Bugzilla Documentation

Release 0.2

David Burns

December 05, 2014

Contents

Bugsy is a tool that allows you to programmatically work with Bugzilla using its native REST API.

To use you will do

```
import bugsy
bugzilla = bugsy.Bugsy()
bug = bugzilla.get(123456)
bug123456.status = 'RESOLVED'
bug123456.resolution = 'FIXED'
bugzilla.put(bug123456)
```

CHAPTER 1

Installing Bugsy

To install Bugsy, simply use pip or easy install

Pip

```
pip install bugsy
```

easy_install

```
easy_install bugsy
```

CHAPTER 2

Using Bugsy

2.1 Getting a bug from Bugzilla

Getting a bug is quite simple. Create a Bugsy object and then get the bug number that you want.

```
import bugsy
bugzilla = bugsy.Bugsy()
bug = bugzilla.get(123456)
```

2.2 Creating a new bug

To create a new bug, create a Bug object, populate it with the items that you need and then use the Bugsy object to put the bug into Bugzilla

```
import bugsy
bug = bugsy.Bug()
bug.summary = "I really realy love cheese"
bug.add_comment("and I really want sausages with it!")

bugzilla = bugsy.Bugsy("username", "password")
bugzilla.put(bug)
bug.id #returns the bug id from Bugzilla
```

2.3 Searching Bugzilla

To search for bugs you will need to create a `Bugsy` object and then you can call *search_for* and chain the search. The `Search` API is a Fluent API o you just chain the items that you need and then call *search* when the search is complete.

```
import bugsy
bugzilla = bugsy.Bugsy()
bugs = bugzilla.search_for\
                .keywords("checkin-needed")\
                .include_fields("flags")\
                .search()
```

More details can be found in from the `Search` class

2.4 Comments

Getting comments from a bug

```
import bugsy
bugzilla = bugsy.Bugsy()
bug = bugzilla.get(123456)
comments = bug.get_comments()
comments[0].text # Returns  "I <3 Sausages"
```

Adding comments to a bug

```
import bugsy
bugzilla = bugsy.Bugsy()
bug = bugzilla.get(123456)
bug.add_comment("And I love bacon too!")
```

To see further details look at:

2.4.1 Bugsy

class bugsy.**Bugsy**(*username=None*, *password=None*, *bugzilla_url='https://bugzilla.mozilla.org/rest'*)
Bugsy allows easy getting and putting of Bugzilla bugs

get(*bug_number*)
Get a bug from Bugzilla. If there is a login token created during object initialisation it will be part of the query string passed to Bugzilla

Parameters **bug_number** – Bug Number that will be searched. If found will return a Bug object.

```
>>> bugzilla = Bugsy()
>>> bug = bugzilla.get(123456)
```

put(*bug*)
This method allows you to create or update a bug on Bugzilla. You will have had to pass in a valid username and password to the object initialisation and recieved back a token.

Parameters **bug** – A Bug object either created by hand or by using get()

If there is no valid token then a BugsyException will be raised. If the object passed in is not a Bug then a BugsyException will be raised.

```
>>> bugzilla = Bugsy()
>>> bug = bugzilla.get(123456)
>>> bug.summary = "I like cheese and sausages"
>>> bugzilla.put(bug)
```

request(*path*, *method='GET'*, ***kwargs*)
Perform a HTTP request.

Given a relative Bugzilla URL path, an optional request method, and arguments suitable for requests.Request(), perform a HTTP request.

class bugsy.**BugsyException**(*msg*)
If while interacting with Bugzilla and we try do something that is not supported this error will be raised.

class bugsy.**LoginException**(*msg*)
If a username and password are passed in but we don't receive a token then this error will be raised.

2.4.2 Bug

class bugsy.**Bug**(*bugsy=None*, ***kwargs*)

This represents a Bugzilla Bug

OS

Property for getting or setting the OS that the bug occured on

```
>>> bug.OS
"All"
```

add_comment(*comment*)

Adds a comment to a bug. If a bug does not have a bug ID then you need call *put* on the Bugsy class.

```
>>> bug.add_comment("I like sausages")
>>> bugzilla.put(bug)
```

If it does have a bug id then this will do a post to the server

```
>>> bug.add_comment("I like eggs too")
```

component

Property for getting the bug component

```
>>> bug.component
General
```

get_comments()

Obtain comments for this bug.

Returns a list of Comment instances.

id

Property for getting the ID of a bug.

```
>>> bug.id
123456
```

platform

Property for getting the bug platform

```
>>> bug.platform
"ARM"
```

product

Property for getting the bug product

```
>>> bug.product
Core
```

resolution

Property for getting or setting the bug resolution

```
>>> bug.resolution
"FIXED"
```

status

Property for getting or setting the bug status

```
>>> bug.status
"REOPENED"
```

summary
Property for getting and setting the bug summary

```
>>> bug.summary
"I like cheese"
```

to_dict()
Return the raw dict that is used inside this object

update()
Update this object with the latest changes from Bugzilla

```
>>> bug.status
'NEW'
#Changes happen on Bugzilla
>>> bug.update()
>>> bug.status
'FIXED'
```

version
Property for getting the bug platform

```
>>> bug.version
"TRUNK"
```

class bugsy.**BugException**(*msg*)
If we try do something that is not allowed to a bug then this error is raised

2.4.3 Comment

Changed in version 0.3.

class bugsy.**Comment**(***kwargs*)
Represents a single Bugzilla comment.

2.4.4 Search

Changed in version 0.2.

class bugsy.**Search**(*bugsy*)
This allows searching for bugs in Bugzilla

assigned_to(**args*)
When search() is called it will search for bugs assigned to these users

Parameters **args** – items passed in will be turned into a list

Returns Search

```
>>> bugzilla.search_for.assigned_to("dburns@mozilla.com")
```

bug_number(*bug_numbers*)
When you want to search for a bugs and be able to change the fields returned.

Parameters **bug_numbers** – A string for the bug number or a list of strings

Returns Search

```
>>> bugzilla.search_for.bug_number(['123123', '123456'])
```

change_history_fields(*fields*, *value=None*)

include_fields(**args*)

Include fields is the fields that you want to be returned when searching. These are in addition to the fields that are always included below.

Parameters **args** – items passed in will be turned into a list

Returns Search

```
>>> bugzilla.search_for.include_fields("flags")
```

The following fields are always included in search: 'version', 'id', 'summary', 'status', 'op_sys', 'resolution', 'product', 'component', 'platform'

keywords(**args*)

When search() is called it will search for the keywords passed in here

Parameters **args** – items passed in will be turned into a list

Returns Search

```
>>> bugzilla.search_for.keywords("checkin-needed")
```

search()

Call the Bugzilla endpoint that will do the search. It will take the information used in other methods on the Search object and build up the query string. If no bugs are found then an empty list is returned.

```
>>> bugs = bugzilla.search_for\
...                 .keywords("checkin-needed")\
...                 .include_fields("flags")\
...                 .search()
```

summary(**args*)

When search is called it will search for bugs with the words passed into the methods

Parameters **args** – items passed in will be turned into a list

Returns Search

```
>>> bugzilla.search_for.summary("663399")
```

timeframe(*start*, *end*)

When you want to search bugs for a certain time frame.

Parameters

- **start** –
- **end** –

Returns Search

whiteboard(**args*)

When search is called it will search for bugs with the words passed into the methods

Parameters **args** – items passed in will be turned into a list

Returns Search

```
>>> bugzilla.search_for.whiteboard("affects")
```

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*

b