and outgoing documents to UTF-8. As a developer, you do not have to take care of that unless the document intrinsic doesn't specify an encoding or Beautiful Soup is unable to detect one.
- It is also considered to be **faster** when compared to other general parsing or scraping techniques.

3. Using full Python to web scrap on IRIS

3.1. The Production

3.1.1. Starting the Production

While in the iris-web-scraping folder, open a terminal and enter :

```
docker-compose up
```

The very first time, it may take a few minutes to build the image correctly and install all the needed modules for Python.

3.1.2. Access the Production

Following this link, access the production : Access the Production

3.1.3. Closing the Production

```
docker-compose down
```

3.2. Step 1 : Find the URL of the webpage that you want to scrape.

The example url here is :

url : "http://quotes.toscrape.com/"

The webpage that we are gonna scrape data from is a simple website for webscraping training, this is one of the simplest page to scrap, but if you are interested you can try the other harder challenges by modifying the code.

We are going to scrap the Quotes and the Authors from this page.

We will be using two Python libraries.
These were automatically installed at start up.

- **requests** Requests is a HTTP library for the Python programming language. The goal of the project is to make HTTP requests simpler and more human-friendly.
- **bs4 for BeautifulSoup** Beautiful Soup is a Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.

3.3. Step 2 : Select the required elements by inspecting

If you go on "http://quotes.toscrape.com/", and inspect the page using your browser (right click anywhere on the page and press Inspect), you will be able to see the elements of the html and you'd be able to understand what to scrap.

As you can see, we have multiples div class="quote" that contains a quote each, quote that we want to scrap. Then, in each of these div, we have a span class="text" and a small class="author".

We now know what we want to gather and how to access them.

NOTE that you can Inspect using the production too

3.4. Step 3 : Understand the code to get the content of the selected elements

First we need to requests the HTML from the website and parse it into a bs4 object :

```
req = requests.get(request.url)
soupdata = bs4.BeautifulSoup(req.text, features="html.parser")
```

Here is the code that need to be changed for another webpage or another type of scraping :

Access the file src/python/bo/py and go in the onscraprequest function.

We will be using the findAll functionality on BeautifulSoup to look for all the tags which contains the type div and the class quote :

```
divs = soupdata.findAll("div",{"class":"quote"})
```

Then, for each quote, we want to get the type span and class text, and the type small and class author :

```
for i in range(len(divs)):
    text = divs[i].find("span",{"class":"text"}).text
    author = divs[i].find("small", {"class":"author"}).text
```

We then put all those results in our IRIS message and send them back to you, the user.

3.5. Step 4 : Use the production

You must access the Production following this link :

```
http://localhost:52795/csp/irisapp/EnsPortal.ProductionConfig.zen?PRODUCTION=iris.Production
```

And connect using :

SuperUser as username and SYS as password.

Now you can Start the production.

3.5.1. Inspecting

If you wish to use the production to inspect it's possible, it may not be as practical as using your brower but some people may find it interesting.

To call the inspecting, click on the Python.ScrapingOperation, and select in the right tab Actions, you can Test the production.

In this test window, select :

Type of request : Grongier.PEX.Message

For the classname you must enter :

```
msg.InspectRequest
```

And for the json, you must enter the url you want to inspect :

```
{
  "url":"http://quotes.toscrape.com/"
}
```

From here press Invoke Testing Service and watch the visual trace.

By going on the last message and clicking on contents you shall see the inspected data form the url.

3.5.2. Scraping

To call the scraping, click on the Python.ScrapingOperation, and select in the right tab Actions, you can Test the production.

In this test window, select :

Type of request : Grongier.PEX.Message

For the classname you must enter :

```
msg.ScrapRequest
```

And for the json, you must enter the url you want to scrap :

```
{
  "url":"http://quotes.toscrape.com/"
}
```

From here press Invoke Testing Service and watch the visual trace.

By going on the last message and clicking on contents you shall see the scraped data.

4. Conclusion

Here is the simplest example of scraping, it can be easily used by anyone and is implemented on IRIS, this means that with just a few tweaks you can connect this Operation to a CRUD API or to a automatic service that gather data from the web and input it into the IRIS DATABASE
This last link is in fact a link to a Formation in Python on IRIS that shows how to use this module properly and how to connect to the IRIS DB or an external PostGres DB and doing so using a CRUD API.

4.1. Credits

See this post on the DC as my inspiration to do this GitHub.

#Big Data #Databases #Python #Tools #Visualization #InterSystems IRIS #VSCode
Check the related application on InterSystems Open Exchange

Source URL:https://community.intersystems.com/post/web-scraping-iris-using-only-python