

Information Integration on the Web: Homework 1

Due on January 23, 2015

Professor Ambite & Knoblock

Tushar Tiwari

STATEMENT OF INTEGRITY

I, Tushar Tiwari, declare that the submitted work is original and adheres to all University policies and acknowledge the consequences that may result from a violation of those rules.

SCREENSHOTS

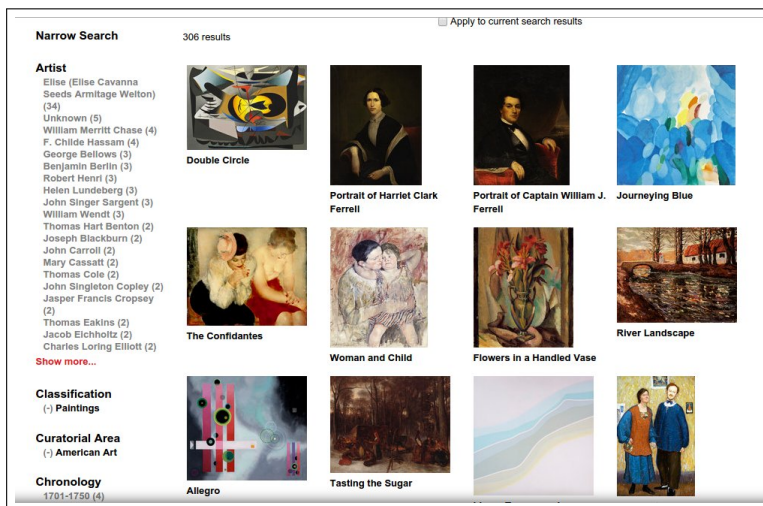
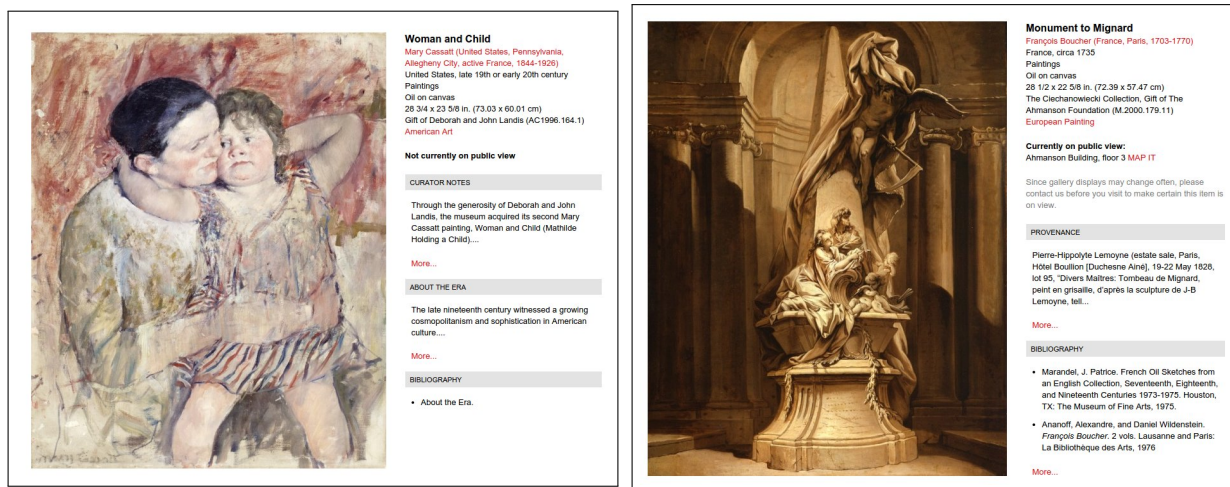


Figure 1: The search page with the option American Paintings already chosen.

This provides us with a custom url that can be exploited to navigate and then extract all paintings.
[http://collections.lacma.org/search/site/?page=0&f\[0\]=bm_field_has_image%3Atrue&f\[1\]=im_field_curatorial_area%3A2&f\[2\]=im_field_classification%3A22](http://collections.lacma.org/search/site/?page=0&f[0]=bm_field_has_image%3Atrue&f[1]=im_field_curatorial_area%3A2&f[2]=im_field_classification%3A22)



(a) An american painting

(b) A european painting

Figure 2: Examples of the paintings on the website

SAMPLE RECORD

Let's look at a sample painting and the output that it produces. For purposes of demonstration the link to this sample painting is hard-coded into the program so that the output produced is only for the painting in consideration.



Figure 3: The chosen painting found at <http://collections.lacma.org/node/184481>.

```

179 def scrape_page(ROOT_URL, hrefs):
180     coll = []
181     for (name, href, _id) in hrefs:
182         URL = ROOT_URL + href
183         URL = r"http://collections.lacma.org/node/184481"
184         try:
185             response = requests.get(URL)
186             soup = BeautifulSoup(response.text).find('div',
187             {'id': 'content'})
188             result = scrape_painting(name, _id, URL, soup)
189             coll.append(result)
190         except:

```

(a) Hardcoding the url into the code

```

tushart91@DEXTER ~info/Homework/HW1/HW1* <master*>
~$ make
rm -f output*.json
rm -f *.pyc
./python-3/bin/python3.4 agent.py
Total Paintings: 1
Combined Total Paintings: 1
print(json.dumps(object[
tushart91@DEXTER ~info/Homework/HW1/HW1* <master*>

```

(b) Execution of make command

Figure 4: Sample Run

Note:

- The url for the painting is hardcoded only for purposes of demonstration. The usual run will automatically extract image urls from the search page.
- Not all paintings will have all data fields associated with them since the data is unavailable for those paintings. These missing fields have been ignored as they serve no purpose except for increasing the file size which is not desirable.

```
1  [
2    {
3      "_id": 184481,
4      "acquired_from": "Bequest of Fannie and Alan Leslie",
5      "acquired_id": "M.2006.73.2",
6      "artist": {
7        "birthplace": {
8          "country": "United States"
9        },
10       "dates": {
11         "born": "1899",
12         "died": "1978"
13       },
14       "link": "http://collections.lacma.org/node/165564",
15       "name": {
16         "first": "Charles",
17         "last": "Howard",
18         "middle": "Houghton"
19       }
20     },
21     "category": "american",
22     "copyright": "Estate of Charles Houghton Howard",
23     "created_country": "United States",
24     "created_year": "1950",
25     "dimensions": {
26       "depth": "3.81",
27       "height": "82.55",
28       "units": "cm",
29       "width": "105.41"
30     },
31     "display": "Framed",
32     "image_link": "http://collections.lacma.org/sites/default/files/
33       remote_images/piction/ma-31205657-WEB.jpg",
34     "link": "http://collections.lacma.org/node/184481",
35     "name": "Double Circle",
36     "on_view": "Art of the Americas Building, floor 3",
37     "painting_type": "Oil on canvas"
38   }
39 ]
```

Figure 5: Output JSON of sample run

CHALLENGES

- Unicode characters are a problem in Python 2 since the default string supports only ascii characters. Due to this, certain string operations become very complicated. The problem was overcome by porting the code to Python 3 where the default string supports unicode.
- Extracting the artist's details required complex regular expressions since all information pertaining to the artist, like name, date of birth, place of birth, was provided in a single string in the html.

TOOLS USED

PYTHON WITH BEAUTIFULSOUP

Python is one of the most easiest yet powerful languages out there which works cross-platform. Installing and distributing dependencies is also quite easy. Beautiful Soup is a Python library designed for quick turnaround projects like screen-scraping. It parses anything you give it, and does the tree traversal stuff for you. You can tell it "Find all the links", or "Find all the links of class externalLink", or "Find all the links whose urls match "foo.com", or "Find the table heading that's got bold text, then give me that text.". Some of the features that make it powerful are:

- Provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application
- Automatically converts incoming documents to Unicode and outgoing documents to UTF-8. You don't have to think about encodings, unless the document doesn't specify an encoding and Beautiful Soup can't detect one. Then you just have to specify the original encoding.
- Sits on top of popular Python parsers like lxml and html5lib, allowing you to try out different parsing strategies or trade speed for flexibility.

FIELDS NOT EXTRACTED

- The size of the painting in inches is not extracted because the same size is also presented in centimeters which is more accurate.
- The category of art, i.e. paintings, sculptures, etc., is skipped because we have hardcoded the category to be paintings in our url. Thus, all results fetched are paintings.
- The museum disclaimers are also overlooked because they are not related to the painting.