Bugzilla Documentation

Release 0.2

David Burns

July 05, 2014

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

To use you will do

```python
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

To see further details look at:

2.3.1 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)
```

search_for
The search_for property.

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.

2.3.2 Bug

class bugsy.**Bug**(*bugzilla_url=None*, *token=None*, ***kwargs*)
This represents a Bugzilla Bug

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

```
>>> bug.OS
"All"
>>> bug.OS = "Linux"
```

add_comment(*comment*)
Adds a comment to a bug. Once you have added it you will need to call put on the Bugsy object

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

component
Property for getting the bug component

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

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

status

Property for getting or setting the bug status

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

summary

Property for getting and setting the bug summary

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

2.3.3 `Search`

Changed in version 0.2.

class `bugsy.`**`Search`**`(`*bugzilla_url*, *token*`)`
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")
```

`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")
```

`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*

Python Module Index