object-graph-builder

Jan 19, 2021

Contents

Provides a multi-step dependency injection container builder to allow adding/recreating object graphs in different stages through an app life-cycle

CHAPTER 1

Features

- Offering a container builder, so one can dynamically build the object graph with many apps adding their specs, classes, modules
- Dependency Injection Container (pinject)

CHAPTER 2

Requirements

- Python 3.6+

CHAPTER 3

Usage

CHAPTER 4

Prepare for development

A Python 3.6+ interpreter is required in addition to pipenv.

```
$ make init
```

Now you're ready to run the tests:

```
$ make test
```

CHAPTER 5

Resources

- Documentation
- Bug Tracker
- Code

Contents:

5.1 Installation

- Install with pip:

```
pip install object-graph-builder
```

5.2 1. Framework Agnostic App

In container.py

```
import object_graph.builder

builder =  object_graph.builder.ObjectGraphBuilder()
```

App 1

```
import container
import pinject


class MyService(object):
        def __init__(long_name: SomeReallyLongClassName):
```

(continues on next page)

(continued from previous page)

```
                self.my_dep = long_name

class MyBindingSpec(pinject.BindingSpec):
         def configure(self, bind):
                 bind('long_name', to_class=SomeReallyLongClassName)

container.builder.addBindingSpec(MyBindingSpec)
```

App 2

```
import container


container.builder.addModules([app2.module1, app2.module2])
```

Client

```
import container

object_graph = container.builder.get_object_graph()

my_service = object_graph.provide(MyService)
```

5.3 2. Django Example

One can define the builder in your settings.py and then import it in each app and add the configurations you need

In settings.py

```
import object_graph.builder

object_graph_builder =  object_graph.builder.ObjectGraphBuilder()
```

App 1 in apps.App1Config.ready()

```
from config.settings import object_graph_builder
import pinject


class MyService(object):
        def __init__(long_name: SomeReallyLongClassName):
                self.my_dep = long_name

class MyBindingSpec(pinject.BindingSpec):
         def configure(self, bind):
                 bind('long_name', to_class=SomeReallyLongClassName)

object_graph_builder.addBindingSpec(MyBindingSpec)
```

App 2 in apps.App2Config.ready()

```
from config.settings import object_graph_builder


object_graph_builder.addModules([app2.module1, app2.module2])
```

Client

```
object_graph = object_graph_builder.get_object_graph()

my_service = object_graph.provide(MyService)
```

Each time you call *object_graph_builder.get_object_graph()*, it will check if it needs to rebuild the object graph.

5.4 Changelog

5.4.1 0.0.2

- Offering a container builder, so one can dynamically build the object graph with many apps adding their specs, classes, modules

CHAPTER 6

Indices and tables

- genindex
- modindex
- search