Workday Documentation

Release 0.4.0

Anthony Shaw

Dec 31, 2020

Contents

The Workday python package is for connecting to and leveraging the Workday Web Services from Python 2 or 3.

This project is not part of Workday or operated by Workday, it is an open-source package for consuming their API.

CHAPTER 1

Installation

At the command line:

```
$ pip install workday
```

CHAPTER 2

Usage

In this simple example, a client is instantiated with the endpoint to an API called 'talent'.

The *Get_Languages* method is called on that API and the data return is printed on the screen.

```
import workday
from workday.auth import WsSecurityCredentialAuthentication

client = workday.WorkdayClient(
    wsdls={'talent': 'https://workday.com/tenant/434$sd.xml'},
    authentication=WsSecurityCredentialAuthentication('user', 'password'),
    )

print(client.talent.Get_Languages().data)
```

CHAPTER 3

API Reference

3.1 Client object

The WorkdayClient is used to connect to the API and it offers

class workday.**WorkdayClient**(*wsdls*, *authentication*, *proxy_url=None*, *disable_ssl_verification=False*)

Entry point for the workday APIs.

CHAPTER 4

Types

4.1 workday.soap.WorkdayResponse class

class workday.soap.**WorkdayResponse**(*response*, *service*, *method*, *called_args*, *called_kwargs*)

Bases: object

Response from the Workday API

data

filter

next()

page

page_results

references

total_pages

total_results

CHAPTER 5

Errors

5.1 workday.exceptions module

exception workday.exceptions.**WorkdaySoapApiError**(*fault*)
Bases: Exception

exception workday.exceptions.**WsdlNotProvidedError**(*service*)
Bases: Exception

CHAPTER 6

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://github.com/tonybaloney/workday/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

6.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

6.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

6.1.4 Write Documentation

workday could always use more documentation, whether as part of the official workday docs, in docstrings, or even on the web in blog posts, articles, and such.

6.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/tonybaloney/workday/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

6.2 Get Started!

Ready to contribute? Here's how to set up *workday* for local development.

1. Fork the *workday* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/workday.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv workday
$ cd workday/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 workday tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

6.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/tonybaloney/workday/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_workday
```

CHAPTER 7

History

7.1 0.4.0 (2018-06-27)

- Implemented paging by making WorkdayResponse objects iterable

7.2 0.3.0 (2018-06-23)

- Added test framework, setup package for distribution

7.3 0.2.0 (2018-06-22)

- WS-Security support
- Protected WSDL support
- Paging support

7.4 0.1.0 (2018-06-22)

- First release on PyPI.
- Template for Talent API (SOAP) method execution

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

Index