CHAPTER 3

Scraping Magic

In this chapter we will scrap the web site of Magic the gathering and in particular the card database. (Yes I play Magic not super good but well I have fun). Here is one example http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=389430 as shown in Figure 3-1. Now we will try to show you how we explore the HTML page using the excellent Pharo inspector: diving in the tree nodes and checking live their attributes or children is simply super cool.

3.1 Getting a tree

The first thing was to make sure that we can get a tree from the web page. For this task we used the `XMLHTMLParser` class and sends it the message `parseURL:`. How did we find this message... Simply looking on the class side methods of the class. How did we find the class, well looking at the subclass of `XMLDOMParser` because HTML is close to XML or the inverse :).

```
| tree |
tree := (XMLHTMLParser parseURL:
    'http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=389430')
```

3.2 First the card visual

First we would like to grab the card visual because this is fun and cool. When we open the card visual in a separate window we see that the url is http://gatherer.wizards.com/Handlers/Image.ashx?multiverseid=389430&type=card. Therefore we started to look for Handlers in the nodes as shown in Figure 3-2.

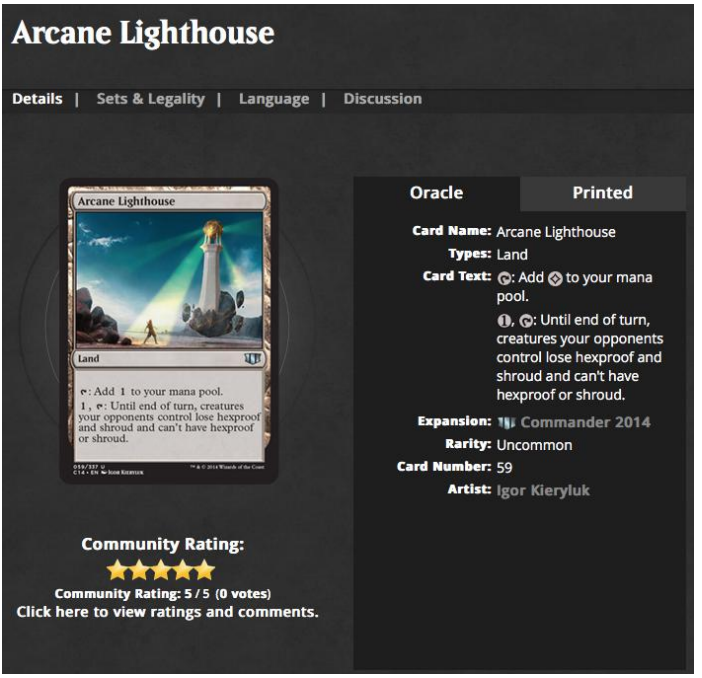

Figure 3-1 http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=389430.

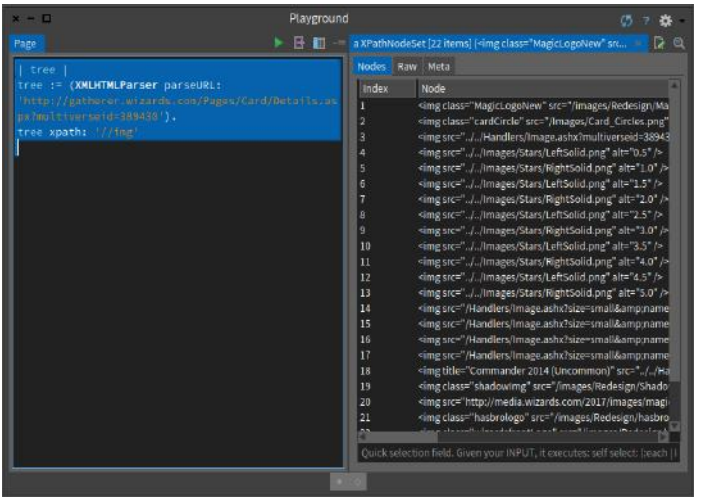

Figure 3-2 Exploring images.

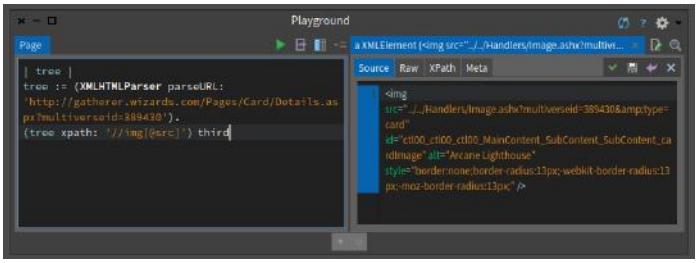

Figure 3-3 Exploring images.

```
| tree |
tree := (XMLHTMLParser parseURL:
    'http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=389430').
tree xpath: '//img'
```

No so cool but working...

Toying with the inspector, we come up with the following ugly expression to get the name of the JPEG

```
| tree |
tree := (XMLHTMLParser parseURL:
    'http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=389430').
((tree xpath: '//img') third @ 'src') first value allButFirst: 5
>>> 'Handlers/Image.ashx?multiverseid=389430&type=card'
```

Ugly isn't it? This happens often when scraping HTML, but we can do better. By the way note also that we start to enter directly XPath command using the XPath pane and using the doit and go facilities of the inspector. This way we do not have to get the page from internet all the time.

3.3 Revisiting it

We could not really show you such ugly expressions so we had to find a better one.

So first we look at the img that has src as atttribute as shown below and in Figure 3-3.

```
| tree |
tree := (XMLHTMLParser parseURL:
    'http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=389430').
(tree xpath: '//img[@src]')
```

Then as shown in Figure 3-4 we inspected the right node.

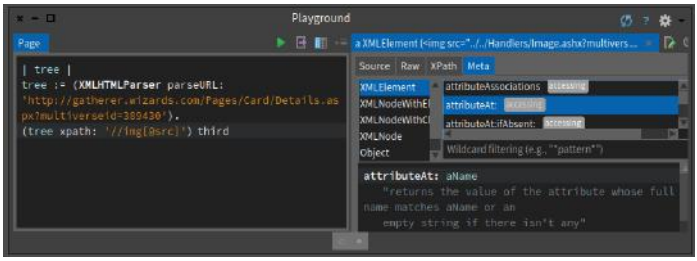

Figure 3-4 Narrowing the node.

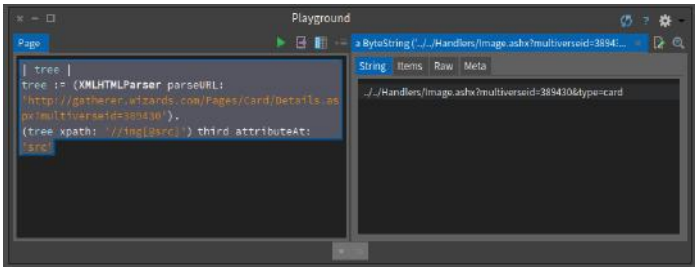

Figure 3-5 Exploring the class API on the spot: looking to see if there is a attribute something method.

Finally since we were on this exact node, we looked in its class to see if we could get an API to get the attribute in a nice way as shown in Figure 3-5.

```
| tree |
tree := (XMLHTMLParser parseURL:
    'http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=389430').
(tree xpath: '//img[@src]') third attributeAt: 'src'
```

Now that we have the visual path, we can use the HTTP client of Pharo to get the image as shown in Figure 3-6.

```
| tree path |
tree := (XMLHTMLParser parseURL:
    'http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=389430').
path := ((tree xpath: '//img[@src]')  third attributeAt: 'src')
    allButFirst: 5.
(ZnEasy getJpeg: 'http://gatherer.wizards.com/',path) asMorph
    openInWorld
```

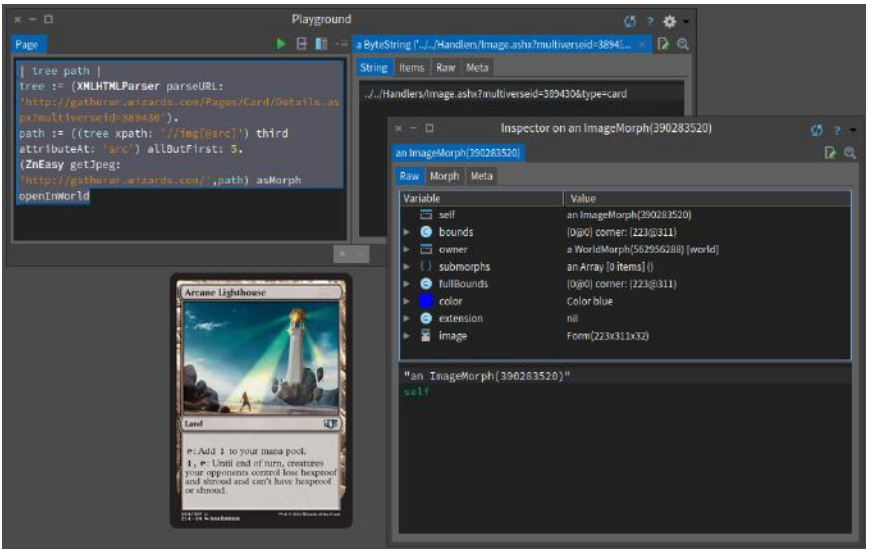

Figure 3-6 Getting the card visual inside Pharo.

3.4 Getting data

Since this web page is probably generated, we look for example for the artist string in the source and we found the following matches:

```
ClientIDs.artistRow =
    'ctl00_ctl00_ctl00_MainContent_SubContent_SubContent_artistRow';
```

This one is more interesting:

```
<div
    id="ctl00_ctl00_ctl00_MainContent_SubContent_SubContent_artistRow"
    class="row">
   <div class="label">
      Artist:</div>
   <div
    id="ctl00_ctl00_ctl00_MainContent_SubContent_SubContent_ArtistCredit"
    class="value">
     <a
    href="/Pages/Search/Default.aspx?action=advanced&amp;artist=[%22Igor
    Kieryluk%22]">Igor Kieryluk</a></div>
```

We can build queries to identify node elements having this id. To avoid to perform an internet request each time, we typed directly XPath path in the XPath pane of the inspector as shown in Figure 3-7. Now trying to get faster we looked at all the class=”row” as shown in Figure 3-7

```
//div[@class='row']
```

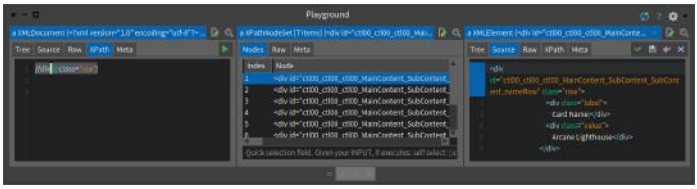

Figure 3-7 Getting the card information.

The following expression returns the pair label and value for example for the card name label and its value.

```
//div[@class='row']/div[@class='label']|
    //div[@class='row']/div[@class='value']
```

So we can now query all the fields

```
| tree |
tree := (XMLHTMLParser parseURL:
    'http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=389430').
container := tree xpath:
    '//div[@class=''row'']/div[@class=''label'']|
    //div[@class=''row'']/div[@class=''value'']'.
container collect: [ :each | each contentString trimBoth  ].
>>> a XMLOrderedList('Card Name:' 'Arcane Lighthouse' 'Types:'
    'Land' 'Card Text:'
': Add  to your mana pool. , : Until end of turn, creatures your
    opponents control
lose hexproof and shroud and can''t have hexproof or shroud.'
'Expansion:' 'Commander 2014' 'Rarity:' 'Uncommon' 'Card Number:'
    '59' 'Artist:' 'Igor Kieryluk')
```

Now we can convert this into a dictionary

```
| tree  |
tree := (XMLHTMLParser parseURL:
    'http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=389430').
container := tree xpath:
    '//div[@class=''row'']/div[@class=''label'']|
    //div[@class=''row'']/div[@class=''value'']'.
((container collect: [ :each | each contentString trimBoth  ])
  asOrderedCollection groupsOf: 2 atATimeCollect: [ :x :y | x -> y])
    asDictionary
```

And convert it into JSON for fun

```
| tree dict |
tree := (XMLHTMLParser parseURL:
    'http://gatherer.wizards.com/Pages/Card/Details.aspx?multiverseid=389430').
container := tree xpath:
    '//div[@class=''row'']/div[@class=''label'']|
```

```
    //div[@class=''row'']/div[@class=''value'']'.
dict := ((container collect: [ :each | each contentString trimBoth
    ])
  asOrderedCollection groupsOf: 2 atATimeCollect: [ :x :y | x -> y])
    asDictionary.

NeoJSONWriter toStringPretty:dict
>>>

'{
  "Card Number:" : "59",
  "Card Name:" : "Arcane Lighthouse",
  "Artist:" : "Igor Kieryluk",
  "Types:" : "Land",
  "Card Text:" : ": Add  to your mana pool. , : Until end of turn,
    creatures your opponents control lose
   hexproof and shroud and can''t have hexproof or shroud.",
  "Expansion:" : "Commander 2014",
  "Rarity:" : "Uncommon"
}'
```

Now we can apply the same technique to access all the cards and also different pages to extract all the card unique id and query the database. But this is left as an exercise.

3.5 Conclusion

We show you how we could access the page and navigate interactively through it using XPath and live programming feature of Pharo. This chapter should show the great value to be able to tweak you live a document and navigate to find the information you really want.