pythondi

Release 1.2.2

Sep 02, 2022

Contents

pythondi is Lightweight dependency injection library for python.

pythondi is developed on GitHub. Contributions are welcome!

CHAPTER 1

Installation

```
pip3 install sanic
```

GitHub: https://github.com/teamhide/pythondi

PyPI: https://pypi.org/project/pythondi

CHAPTER 2

Guides

2.1 API Reference

2.2 Examples

This section of the documentation is a simple collection of example code that can help you get a quick start on your application development.

2.2.1 Basic Examples

```
from pythondi import Provider, configure, configure_after_clear, inject
from .repo import Repo, SQLRepo


class Usecase:
    @inject()
    def __init__(self, repo: Repo):
        self.repo = repo


if __name__ == '__main__':
    # Init provider
    provider = Provider()
    provider.bind(Repo, SQLRepo)

    # Inject with configure
    configure(provider=provider)

    # Or if you want to fresh inject, use `configure_after_clear`
    configure_after_clear(provider=provider)
```

(continues on next page)

(continued from previous page)

```
    # Init class without arguments
    u = Usecase()
```

2.2.2 Flask Example

```
from flask import Flask, Blueprint, jsonify

from pythondi import Provider, configure, inject
import abc

bp = Blueprint('home', __name__)


class Repo:
    """Interface class"""
    __metaclass__ = abc.ABCMeta

    @abc.abstractmethod
    def get(self):
        pass


class SQLRepo(Repo):
    """Impl class"""
    def __init__(self):
        pass

    def get(self):
        print('SQLRepo')


@bp.route('/')
def home():
    usecase = Usecase()
    usecase.repo.get()
    return jsonify({'hello': 'world'})


class Usecase:
    @inject()
    def __init__(self, repo: Repo):
        self.repo = repo


def create_app():
    provider = Provider()
    provider.bind(Repo, SQLRepo)
    configure(provider=provider)
    app = Flask(__name__)
    app.register_blueprint(bp)
    return app


if __name__ == '__main__':
    app = create_app()
```

(continues on next page)

(continued from previous page)

```
    app.run(debug=True)
```

2.2.3 Sanic Example

```
import abc

from sanic import Sanic, Blueprint
from sanic.response import json

from pythondi import Provider, configure, inject


class Repo:
    """Interface class"""
    __metaclass__ = abc.ABCMeta

    @abc.abstractmethod
    def get(self):
        pass


class SQLRepo(Repo):
    """Impl class"""
    def __init__(self):
        pass

    def get(self):
        print('SQLRepo')


bp = Blueprint('home', url_prefix='/')


@bp.route('/')
async def home(request):
    usecase = Usecase()
    usecase.repo.get()
    return json({'hello': 'world'})


class Usecase:
    @inject()
    def __init__(self, repo: Repo):
        self.repo = repo


def create_app():
    provider = Provider()
    provider.bind(Repo, SQLRepo)
    configure(provider=provider)
    app = Sanic(__name__)
    app.blueprint(bp)
    return app
```

(continues on next page)

(continued from previous page)

```
if __name__ == '__main__':
    app = create_app()
    app.run(debug=True)
```

2.2.4 Django Example

```
"""
In case of django, just put the initializing code inside of django startup

You can use project folder's __init__.py or urls.py
"""
```