

---

# **glorpen-di Documentation**

*Release 1.5.0*

**Arkadiusz Dzięgiel**

**Oct 27, 2018**



---

## Contents

---

<b>1</b>	<b>Official repositories</b>	<b>3</b>
<b>2</b>	<b>Supported design patterns</b>	<b>5</b>
<b>3</b>	<b>Contents</b>	<b>7</b>
3.1	glorpen.di API Documentation . . . . .	7
3.2	Usage examples . . . . .	10
3.3	Using type hints for auto injection . . . . .	12
3.4	Adding custom scope . . . . .	12
<b>4</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>



Another Dependency Injection library for Python.

This package has following three guidelines:

- any class configured by *Container* mechanism should not be modified in any way
- there is no need for external services definition files for *Container*
- no *Container* compiling and service tagging - we have introspection and dynamic language for this task

And so this package provides:

- **no** xml configuration
- **no** annotations (more cons than pros)
- **no** changes to services code



# CHAPTER 1

---

## Official repositories

---

For forking and other funnies.

BitBucket: <https://bitbucket.org/glorpen/glorpen-di>

GitHub: <https://github.com/glorpen/glorpen-di>





---

### Supported design patterns

---

Service instance can be created by:

- factory service
- calling class object with arguments

Instance options can be altered by:

- constructor arguments
- setters
- calling methods
- using configurator service

Each service has defined scope, service cannot request other service from narrower scope.



## 3.1 glorpen.di API Documentation

### 3.1.1 glorpen.di

Dependency injection component.

`glorpen.di.__version__`  
Current package version.

**class** `glorpen.di.Container`  
Alias to `glorpen.di.container.Container`.

### 3.1.2 glorpen.di.container

**class** `glorpen.di.container.Alias` (*target*)  
Alias for service.

**class** `glorpen.di.container.Container`  
Implementation of DIC container.

**add\_alias** (*service, alias*)  
Adds an alias for given service

**add\_parameter** (*name, value*)  
Adds a key-value parameter.

**add\_service** (*name*)  
Adds service definition to this container.

*name* argument should be a class, import path, or string if `Service.implementation()` will be used.

**Returns:** `Service`

**get** (*svc*)

Gets service instance.

**Raises:** UnkownServiceException

**get\_definition** (*svc*)

Returns definition for given service name.

**get\_parameter** (*name*)

Gets parameter.

**Raises:** UnkownParameterException

**set\_scope\_hierarchy** (*\*scopes*)

Sets used scopes hierarchy.

Arguments should be scopes sorted from widest to narrowest. A service in wider scope cannot request service from narrower one.

Default is: `[glorpen.di.scopes.ScopeSingleton, glorpen.di.scopes.ScopePrototype]`.

**Args:** classes or instances of `glorpen.di.scopes.ScopeBase`

**class** `glorpen.di.container.Deferred` (*service=None, method=None, param=None*)

Class for marking values for lazy resolving.

Values are resolved by `Container` upon service creation.

**resolve** (*getter, param\_getter*)

Given (service) getter and param\_getter, returns resolved value.

**class** `glorpen.di.container.Kwargs` (*\*\*kwargs*)

Simply wraps given kwargs for later use.

**classmethod** `merge` (*\*args*)

Merges iterable arguments, can be `dict` or `Kwargs` instance. @return: Resulting dict

**class** `glorpen.di.container.Service` (*name\_or\_impl*)

Service definition.

When filling arguments for constructor and method calls you can use:

- `my_var__svc=MyClass` - will inject service `MyClass` to `my_var`
- `my_var__param="my.param"` - will inject parameter named "my.param" to `my_var`

Implementation value for service can be:

- class instance
- string with import path
- callable

**call** (*method, kwargs=None, \*\*kwargs\_inline*)

Adds a method call after service creation with given arguments.

**Returns:** `Service`

**call\_with\_signature** (*method, kwargs=None, \*\*kwargs\_inline*)

Adds a method call after service creation with given arguments. Arguments detected from function signature are added if not already present.

**Returns:** `Service`

**configurator** (*service=None, method=None, callable=None, kwargs=None, \*\*kwargs\_inline*)

Adds service or callable as configurator of this service.

**Args:** service + method, callable: given service method/callable will be called with instance of this service.

**Returns:** *Service*

**factory** (*service=None, method=None, callable=None, kwargs=None, \*\*kwargs\_inline*)

Sets factory callable.

**Returns:** *Service*

**implementation** (*v*)

Sets service implementation (callable).

**Returns:** *Service*

**kwargs** (*\*\*kwargs*)

Adds service constructor arguments.

**Returns:** *Service*

**kwargs\_from\_signature** ()

Adds arguments found in class signature, based on provided function hints.

**Returns:** *Service*

**kwargs\_modifier** (*service=None, method=None, callable=None, kwargs=None, \*\*kwargs\_inline*)

Adds service or callable as constructor arguments modifier of this service.

**Args:** service + method, callable: given service method/callable will be called with kwargs of this service.

**Returns:** *Service*

**scope** (*scope\_cls*)

Sets service scope.

**Returns:** *Service*

**set** (*\*\*kwargs*)

Will `setattr()` given arguments on service.

**Returns:** *Service*

`glorpen.di.container`.**fluid** (*f*)

Decorator for applying fluid pattern to class methods and to disallow calling when instance is marked as frozen.

`glorpen.di.container`.**normalize\_name** (*o*)

Gets object "name" for use with *Container*.

**Args:** o (object): Object to get name for

**Returns:** str

**Raises:** Exception

### 3.1.3 glorpen.di.scopes

Built-in scopes.

**class** `glorpen.di.scopes.ScopeBase`

Base class for all scopes.

**class** glorpen.di.scopes.ScopePrototype  
Scope that creates new instance of given service each time it is requested.

**class** glorpen.di.scopes.ScopeSingleton  
Scope that creates instance of given service only once.

### 3.1.4 glorpen.di.exceptions

Exceptions used by *glorpen.di*.

**exception** glorpen.di.exceptions.ContainerException  
Base exception

**exception** glorpen.di.exceptions.InjectionException(*svc\_name*, *cls*,  
*method\_name=None*)  
Raised when service or its method cannot be created or called

**exception** glorpen.di.exceptions.InvalidAliasTargetException(*name*)  
Raised when adding alias to alias or not existing service

**exception** glorpen.di.exceptions.RecursionException(*s\_def*, *requester\_chain*)  
Raised when service definition is requiring itself.

**exception** glorpen.di.exceptions.ScopeWideningException(*s\_def*, *requester\_chain*)  
Raised when *Service A* depends on *Service B* from narrower scope.

**exception** glorpen.di.exceptions.ServiceAlreadyCreated(*svc\_name*)  
Raised when service definition is changed but service is already created by *glorpen.di.container.Container*.

**exception** glorpen.di.exceptions.UnknownParameterException(*name*)  
Raised when requesting parameter which is not registered in *glorpen.di.container.Container*.

**exception** glorpen.di.exceptions.UnknownScopeException(*scope*, *svc\_name*)  
Raised when service has *scope* not assigned to *glorpen.di.container.Container* by *glorpen.di.container.Container.set\_scope\_hierarchy()*.

**exception** glorpen.di.exceptions.UnknownServiceException(*svc\_name*)  
Raised when requesting service name which is not registered in *glorpen.di.container.Container*.

### 3.1.5 glorpen.di.tests

## 3.2 Usage examples

### 3.2.1 Injecting services and parameters

```
from glorpen.di import Container

class MyParamService(object):
    pass

class MyService(object):
    def __init__(self, obj, text, value):
        super(MyService, self).__init__()
        print("service instance: %s" % obj)
        print("container parameter: %s" % text)
```

(continues on next page)

(continued from previous page)

```

    print("provided value: %s" % value)

c = Container()

c.add_service(MyParamService)
c.add_parameter('my-param', "value from container")
c.add_service(MyService).kwargs(obj__svc=MyParamService, text__param="my-param",
↪value="defined value")

c.get(MyService)

```

Running snippet will print:

```

service instance: <__main__.MyParamService object at 0x7f2fef6e9828>
container parameter: value from container
provided value: defined value

```

### 3.2.2 Arguments

In cases when you want to use inject parameter already used by internal methods, eg. *glorpen.di.container.Service.call()*, you can pass args with *glorpen.di.container.Kwargs* helper class.

```

svc.configurator(callable=configurator, kwargs=Kwargs(router__svc=Router), other_
↪param="param")
svc.call("some_method", kwargs=Kwargs(router__svc=Router), other_param="param")

```

Arguments defined by *glorpen.di.container.Kwargs* are overriding ones provided by *\*\*kwargs* notation.

### 3.2.3 Configurators

Configurators are services or callables used to configure main service. Provided callables are given main service instance as first argument.

```

def configurator(obj, some_service):
    obj.some_thing = some_service()
svc.configurator(callable=configurator, some_service__svc=MyClass)
svc.configurator(service=ConfiguratorClass, method="some_method", some_service__
↪svc=MyClass)

```

### 3.2.4 Factories

Services that create other objects. It is possible to provide parameters/other services from Container to given callables.

```

def factory(some_service):
    return some_service("arg1")
svc1.factory(callable=factory, some_service__svc=MyClass)

class FactoryClass(object):
    def create_new(self, some_service):
        return some_service("arg1")
svc2.factory(service=FactoryClass, method="create_new", some_service__svc=MyClass)

```

### 3.2.5 Calling methods and settings properties

To call method on service creation:

```
svc.call("some_method", some_service__svc=MyClass)
```

To set properties on service creation:

```
svc.set(my_prop__svc=MyClass)
```

## 3.3 Using type hints for auto injection

Sometimes it is easier to just auto-fill function arguments, when using Python3 it can be done by arguments type hinting (see `typing` for more information).

You can enable function hints lookup by using `glorpen.di.container.Service.kwargs_from_signature()` for constructor arguments and `glorpen.di.container.Service.call_with_signature()` for methods.

```
from glorpen.di import Container

class MyParamService(object):
    pass

class MyService(object):
    def __init__(self, param:MyParamService):
        super(MyService, self).__init__()
        print("MyService.__init__: %s" % param.__class__.__name__)

    def some_method(self, param:MyParamService):
        print("MyService.some_method: %s" % param.__class__.__name__)

c = Container()

c.add_service(MyParamService)
c.add_service(MyService).kwargs_from_signature().call_with_signature("some_method")

print("Container.get: %s" % c.get(MyService).__class__.__name__)
```

Snippet will create following output:

```
MyService.__init__: MyParamService
MyService.some_method: MyParamService
Container.get: MyService
```

## 3.4 Adding custom scope

You can define new scope by extending `glorpen.di.scopes.ScopeBase` and using `glorpen.di.container.Container.set_scope_hierarchy()`.

```
from glorpen.di.scopes import ScopePrototype, ScopeSingleton, ScopeBase
from random import randint
```

(continues on next page)



(continued from previous page)

```
class RandomScope (ScopeBase) :
    """Returns new or cached instances based on random factor."""
    def __init__(self, randomness=3):
        super(RandomScope, self).__init__()
        self.rnd = randomness
        self.instances = {}

    def get(self, creator, name):
        if not name in self.instances or randint(0, self.rnd) == 0:
            self.instances[name] = creator()
        return self.instances[name]

c = Container()

# add scope with parameter
c.set_scope_hierarchy(ScopeSingleton, RandomScope(5), ScopePrototype)

# configure "str" service so we can see instances count
counter = 0
def configurator(kwargs):
    global counter
    kwargs.setdefault("object", "instance number: %d" % counter)
    counter+=1

c.add_service('arg.test').implementation(str)\
    .configurator(args_callable=configurator)\
    .scope(RandomScope)

for i in range(0,10):
    print(c.get("arg.test"))
```

Running script will print:

```
instance number: 0
instance number: 0
instance number: 0
instance number: 0
instance number: 1
instance number: 2
instance number: 2
instance number: 3
instance number: 4
instance number: 4
```



## CHAPTER 4

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



**g**

`glorpen.di`, 7  
`glorpen.di.container`, 7  
`glorpen.di.exceptions`, 10  
`glorpen.di.scopes`, 9  
`glorpen.di.tests`, 10



---

## Symbols

`__version__` (in module `glorpen.di`), 7

### A

`add_alias()` (`glorpen.di.container.Container` method), 7  
`add_parameter()` (`glorpen.di.container.Container` method), 7  
`add_service()` (`glorpen.di.container.Container` method), 7  
Alias (class in `glorpen.di.container`), 7

### C

`call()` (`glorpen.di.container.Service` method), 8  
`call_with_signature()` (`glorpen.di.container.Service` method), 8  
`configurator()` (`glorpen.di.container.Service` method), 8  
Container (class in `glorpen.di`), 7  
Container (class in `glorpen.di.container`), 7  
ContainerException, 10

### D

Deferred (class in `glorpen.di.container`), 8

### F

`factory()` (`glorpen.di.container.Service` method), 9  
`fluid()` (in module `glorpen.di.container`), 9

### G

`get()` (`glorpen.di.container.Container` method), 7  
`get_definition()` (`glorpen.di.container.Container` method), 8  
`get_parameter()` (`glorpen.di.container.Container` method), 8  
`glorpen.di` (module), 7  
`glorpen.di.container` (module), 7  
`glorpen.di.exceptions` (module), 10  
`glorpen.di.scopes` (module), 9  
`glorpen.di.tests` (module), 10

### I

`implementation()` (`glorpen.di.container.Service` method), 9  
InjectionException, 10  
InvalidAliasTargetException, 10

### K

Kwargs (class in `glorpen.di.container`), 8  
`kwargs()` (`glorpen.di.container.Service` method), 9  
`kwargs_from_signature()` (`glorpen.di.container.Service` method), 9  
`kwargs_modifier()` (`glorpen.di.container.Service` method), 9

### M

`merge()` (`glorpen.di.container.Kwargs` class method), 8

### N

`normalize_name()` (in module `glorpen.di.container`), 9

### R

RecursionException, 10  
`resolve()` (`glorpen.di.container.Deferred` method), 8

### S

`scope()` (`glorpen.di.container.Service` method), 9  
ScopeBase (class in `glorpen.di.scopes`), 9  
ScopePrototype (class in `glorpen.di.scopes`), 9  
ScopeSingleton (class in `glorpen.di.scopes`), 10  
ScopeWideningException, 10  
Service (class in `glorpen.di.container`), 8  
ServiceAlreadyCreated, 10  
`set()` (`glorpen.di.container.Service` method), 9  
`set_scope_hierarchy()` (`glorpen.di.container.Container` method), 8

### U

UnknownParameterException, 10  
UnknownScopeException, 10  
UnknownServiceException, 10