Watson - DI

Release 2.1.0

September 30, 2014

Contents

Watson Di is a simple dependency injection container that can store and retrieve dependencies.

CHAPTER 1

Build Status

CHAPTER 2

Installation

```
pip install watson-di
```

CHAPTER 3

Testing

Watson can be tested with py.test. Simply activate your virtualenv and run `python setup.py test`.

CHAPTER 4

Contributing

If you would like to contribute to Watson, please feel free to issue a pull request via Github with the associated tests for your code. Your name will be added to the AUTHORS file under contributors.

CHAPTER 5

Table of Contents

5.1 Usage

The container is configured via a dict containing the following keys:

params A dict of data that can be injected into a dependency. If the value of the key is the same as the name of another dependency then the dependency will be referenced.

definitions A dict of definitions that are to be loaded by the container. Available keys within a definition are:

- **item** The qualified name of a class or function
- **type** singleton (only load the dependency once) or prototype (instantiate and return a new dependency on each request)
- **init** A list or dict of items to be injected into the dependency on instantiation.
- **setter** A list or dict of methods to be called upon instantiation.
- **property:** A list or dict of methods to be called upon instantiation.

Only 'item' is a required key.

processors A dict of events to be listened for and processors to be called.

```
container = IocContainer({
    'params': {
        'db.host': 'localhost'
    },
    'definitions': {
        'database': {
            'item': 'db.adapters.MySQL'
            'init': {
                'host': 'db.host',
                'username': 'simon',
                'password': 'test',
                'db': 'test'
            }
        }
    }
})
db = container.get('database')  # an instance of db.adapters.MySQL
```

5.2 Reference Library

5.2.1 watson.di

class `watson.di.`**`ContainerAware`**

An interface for classes that should have a container.

Primarily used by the IocContainer, any class that subclasses it will have the container it was called from automatically injected into it.

This allows classes to use the container as a service locator.

By defining a __ioc_definition__ on the class, any class that is retrieved from the container that hasn't been defined can create itself based off the definition.

`container`

watson.di.container.IocContainer

A reference to the container

`__ioc_definition__`

dict

A definition required to create the object

`container`

Returns The instance of the injected container.

5.2.2 watson.di.container

class `watson.di.container.`**`IocContainer`**(*config=None*)

A simple dependency injection container that can store and retrieve dependencies for an application.

The container is configured via a dict containing the following keys.

params A dict of data that can be injected into a dependency. If the value of the key is the same as the name of another dependency then the dependency will be referenced.

definitions A dict of definitions that are to be loaded by the container. Available keys within a definition are as follows.

item The qualified name of a class or function

type singleton (only load the dependency once) or prototype (instantiate and return a new dependency on each request)

init A list or dict of items to be injected into the dependency on instantiation.

setter A list or dict of methods to be called upon instantiation.

property Same as setter

Only 'item' is a required key.

processors A dict of events to be listened for and processors to be called.

Example:

```
container = IocContainer({
    'params': {
        'db.host': 'localhost'
    },
    'definitions': {
        'database': {
            'item': 'db.adapters.MySQL'
            'init': {
                'host': 'db.host',
                'username': 'simon',
                'password': 'test',
                'db': 'test'
            }
        }
    }
})
db = container.get('database')  # an instance of db.adapters.MySQL
```

config
dict

A dict containing the definitions, params and processors.

__instantiated__
dict

A cache of already instantiated dependencies.

__init__(*config=None*)
Initializes the container and set some default configuration options.

Parameters **config** (*dict*) – The params, definitions and processors.

_get_dependency(*definition*)
Loads a definition item.

add(*name*, *obj*, *type_='singleton'*)
Add an instantiated dependency to the container.

Parameters

- **name** (*string*) – The name used to reference the dependency
- **obj** (*mixed*) – The dependency to add
- **type** (*string*) – prototype|singleton depending on if it should be instantiated on each IocContainer.get call.

add_definition(*name*, *definition*)
Adds a dependency definition to the container.

Parameters

- **name** (*string*) – The name used to reference the dependency
- **definition** (*dict*) – The definition of the dependency.

attach_processor(*event*, *processor*)
Attach a processor to the container.

Attaches a processor to the container that will be triggered on a specific event.

Parameters

- **event** (*string*) – The name of the event (watson.di.container.POST_EVENT or PRE_EVENT)
- **processor** (*watson.di.processors.BaseProcessor*) – The processor to attach.

`definitions`
Convenience method for retrieving the definitions.

Returns A dict of params.

Return type dict

`get`(*name*)
Retrieve a dependency from the container.

Parameters **name** (*string*) – The name of the dependency to retrieve.

Raises `KeyError` – If the definition or item within the definition are not specified.

Returns The dependency

Return type mixed

`params`
Convenience method for retrieving the params.

Returns A dict of params.

Return type dict

5.2.3 watson.di.processors

class `watson.di.processors.`**`AttributeInjection`**
Responsible for injecting required values into attributes.

Parameters **event** (*watson.events.types.Event*) – The event dispatched from the container.

Returns The dependency

Return type mixed

class `watson.di.processors.`**`Base`**
The base processor that all other processors should extend.

When a processor is called from the container the following parameters are sent through with the event.

- definition: The dict definition of the dependency
- dependency: The name of the dependency

Depending on the event, a different target will also be sent with the event.

- watson.di.container.PRE_EVENT: The dict definition of the dependency
- watson.di.container.POST_EVENT: The initialized dependency

class `watson.di.processors.`**`ConstructorInjection`**
Responsible for initializing the dependency.

Responsible for initializing the dependency and injecting any required values into the constructor.

Parameters **event** (*watson.events.types.Event*) – The event dispatched from the container.

Returns The dependency

Return type mixed

class `watson.di.processors.`**`ContainerAware`**
Injects the container into a dependency.

Responsible for injecting the container in any class that extends watson.di.ContainerAware. The container is then accessible via object.container

Parameters **event** (*watson.events.types.Event*) – The event dispatched from the container.

Returns The dependency

Return type mixed

class `watson.di.processors.`**`SetterInjection`**

Responsible for injecting required values into setter methods.

Parameters **event** (*watson.events.types.Event*) – The event dispatched from the container.

Returns The dependency

Return type mixed

`watson.di.processors.`**`get_param_from_container`**(*param*, *container*)

Internal function used by the container.

Retrieve a parameter from the container, and determine whether or not that parameter is an existing dependency.

Returns The dependency (if param name is the same as a dependency), the param, or the value of the param.

Return type mixed

Python Module Index