User:Pavel Dusek/Sandbox

Under construction / Forgotten

This article was marked by its author as *Under construction*, but the last edit is older than 30 days. If you want to edit this page, please try to contact its author first (you fill find him in the history (https://www.wikilectures.eu/index.php?title=Pavel_Dusek/Sandbox&action=history)). Watch the as well. If the author will not continue in work, remove the template `{{Under construction}}` and the page.

Last update: Sunday, 08 Jan 2012 at 2.39 pm.

Captcha edit test Captcha edit test

MediaWiki:Dump.py

Dump.py is a simple script written in python and run every day on the server. It extracts data from the database and creates an xml dump file located on the url: http://www.wikilectures.eu/extensions/Dumps/dump.xml (2 MB). The dump file may be used as a source file for an user friendly bot with GUI called [[name needed (OmniBot?)]] that is being created with the purpose of general usage among all members of the WikiLectures team.

The dump file contains latest versions of all pages in the main namespace and the talk namespace. The format of the file is according to Media Wiki xml dump format.

For the data extraction, the script uses the export special page.

Source code (v. 1.0)

The script may be run from the command line by typing:

```
python dump.py --file /path/to/dumpfilename.xml --url "http://www.wikilectures.eu/"
```

```
#coding=utf-8
#!/usr/bin/python
import xml.dom.minidom
import urllib, urllib2
from optparse import OptionParser
debug = False

class Api:
    #apiurl = "http://www.wikiskripta.eu/api.php"
    apiurl = "http://www.wikilectures.eu/api.php"
    url = ""
    changedPages = []
    nonId = ['0', '1']
    mainNamespaceIds = []
    mainNamespacePages = []
    talkNamespacePages = []

    def __init__(self, url, pocetHodin):
        self.apiurl = url + "api.php"
        self.url = url
        self.mainNamespacePages = self.allpages(0)
        self.talkNamespacePages = self.allpages(1)
    def request(self, values, tagname):
        """Provede žádost API o stránku. Je nutné, aby žádost byla koncipována tak, že vrací informace pouze v jednom tagu. Funkce frací informace v podobě dictionary, kam přidá i token pro pokračování (querycontinue)."""
        data = urllib.urlencode(values)
        request = urllib2.Request(self.apiurl, data)
        response = urllib2.urlopen(request)
        xmlText = response.read()
        dom = xml.dom.minidom.parseString(xmlText)
        if (debug): print dom.toprettyxml()
        returnValue = {}
        try:
            tag = dom.getElementsByTagName(tagname)[0]
            for key in tag.attributes.keys():
                returnValue[key] = tag.getAttribute(key)
            if (tag.firstChild):
                returnValue[u'text'] = tag.firstChild.nodeValue
        except Exception:
            print "Could not get DOM node:", tagname

        try:
            if (dom.getElementsByTagName("query-continue")):
                querycontinueNode = dom.getElementsByTagName("query-continue")[0].childNodes[0]
                for key in querycontinueNode.attributes.keys():
                    returnValue[key] = querycontinueNode.getAttribute(key)
        except Exception:
            print "Could not get DOM node: query-continue"
        return returnValue
```

```
    def requestList(self, values, tagname, attribute):
        """Provede žádost API o stránku. Je nutné, aby žádost byla koncipována tak, že vrací informace jsou v různých tazích, ale všechny tagy tohoto názvu mají stejnou strukturu atributů. Funkce vrací informace atributu v podobě list."""
        data = urllib.urlencode(values)
        request = urllib2.Request(self.apiurl, data)
        response = urllib2.urlopen(request)
        xmlText = response.read()
        dom = xml.dom.minidom.parseString(xmlText)
        if (debug): print dom.toprettyxml()
        returnValue = []
        try:
            tags = dom.getElementsByTagName(tagname)
            for tag in tags:
                if (attribute):
                    returnValue.append(tag.getAttribute(attribute))
                else:
                    returnValue.append(tag.childNodes[0].toxml())
        except Exception:
            print "Error: ", e
            print "Could not get DOM node:", tagname
        return returnValue

    def getText(self, pageid):
        values = {'action': 'query', 'prop': 'revisions', 'pageids': pageid, 'rvprop': 'content', 'format': 'xml' }
        request = self.request(values, 'rev')
        if (request != {} and u'text' in request.keys()):
            return request[u'text']
        else:
            return False

    def getExternalLinks(self, page):
        """Získá externí odkazy ze článku. Vrací jako list."""
        values = {'action': 'query', 'prop': 'extlinks', 'titles': page, 'format': 'xml'}
        request = self.requestList(values, "el", None)
        if (debug): print request
        return request

    def allpages(self, namespace):
        """Funkce vrátí seznam všech článků (id) ve jmenném prostoru namespace."""
        pages = []
        values = {'action': 'query', 'list': 'allpages', 'apnamespace': str(namespace), 'apfilterredir': 'nonredirects', 'aplimit': '1', 'format': 'xml'}
        request = self.request(values, 'p')
        #pages.append(request[u'pageid'])
        pages.append(request[u'title'])
        while (u'apfrom' in request.keys()):
            values = {
                'action': 'query',
                'list': 'allpages',
                'apnamespace': str(namespace),
                'apfilterredir': 'nonredirects',
                'aplimit': '1',
                'apfrom': request[u'apfrom'].encode("utf-8"),
                'format': 'xml'
            }
            request = self.request(values, 'p')
            self.mainNamespaceIds.append(request[u'pageid'])
            pages.append(request[u'title'])
            if (debug): print request[u'title'],",",request[u'pageid']
        return pages

    def recentChanges(self, pocetHodin):
        """Funkce vrátí seznam článků (id), u nichž doslo za posledních `pocetHodin` hodin ke změně. Vybírají se pouze editace z hlavního jmenného prostoru."""
        recentchanges = []
        #rcstart = time.strftime("%Y-%m-%dT%H:%M:%SZ", time.gmtime(time.time() - 12 * 3600))
        rcstart = time.strftime("%Y-%m-%dT%H:%M:%SZ", time.gmtime())
        rcend = time.strftime("%Y-%m-%dT%H:%M:%SZ", time.gmtime(time.time() - 12 * 3600 - pocetHodin * 3600))
        values = {'action': 'query', 'list': 'recentchanges', 'rcprop': 'title|ids|timestamp', 'rclimit': 1, 'rcstart': rcstart, 'rcend': rcend, 'format': 'xml'}
        request = self.request(values, "rc")
        while (u'rcstart' in request.keys()):
            values['rcstart'] = request[u'rcstart']
            request = self.request(values, "rc")
            if (not u":" in request[u'title'] and not request[u'pageid'] in recentchanges):
                recentchanges.append(request[u'pageid'])
        self.changedPages = recentchanges
    def getRevisionSummaries(self, pageTitle):
        """Funkce vrátí seznam shrnutí všech editací daného článku."""
        request = []
        rvstart = time.strftime("%Y-%m-%dT%H:%M:%SZ", time.gmtime())
        try:
            values = {'action': 'query', 'prop': 'revisions', 'titles': pageTitle.encode("utf-8"), 'rvprop': 'timestamp|user|comment', 'rvstart': rvstart, 'rvlimit': '500', 'format': 'xml'}
            request = self.requestList(values, 'rev', 'comment')
        except Exception:
            pass
        print request
        return request
    def getDumpXML(self, pages):
        exportURL = self.url + "index.php?title=Special:Export&action=submit"
        values = {'curonly': 1, 'pages': "\n".join([fromUnicode(page) for page in pages])}

        data = urllib.urlencode(values)
        request = urllib2.Request(exportURL, data)
        response = urllib2.urlopen(request)
        xmlText = response.read()
        return xmlText


def fromUnicode(unicodeString):
    returnString = ""
    citliveZnaky = { 382: 'ž', 269: 'č', 283: 'ě', 237: 'í', 352: 'Š', 225: 'á', 345: 'ř', 353: 'š', 253: 'ý', 367: 'ů', 233: 'é', 381: 'Ž', 268: 'Č', 218: 'Ú', 250: 'ú', 357: 'ť', 271: 'ď', 328: 'ň', 243: 'ó', 8230: '…', 8222: '„', 8220: '“', 8722: '−', 318: 'ľ', 270: 'Ď', 244: 'ô', 154: '', 8211: '–', 327: 'Ň', 205: 'Í', 183: '·', 215: '×', 344: 'Ř', 9742: '☎', 9997: '✍', 322: 'ł', 232: 'è', 221: 'Ý', 8212: '—', 160: ' ', 167: '§', 61474: '', 252: 'ü', 177: '±', 945: 'α', 228: 'ä', 960: 'π', 246: 'ö', 946: 'β', 176: '°', 346: 'Ś', 282: 'Ě', 193: 'Á', 352: 'Š', 366: 'Ů', 180: '´', 8217: '’', 231: 'ç', 224: 'à', 201: 'É', 314: 'Ĺ', 8218: ',', 8219: '‛', 914: 'Β' }
    czKeys = citliveZnaky.keys()
    for char in unicodeString:
```

```
        if (ord(char) in czKeys):
            returnString = returnString + citliveZnaky[ord(char)]
        else:
            returnString = returnString + str(char)
    return returnString
if __name__ == '__main__':
    parser = OptionParser()
    parser.add_option("-d", "--debug",
                      action="store_true", dest="debug", default=False, help="print status messages to stdout for debug")
    parser.add_option("-f", "--file",
                      action="store", dest="file", default="dump.xml", help="name of the file where the dump shall be stored, default is
\"dump.xml\"")
    parser.add_option("-u", "--url",
                      action="store", dest="url", default="http://www.wikiskripta.eu/", help="url of the project (with slash)")
    (options, args) = parser.parse_args()
    debug = options.debug
    extlinks = []
    foo = Api(options.url, 24)
    dump = foo.getDumpXML(foo.mainNamespacePages + foo.talkNamespacePages)
    if (options.file):
        f = open(options.file, "w")
    else:
        f = open("dump.xml", "w")
    f.write(dump)
    f.close()
```