```
**********************************************************
    3153 Mon Apr  8 19:03:28 2013
new/usr/src/tools/onbld/Checks/DbLookups.py
3674 onbld Checks should not query opensolaris.org
**********************************************************
```

```
#! /usr/bin/python
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
#

# Copyright 2010, Richard Lowe

#
# Various database lookup classes/methods, i.e.:
#     * monaco
#     * bugs.opensolaris.org (b.o.o.)
#     * redmine (illumos.org)
#

import htmllib
import re
import urllib
import urllib2

try:                            # Python >= 2.5
        from xml.etree import ElementTree
except ImportError:
        from elementtree import ElementTree

class NonExistentBug(Exception):
        def __str__(self):
                return "Bug %s does not exist" % (Exception.__str__(self))

class BugDBException(Exception):
        def __str__(self):
                return "Unknown bug database: %s" % (Exception.__str__(self))

class BugDB(object):
        """Lookup change requests.

        Usage:
        bdb = BugDB()
        r = bdb.lookup("6455550")
        print r["6455550"]["synopsis"]
        r = bdb.lookup(["6455550", "6505625"])
```



```
        print r["6505625"]["synopsis"]
        """

        VALID_DBS = ["illumos"]
        VALID_DBS = ["bugster", "illumos"]

        def __init__(self, priority = ["illumos"]):
        def __init__(self, priority = ("illumos", "bugster")):
                """Create a BugDB object.

                Keyword argument:
                priority: use bug databases in this order
                """
                for database in priority:
                        if database not in self.VALID_DBS:
                                raise BugDBException, database
                self.__priority = priority


        def __illbug(self, cr):
                url = "http://illumos.org/issues/%s.xml" % cr
                req = urllib2.Request(url)

                try:
                        data = urllib2.urlopen(req)
                except urllib2.HTTPError, e:
                        if e.code == 404:
                                raise NonExistentBug(cr)
                        else:
                                raise

                bug = ElementTree.parse(data)

                return {'cr_number': bug.find('id').text,
                        'synopsis': bug.find('subject').text,
                        'status': bug.find('status').attrib['name']
                }


        def __boobug(self, cr):
                cr = str(cr)
                url = "http://bugs.opensolaris.org/view_bug.do"
                req = urllib2.Request(url, urllib.urlencode({"bug_id": cr}))
                results = {}
                try:
                        data = urllib2.urlopen(req).readlines()
                except urllib2.HTTPError, e:
                        if e.code != 404:
                                print "ERROR: HTTP error at " + \
                                        req.get_full_url() + \
                                        " got error: " + str(e.code)
                                raise e
                        else:
                                raise NonExistentBug(cr)
                except urllib2.URLError, e:
                        print "ERROR: could not connect to " + \
                                req.get_full_url() + \
                                ' got error: "' + e.reason[1] + '"'
                        raise e
                htmlParser = htmllib.HTMLParser(None)
                metaHtmlRe = re.compile(r'^<meta name="([^"]+)" content="([^"]*)
                for line in data:
                        m = metaHtmlRe.search(line)
                        if not m:
                                continue
                        val = urllib.unquote(m.group(2))
```

```
126                         htmlParser.save_bgn()
127                         htmlParser.feed(val)
128                         results[m.group(1)] = htmlParser.save_end()
129                 htmlParser.close()

131                 if "synopsis" not in results:
132                         raise NonExistentBug(cr)

134                 results["cr_number"] = cr
135                 results["sub_category"] = results.pop("subcategory")
136                 results["status"] = results.pop("state")
137                 results["date_submitted"] = results.pop("submit_date")

139                 return results

 99         def lookup(self, crs):
100                 """Return all info for requested change reports.

102                 Argument:
103                 crs: one change request id (may be integer, string, or list),
104                      or multiple change request ids (must be a list)

106                 Returns:
107                 Dictionary, mapping CR=>dictionary, where the nested dictionary
108                 is a mapping of field=>value
109                 """
110                 results = {}
111                 if not isinstance(crs, list):
112                         crs = [str(crs)]
113                 for database in self.__priority:
114                         if database == "illumos":
156                         if database == "bugster":
115                                 for cr in crs:
158                                         cr = str(cr)
116                                         try:
160                                                 results[cr] = self.__boobug(cr)
161                                         except NonExistentBug:
162                                                 continue
163                         elif database == "illumos":
164                                 for cr in crs:
165                                         try:
117                                                 results[str(cr)] = self.__illbug
118                                         except NonExistentBug:
119                                                 continue

121                         # the CR has already been found by one bug database
122                         # so don't bother looking it up in the others
123                         for cr in crs:
124                                 if cr in results:
125                                         crs.remove(cr)

127                 return results
```