Watson - DI

Release 1.0.0

September 30, 2014

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

Build Status

Installation

```
pip install watson-di
```

Testing

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

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.

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.
>
> **container**
>
> > **Returns** The instance of the injected container.

5.2.2 watson.di.container

5.2.3 watson.di.processors

W