EditorConfig Python Core Documentation

Release 0.12.0

EditorConfig Team

Sep 27, 2017

Contents

EditorConfig is a file format for defining file-specific coding styles and a set of plugins that allow text editors and IDEs to read this file format. For more information on the EditorConfig project visit the EditorConfig Homepage.

The EditorConfig Python Core is a Python package for locating and parsing EditorConfig files. This package can be used as an import by other Python code or as a stand-alone command line program.

Contents:

CHAPTER 1

Usage

Installation

First you will need to install the EditorConfig Python Core package.

To install from PyPI using pip:

```
pip install editorconfig
```

Discovering EditorConfig properties

The `get_properties` function can be used to discover EditorConfig properties for a given file. Example:

```
import logging
from editorconfig import get_properties, EditorConfigError

filename = "/home/zoidberg/humans/anatomy.md"

try:
    options = get_properties(filename)
except EditorConfigError:
    logging.warning("Error getting EditorConfig properties", exc_info=True)
else:
    for key, value in options.items():
        print "%s=%s" % (key, value)
```

The `get_properties` method returns a dictionary representing EditorConfig properties found for the given file. If an error occurs while parsing a file an exception will be raised. All raised exceptions will inherit from the `EditorConfigError` class.

Handling Exceptions

All exceptions raised by EditorConfig will subclass `EditorConfigError`. To handle certain exceptions specially, catch them first. More exception classes may be added in the future so it is advisable to always handle general `EditorConfigError` exceptions in case a future version raises an exception that your code does not handle specifically.

Exceptions module reference

Exceptions can be found in the `editorconfig.exceptions` module. These are the current exception types:

exception `editorconfig.exceptions.`**`EditorConfigError`**
Parent class of all exceptions raised by EditorConfig

exception `editorconfig.exceptions.`**`ParsingError`**(*filename*)
Error raised if an EditorConfig file could not be parsed

exception `editorconfig.exceptions.`**`PathError`**
Error raised if invalid filepath is specified

exception `editorconfig.exceptions.`**`VersionError`**
Error raised if invalid version number is specified

Exception handling example

An example of custom exception handling:

```
import logging
from editorconfig import get_properties
from editorconfig import exceptions

filename = "/home/zoidberg/myfile.txt"

try:
    options = get_properties(filename)
except exceptions.ParsingError:
    logging.warning("Error parsing an .editorconfig file", exc_info=True)
except exceptions.PathError:
    logging.error("Invalid filename specified", exc_info=True)
except exceptions.EditorConfigError:
    logging.error("An unknown EditorConfig error occurred", exc_info=True)

for key, value in options.iteritems():
    print "%s=%s" % (key, value)
```

CHAPTER 2

Command Line Usage

The EditorConfig Python Core can be used from the command line in the same way as the EditorConfig C Core.

Discovering EditorConfig properties

Installing EditorConfig Python Core should add an `editorconfig.py` command to your path. This command can be used to locate and parse EditorConfig files for a given full filepath. For example:

```
editorconfig.py /home/zoidberg/humans/anatomy.md
```

When used to retrieve EditorConfig file properties, `editorconfig.py` will return discovered properties in *key*=*value* pairs, one on each line.

CHAPTER 3

Writing Plugins

The EditorConfig Python Core can be easily used by text editor plugins written in Python or plugins that can call an external Python interpreter. The EditorConfig Python Core supports Python versions 2.2 to 2.7. Check out the Vim and Gedit plugins for example usages of the EditorConfig Python Core.

Use as a library

For instructions on using the EditorConfig Python Core as a Python library see *Usage*.

Using with an external Python interpreter

The EditorConfig Python Core can be used with an external Python interpreter by executing the `main.py` file. The `main.py` file can be executed like so:

```
python editorconfig-core-py/main.py /home/zoidberg/humans/anatomy.md
```

For more information on command line usage of the EditorConfig Python Core see *Command Line Usage*.

Bundling EditorConfig Python Core with Plugin

A text editor or IDE plugin will either need to bundle the EditorConfig Python Core with the plugin installation package or the will need to assist the user in installing the EditorConfig Python Core. Below are instructions for bundling the EditorConfig Python Core with plugins.

Bundling as a Submodule in Git

Git submodules allow one repository to be included inside another. A submodule stores a remote repositry and commit to use for fetching the embedded repository. Submodules take up very little space in the repository since they do not

actually include the code of the embedded repository directly.

To add EditorConfig Python Core as a submodule in the `editorconfig-core-py` directory of your repository:

```
git submodule add git://github.com/editorconfig/editorconfig-core-py.git editorconfig-
↪core-py
```

Then every time the code is checked out the submodule directory should be initialized and updated:

```
git submodule update --init
```

Bundling as a Subtree in Git

Git subtrees are convenient because, unlike submodules, they do not require any extra work to be performed when cloning the git repository. Git subtrees include one git codebase as a subdirectory of another.

Example of using a subtree for the `editorconfig` directory from the EditorConfig Python Core repository:

```
git remote add -f editorconfig-core-py git://github.com/editorconfig/editorconfig-
↪core-py.git
git merge -s ours --no-commit editorconfig-core-py/master
git read-tree --prefix=editorconfig -u editorconfig-core-py/master:editorconfig
git commit
```

For more information on subtrees consult the subtree merge guide on Github and Chapter 6.7 in the book Pro Git.

Index