
Robot Framework WAVE-library Documentation

Release 0.1.1

asko.soukka@iki.fi

Aug 01, 2017

1	Usage	1
2	Keywords	3
2.1	Open WAVE browser	3
2.2	Check URL for accessibility errors	3
2.3	Check accessibility errors	3
2.4	Log WAVE errors	3
2.5	Get WAVE errors	4
2.6	Show WAVE errors, features and alerts	4
2.7	Hide WAVE errors, features and alerts	4
2.8	Capture WAVE errors	4
2.9	Tag WAVE errors	4
2.10	Capture WAVE error	4
2.11	Crop WAVE error	4
2.12	Capture and crop WAVE error	5
3	Source	7

CHAPTER 1

Usage

Include keywords with:

<code>Library WAVELibrary</code>

Note: Currently, RIDE is unable to find keywords provided by this library when this library is imported with `Library WAVELibrary`. This can be fixed by requiring the library with `Resource WAVELibrary/keywords.robot`. (Currently all keywords are written as user keywords, but later they may be refactored into Python-keywords. If this happens, there will be backwards compatible wrappers available at `keywords.robot`.)

Open WAVE browser

Open Firefox with WAVE-toolbar extension installed.

```
Open browser  about:  browser=firefox  ff_profile_dir=${FF_PROFILE_DIR}
```

Check URL for accessibility errors

Open the given URL and check it for accessibility errors.

```
Go to  ${URL}  
Check accessibility errors
```

Check accessibility errors

Check the current page for accessibility errors

```
Show WAVE errors, features and alerts  
${errors} =  Get WAVE errors  
${found} =  Convert to boolean  ${errors}  
Run keyword if  ${found}  Log WAVE errors  ${errors}  
${url} =  Get location  
Should be equal  ${errors}  ${EMPTY}  Wave reported errors for ${url}  
Hide WAVE errors, features and alerts
```

Log WAVE errors

Tag the current test with *Accessibility issues*-tag, try to take screenshots of each given accessibility issue and append the errors into the current test log.

```
Set tags  Accessibility issues  
Capture page screenshot  
Capture WAVE errors  
Log  ${errors}  level=ERROR
```

Get WAVE errors

Extract and return the found WAVE Toolbar errors from the currently open page.

```

${source} = Get source
${source} = Replace string  ${source}  \n  ${EMPTY}
${source} = Replace string  ${source}  "  \n
${source} = Get lines matching regexp  ${source}  ^ERROR:.*
```

Show WAVE errors, features and alerts

Activate WAVE Toolbar's *Show WAVE errors, features and alerts* action.

```
Execute Javascript  return (function(){ window.wave_viewIcons(); return true; })();
```

Hide WAVE errors, features and alerts

Disable WAVE Toolbar's *Show WAVE errors, features and alerts* action.

```
Execute Javascript  return (function(){ window.wave_viewReset(); return true; })();
```

Capture WAVE errors

Try to take a screen capture of each currently visible WAVE toolbar reported error.

```

@{ids} = Tag WAVE errors
${keyword} = Register keyword to run on failure  No operation
: FOR  ${id}  IN  @{ids}
\   Run keyword and ignore error  Capture WAVE error  ${id}
Register keyword to run on failure  ${keyword}
```

Tag WAVE errors

Tag each WAVE toolbar reported error with a unique id and return the ids to ease access to the errors.

```

${errors} = Execute Javascript  return (function(){ var i, id, ids = [], errors = Array.fil
```

Capture WAVE error

Try to take a screen capture of a currently visible WAVE toolbar reported error tagged with the given id.

```

Element should be visible  id=${ID}
Mouse over  ${id}
Element should be visible  css=.wave4tooltip
Capture and crop WAVE error  ${id}.png  ${id}
```

Crop WAVE error

Crop the captured WAVE Toolbar error image saved with the given filename using the bounding box of the given element ids.


```
${ids} = Convert to string  ${ids}
${ids} = Replace string using regexp  ${ids}  u'  '
@{dimensions} = Execute Javascript  return (function(){ var ids = ${ids}, i, target, box, s
Crop WAVE error image  ${OUTPUT_DIR}  ${filename}  @{dimensions}
```

Capture and crop WAVE error

Capture and crop WAVE toolbar error to the given filename using the bounding box of the given element ids.

```
Capture page screenshot  ${filename}
Crop WAVE error  ${filename}  @{locators}
```


CHAPTER 3

Source

*** Settings ***

Library Selenium2Library

Library String

Library WAVELibrary.Cropping

*** Variables ***

\${FF_PROFILE_DIR} \${CURDIR}/profile

\${ERROR_CROP_MARGIN} 50

*** Keywords ***

Open WAVE browser

[Documentation] Open Firefox with WAVE-toolbar extension installed.

Open browser about: browser=firefox ff_profile_dir=\${FF_PROFILE_DIR}

Check URL for accessibility errors

[Documentation] Open the given URL and check it for accessibility errors.

[Arguments] \${URL}

Go to \${URL}

Check accessibility errors

Check accessibility errors

[Documentation] Check the current page for accessibility errors

Show WAVE errors, features and alerts

\${errors} = Get WAVE errors

\${found} = Convert to boolean \${errors}

Run keyword if \${found} Log WAVE errors \${errors}

\${url} = Get location

Should be equal \${errors} \${EMPTY} Wave reported errors for \${url}

Hide WAVE errors, features and alerts

Log WAVE errors

[Documentation] Tag the current test with *Accessibility issues*-tag,

... try to take screenshots of each given accessibility issue

... and append the errors into the current test log.

[Arguments] \${errors}

Set tags Accessibility issues

```
Capture page screenshot
Capture WAVE errors
Log ${errors}    level=ERROR
```

Get WAVE errors

```
[Documentation]  Extract and return the found WAVE Toolbar errors from the
...              currently open page.
${source} =    Get source
${source} =    Replace string  ${source}  \n  ${EMPTY}
${source} =    Replace string  ${source}  "  \n
${source} =    Get lines matching regexp  ${source}  ^ERROR:.*
[return]  ${source}
```

Show WAVE errors, features and alerts

```
[Documentation]  Activate WAVE Toolbar's *Show WAVE errors, features and alerts*
...              action.
Execute Javascript
...      return (function(){ window.wave_viewIcons(); return true; })();
```

Hide WAVE errors, features and alerts

```
[Documentation]  Disable WAVE Toolbar's *Show WAVE errors, features and alerts*
...              action.
Execute Javascript
...      return (function(){ window.wave_viewReset(); return true; })();
```

Capture WAVE errors

```
[Documentation]  Try to take a screen capture of each currently visible
...              WAVE toolbar reported error.
@{ids} =    Tag WAVE errors
${keyword} =    Register keyword to run on failure  No operation
:FOR  ${id}  IN  @{ids}
\    Run keyword and ignore error  Capture WAVE error  ${id}
Register keyword to run on failure  ${keyword}
```

Tag WAVE errors

```
[Documentation]  Tag each WAVE toolbar reported error with a unique id
...              and return the ids to ease access to the errors.
${errors} =    Execute Javascript
...      return (function(){
...          var i, id, ids = [], errors = Array.filter(
...              document.getElementsByClassName("wave4tip"),
...              function(el) { return el.alt.match(/^ERROR.*/) !== null; }
...          );
...          for (i=0; i < errors.length; i++) {
...              id = 'wave-error-' + (new Date().getTime()).toString();
...              id = id + i.toString();
...              errors[i].id = id;
...              ids.push(id);
...          }
...          return ids;
...      })();
[Return]  ${errors}
```

Capture WAVE error

```
[Documentation]  Try to take a screen capture of a currently visible
...              WAVE toolbar reported error tagged with the given id.
[Arguments]  ${id}
Element should be visible  id=${ID}
Mouse over  ${id}
Element should be visible  css=.wave4tooltip
Capture and crop WAVE error  ${id}.png  ${id}
```

Crop WAVE error

```
[Documentation] Crop the captured WAVE Toolbar error image saved
...             with the given filename using the bounding box of the
...             given element ids.
[Arguments]  ${filename}  @{ids}
${ids} = Convert to string  ${ids}
${ids} = Replace string using regexp  ${ids}  u' '
@{dimensions} = Execute Javascript
...     return (function(){
...         var ids = ${ids}, i, target, box, style, offset={};
...         var left = null, top = null, width = null, height = null;
...         for (i = 0; i <= ids.length; i++) {
...             if (i < ids.length) {
...                 target = window.document.getElementById(ids[i]);
...             } else {
...                 target = window.document.getElementsByClassName(
...                     'wave4tooltip')[0];
...             }
...             box = target.getBoundingClientRect();
...             offset.left = Math.round(box.left + window.pageXOffset);
...             offset.top = Math.round(box.top + window.pageYOffset);
...             if (left === null || width === null) {
...                 width = box.width;
...             } else {
...                 width = Math.max(
...                     left + width, offset.left + box.width
...                 ) - Math.min(left, offset.left);
...             }
...             if (top === null || height === null) {
...                 height = box.height;
...             } else {
...                 height = Math.max(
...                     top + height, offset.top + box.height
...                 ) - Math.min(top, offset.top);
...             }
...             if (left === null) { left = offset.left; }
...             else { left = Math.min(left, offset.left); }
...             if (top === null) { top = offset.top; }
...             else { top = Math.min(top, offset.top); }
...         }
...         return [Math.max(0, left - ${ERROR_CROP_MARGIN}),
...                 Math.max(0, top - ${ERROR_CROP_MARGIN}),
...                 Math.max(0, width + ${ERROR_CROP_MARGIN} * 2),
...                 Math.max(height + ${ERROR_CROP_MARGIN} * 2)];
...     })();
Crop WAVE error image  ${OUTPUT_DIR}  ${filename}  @{dimensions}
```

Capture and crop WAVE error

```
[Documentation] Capture and crop WAVE toolbar error to the given
...             filename using the bounding box of the given element ids.
[Arguments]  ${filename}  @{locators}
Capture page screenshot  ${filename}
Crop WAVE error  ${filename}  @{locators}
```