pyquery: 基于python和jquery语法操作XML

pyquery 可以让你用jquery语法来对xml进行查询。这个API和jquery十分类似。如果利用lxml，pyquery对xml和html的操作将更加快速。

这个库并不是（至少还不是）一个可以和javascript互交的代码库。只是很喜欢jquery API并且在使用python的过程中，我真的很怀念jquery，所以我告诉我自己“让我们在python里面也使用jquery吧！”所以就有了这个库。

这个库可以有多种用途，比如我可以在将来用pyquery对纯http模板就行编辑，或者可以和Deliverance配套使用对样式进行操作.

这个项目现在基于mercurial开发，并用Bitbucket发布。我有权给任何想要审查代码的人权利。如果你想对代码进行贡献，给我电邮吧。

欢迎大家访问我的网站geoinformatics.cn.

内容

用法

用户可以使用PyQuery类从字符串、lxml对象、文件或者url来加载xml文档:

```
>>> from pyquery import PyQuery as pq
>>> from lxml import etree
>>> d = pq("<html></html>")
>>> d = pq(etree.fromstring("<html></html>"))
>>> d = pq(url='http://google.com/')
>>> d = pq(filename=path_to_html_file)
```

d在这里诸如jquery里面的$对象:

```
>>> d("#hello")
[<p#hello.hello>]
>>> p = d("#hello")
>>> p.html()
'Hello world !'
>>> p.html("you know <a href='http://python.org/'>Python</a> rocks")
[<p#hello.hello>]
>>> p.html()
'you know <a href="http://python.org/">Python</a> rocks'
>>> p.text()
'you know Python rocks'
```

用户可以使用jquery提供的一些伪类（但还不支持css）来进行操作，诸如:first :last :even :odd :eq :lt :gt :checked :selected :file:

```
>>> d('p:first')
[<p#hello.hello>]
```

Attributes

You can play with the attributes with the jquery API:

```
>>> p.attr("id")
'hello'
>>> p.attr("id", "plop")
[<p#plop.hello>]
>>> p.attr("id", "hello")
[<p#hello.hello>]
```

或者使用更加符合python模式的方法:

```
>>> p.attr.id = "plop"
>>> p.attr.id
'plop'
>>> p.attr["id"] = "ola"
>>> p.attr["id"]
'ola'
>>> p.attr(id='hello', class_='hello2')
[<p#hello.hello2>]
>>> p.attr.class_
'hello2'
>>> p.attr.class_ = 'hello'
```

样式表 (CSS)

你同时也可以对css进行操作:

```
>>> p.addClass("toto")
[<p#hello.toto.hello>]
>>> p.toggleClass("titi toto")
[<p#hello.titi.hello>]
>>> p.removeClass("titi")
[<p#hello.hello>]
```

或者是css模式:

```
>>> p.css("font-size", "15px")
[<p#hello.hello>]
>>> p.attr("style")
'font-size: 15px'
>>> p.css({"font-size": "17px"})
[<p#hello.hello>]
>>> p.attr("style")
'font-size: 17px'
```

或者类似python模式（‘_’ 符号用 ‘-‘代替):

```
>>> p.css.font_size = "16px"
>>> p.attr.style
'font-size: 16px'
>>> p.css['font-size'] = "15px"
>>> p.attr.style
'font-size: 15px'
>>> p.css(font_size="16px")
[<p#hello.hello>]
>>> p.attr.style
'font-size: 16px'
>>> p.css = {"font-size": "17px"}
>>> p.attr.style
'font-size: 17px'
```

转换 (Traversing)

支持大部分jQuwey转换方法。这里是一些实例。

用户可以用字符选择器来进行过滤:

```
>>> d('p').filter('.hello')
[<p#hello.hello>]
```

也可以对单一元素使用 eq 方法:

```
>>> d('p').eq(0)
[<p#hello.hello>]
```

用户也可以寻找内嵌元素:

```
>>> d('p').find('a')
[<a>, <a>]
>>> d('p').eq(1).find('a')
[<a>]
```

```
>>> d('p').find('a').end()
[<p#hello.hello>, <p#test>]
>>> d('p').eq(0).end()
[<p#hello.hello>, <p#test>]
>>> d('p').filter(lambda i: i == 1).end()
[<p#hello.hello>, <p#test>]
```

操作 (Manipulating)

用户可以给标签最好添加内容。:

```
>>> d('p').append('check out <a href="http://reddit.com/r/python"><span>reddit</span></a>')
[<p#hello.hello>, <p#test>]
>>> print d
<html>
...
<p class="hello" id="hello" style="font-size: 17px">you know <a href="http://python.org/">Python
hello <a href="http://python.org">python</a> !
check out <a href="http://python.org/">Python</a> rockscheck out <a href="http://reddit.com/r/p
...
```

或者最前面:

```
>>> p.prepend('check out <a href="http://reddit.com/r/python">reddit</a>')
[<p#hello.hello>]
>>> p.html()
'check out <a href="http://reddit.com/r/python">reddit</a>you know ...'
```

前置或者椎间一个元素到另一个中:

```
>>> p.prependTo(d('#test'))
[<p#hello.hello>]
>>> d('#test').html()
'<p class="hello" ...</p>...hello...python...'
```

插入一个元素:

```
>>> p.insertAfter(d('#test'))
[<p#hello.hello>]
>>> d('#test').html()
```

```
'<a href="http://python.org">python</a> !...'
```

或者在前面:

```
>>> p.insertBefore(d('#test'))
[<p#hello.hello>]
>>> d('body').html()
'\n<p class="hello" id="hello" style="font-size: 17px">...'
```

对每一个元素进行操作:

```
>>> p.each(lambda e: e.addClass('hello2'))
[<p#hello.hello2.hello>]
```

删除元素:

```
>>> d.remove('p#id')
[<html>]
>>> d('p#id')
[]
```

元素替换:

```
>>> p.replaceWith('<p>testing</p>')
[<p#hello.hello2.hello>]
>>> d('p')
[<p>, <p#test>]
```

或者另一种方式:

```
>>> d('<h1>arya stark</h1>').replaceAll('p')
[<h1>]
>>> d('p')
[]
>>> d('h1')
[<h1>, <h1>]
```

删除选择器里面的内容:

```
>>> d('h1').empty()
[<h1>, <h1>]
```

用户也可以获得其中的内容:

```
>>> print d
<html>
<body>
<h1/><h1/></body>
</html>
```

用户可以生成html:

```
>>> from pyquery import PyQuery as pq
>>> print pq('<div>Yeah !</div>').addClass('myclass') + pq('<b>cool</b>')
<div class="myclass">Yeah !</div><b>cool</b>
```

AJAX

用户可以查询wsgi应用，只要安装了WebOb (这并不是pyquery所必需的)。在这个例子中，测试应用将返回输入的对象 / 以及提交的按钮 /submit:

```
>>> d = pq('<form></form>', app=input_app)
>>> d.append(d.get('/'))
[<form>]
>>> print d
<form><input name="youyou" type="text" value=""/></form>
```

在其他节点也可以使用该应用:

```
>>> d.get('/').app is d.app is d('form').app
True
```

用户也可以请求其他的路径:

```
>>> d.append(d.get('/submit'))
[<form>]
>>> print d
<form><input name="youyou" type="text" value=""/><input type="submit" value="OK"/></form>
```

如果安装了 Paste, 你可以使用Proxy直接获得url所对应的应用:

```
>>> a = d.get('https://bitbucket.org/olauzanne/pyquery/')
>>> a
[<html>]
```

用户可以获得其返回值:

```
>>> print a.response.status
301 Moved Permanently
```

返回的属性是一个WebOb 返回值

生成绝对链接

用户可以生成绝对链接，这在抓屏过程中很有效:

```
>>> d = pq(url='http://www.w3.org/', parser='html')
>>> d('a[title="W3C Activities"]').attr('href')
'/Consortium/activities'
>>> d.make_links_absolute()
[<html>]
>>> d('a[title="W3C Activities"]').attr('href')
'http://www.w3.org/Consortium/activities'
```

使用不同的解析器

pyquery默认使用lxml.xml作为解析器，所以如果用户的应用不能使用，则可以尝试用lxml.html进行html解析。xml解析器有时候会有些问题。特别是当处理xhtml页面的时候，因为解析器会触发一个错误当遇到一个没有的xml树时(以 w3c.org为例)。

你也可以选择特定的解析器:

```
>>> pq('<html><body><p>toto</p></body></html>', parser='xml')
[<html>]
>>> pq('<html><body><p>toto</p></body></html>', parser='html')
[<html>]
>>> pq('<html><body><p>toto</p></body></html>', parser='html_fragments')
[<p>]
```

html和html_fragments解析器来自lxml.html。

测试

如果哦你想运行测试，你可以看到以上内容，遵循以下步骤:

```
$ hg clone https://bitbucket.org/olauzanne/pyquery/
$ cd pyquery
$ python bootstrap.py
$ bin/buildout
$ bin/test
```

遵循以下步骤来构建Sphinx文档:

```
$ cd docs
$ make html
```

如果你没有安装lxml，请使用如下命令:

```
$ STATIC_DEPS=true bin/buildout
```

更多文档

First there is the Sphinx documentation 点击此处. 更多关于该API的文档，请使用 jquery 网站.我现在参考书写API来自color cheat sheet. 你随时可以参考 源代码。

TODO

- 选择器 (SELECTORS): 仍然缺少的jQuery伪类 (:radio, :password, ...)
- 属性 (ATTRIBUTES): 完成
- 样式表 (CSS): 完成
- HTML: done
- 操作 (MANIPULATING): 仍然缺少wrapAll和wrapInner方法。
- 转换 (TRAVERSING): 以完成大多半。
- 事件: 还没有和服务器端进行交互，以后可以用ajax
- 核心UI效果: 实现了 hide 和 show。其他和服务器端交互不相干
- AJAX: 和wsgi应用配合使用的部分

pyquery.pyquery - PyQuery 完全 API

class pyquery.pyquery.**PyQuery**(*args, **kwargs)

主要的类

addClass(value)

给元素添加css类:

```
>>> d = PyQuery('<div></div>')
>>> d.addClass('myclass')
```

[<div.myclass>]

after(value)

给节点添加值

append(value)

给每个节点添加值

appendTo(value)

给值添加节点

attr

支持set/get/del 属性，类似javascript样式。

base_url

返回当前html文档的url，如果不存在则返回None。

before(value)

在节点前插入值

clone()

css

支持 set/get/del 属性，类似javascript样式。

each(func)

给每个节点添加func。

empty()

删除节点内容。

end()

退出当前转换并返回上一层。

```
>>> m = '<p><span><em>Whoah!</em></span></p><p><em> there</em></p>'
>>> d = PyQuery(m)
>>> d('p').eq(1).find('em').end().end()
[<p>, <p>]
```

eq(index)

返回被检索的元素。

```
>>> d = PyQuery('<p class="hello">Hi</p><p>Bye</p><div></div>')
>>> d('p').eq(0)
[<p.hello>]
>>> d('p').eq(1)
[<p>]
```

filter(selector)

运用(字符串或者函数)进行过滤.

```
>>> d = PyQuery('<p class="hello">Hi</p><p>Bye</p>')
>>> d('p')
[<p.hello>, <p>]
>>> d('p').filter('.hello')
[<p.hello>]
>>> d('p').filter(lambda i: i == 1)
[<p>]
>>> d('p').filter(lambda i: PyQuery(this).text() == 'Hi')
[<p.hello>]
```

find(selector)

运用选择器来对元素进行查找。

```
>>> m = '<p><span><em>Whoah!</em></span></p><p><em> there</em></p>'
>>> d = PyQuery(m)
>>> d('p').find('em')
```

```
[<em>, <em>]
>>> d('p').eq(1).find('em')
[<em>]
```

hasClass(name)

如果拥有该class则返回True:

```
>>> d = PyQuery('<div class="myclass"></div>')
>>> d.hasClass('myclass')
```

True

height(value=<NoDefault>)

设置或获取元素的高度值

hide()

给display添加none于元素样式

html(value=<NoDefault>)

设置或者获取子节点html的表达方法。

获取文本值:

```
>>> doc = PyQuery('<div><span>toto</span></div>')
>>> print doc.html()
<span>toto</span>
```

设置文本值:

```
>>> doc.html('<span>Youhou !</span>')
[<div>]
>>> print doc
<div><span>Youhou !</span></div>
```

insertAfter(value)

在值后面添加节点

insertBefore(value)

在值前添加节点

is_(selector)

如果选择器符合当前元素则返回True，否则返回False。 >>> d = PyQuery(' Hi Bye ') >>> d('p').eq(0).is_('.hello') True >>> d('p').eq(1).is_('.hello') False

make_links_absolute(base_url=None)

生成绝对链接。

map(func)

当用户完成转换后，返回一个新的PyQuery对象。

func 需要两个参数 - ‘index’ 和 ‘element’ . 在func里面，元素可以用 ‘this’ 代替。

```
>>> d = PyQuery('<p class="hello">Hi there</p><p>Bye</p><br />')
>>> d('p').map(lambda i, e: PyQuery(e).text())
['Hi there', 'Bye']
```

```
>>> d('p').map(lambda i, e: len(PyQuery(this).text()))
[8, 3]
```

```
>>> d('p').map(lambda i, e: PyQuery(this).text().split())
['Hi', 'there', 'Bye']
```

not_(selector)

返回不符合选择器的元素。

```
>>> d = PyQuery('<p class="hello">Hi</p><p>Bye</p><div></div>')
>>> d('p').not_('.hello')
[<p>]
```

prepend(value)

给节点前附加一个值

prependTo(value)

给值前附加一个节点。

remove(expr=<NoDefault>)

删除节点。

removeAttr(name)

删除一个属性:

```
>>> d = PyQuery('<div id="myid"></div>')
>>> d.removeAttr('id')
```

[ ]

removeClass(value)

删除一个元素的css类

```
>>> d = PyQuery('<div class="myclass"></div>')
>>> d.removeClass('myclass')
[<div>]
```

replaceAll(expr)

用表达式替换节点。

replaceWith(value)

用值替换节点。

show()

将元素的display属性更变为block。

text(value=<NoDefault>)

设置或者获得子元素的文本。

获得文本值:

```
>>> doc = PyQuery('<div><span>toto</span><span>tata</span></div>')
>>> print doc.text()
toto tata
```

设置文本值:

```
>>> doc.text('Youhou !')
[<div>]
>>> print doc
<div>Youhou !</div>
```

toggleClass(value)

开关元素的某个css类

```
>>> d = PyQuery('<div></div>')
>>> d.toggleClass('myclass')
[<div.myclass>]
```

val(value=<NoDefault>)

设置或者获得元素值:

```
>>> d = PyQuery('<input />')
>>> d.val('Youhou')
```

[<input>] >>> d.val() ‘Youhou’

width(value=<NoDefault>)

set/get width of element

wrap(value)

每个对象将实时添加HTML字符串:

```
>>> d = PyQuery('<span>youhou</span>')
```

```
>>> d.wrap('<div></div>')
[<div>]
>>> print d
<div><span>youhou</span></div>
```

wrapAll(value)

将所有的元素包装到一个元素中:

```
>>> d = PyQuery('<div><span>Hey</span><span>you !</span></div>')
>>> print d('span').wrapAll('<div id="wrapper"></div>')
```

Heyyou !

pyquery.ajax – PyQuery AJAX 扩展

class pyquery.ajax.**PyQuery**(*args, **kwargs)

get(path_info, **kwargs)

获取wsgi应用或者url的路径。

post(path_info, **kwargs)

从wsgi应用或者url来POST某个路径。