# Watson - Events

***Release 1.0.3***

April 08, 2015

Contents

Trigger and handle event flow with your application.

CHAPTER 1

# Build Status

CHAPTER 2

# Installation

```
pip install watson-events
```

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

Using the event dispatcher is a three step process.

1. Create the event dispatcher
2. Add your event listener to the dispatcher
3. Trigger the dispatcher

```
from watson.events import dispatcher, types

dispatcher = dispatcher.EventDispatcher()  # create the dispatcher
dispatcher.add('MyEvent', lambda x: x.name)  # add your event listener
result = dispatcher.trigger(types.Event('SampleEvent'))  # trigger the event

print(result.first())  # 'SampleEvent'
```

## 5.2 Reference Library

### 5.2.1 watson.events.collections

**class** watson.events.collections.**Listener**

A list of listeners to be used in an EventDispatcher.

A Listener Collection is a list of callbacks that are to be triggered by an event dispatcher. Each item in the list contains the callback, a priority, and whether or not the callback should only be triggered once.

**add**(*callback*, *priority=1*, *only_once=False*)

Adds a new callback to the collection.

**Parameters**

- **callback** (*callable*) – the function to be triggered
- **priority** (*int*) – how important the callback is in relation to others
- **only_once** (*bool*) – the callback should only be fired once and then removed

**Raises** **TypeError if non-callable is added.** –

**remove**(*callback*)

Removes all callbacks matching *callback* from the collection.

**Parameters** **callback** (*callable*) – the callback to be removed.

**sort_priority**()
Sort the collection based on the priority of the callbacks.

watson.events.collections.**ListenerDefinition**
alias of Listener

**class** watson.events.collections.**Result**
A list of responses from a EventDispatcher.trigger call.

A result collection contains all the resulting output from an event that has been triggered from an event dispatcher. It provides some convenience methods to deal with the results.

**first**(*default=None*)
Return the first result from the list.

**Parameters** **default** (*mixed*) – The value to return if the index doesn't exist

**Returns** The first result

**Return type** mixed

**last**(*default=None*)
Return the last result from the list.

**Parameters** **default** (*mixed*) – The value to return if the index doesn't exist

**Returns** The first result

**Return type** mixed

### 5.2.2 watson.events.dispatcher

**class** watson.events.dispatcher.**EventDispatcher**
Register and trigger events that will be executed by callables.

The EventDispatcher allows user defined events to be specified. Any listener that is triggered will have the event that was triggered passed to it as the first argument. Attributes can be added to the event params (see watson.events.types.Event) which can then be accessed by the listener.

Example:

```
dispatcher = EventDispatcher()
dispatcher.add('MyEvent', lambda x: x.name)
result = dispatcher.trigger(Event('SampleEvent'))
result.first()  # 'SampleEvent'
```

**add**(*event*, *callback*, *priority=1*, *only_once=False*)
Add an event listener to the dispatcher.

Adds an event listener to the relevant event listener collection. If a listener is set to once_only, it will be removed when the event is triggered on the EventDispatcher.

**Parameters**

- **event** (*string*) – The name of the event
- **callback** (*callable*) – A callable function to be triggered
- **priority** (*int*) – The priority of the listener (higher == more important)
- **once_only** (*boolean*) – When triggered, the listener will be removed

**Returns** A list of listeners attached to the event

**Return type** ListCollection

**clear**()
Clears all registered events from the event dispatcher.

**events**
Returns the events registered on the event dispatcher.

**has**(*event*, *callback=None*)
Return whether or not a callback is found for a particular event.

**remove**(*event*, *callback=None*)
Remove an event listener from the dispatcher.

Removes an event listener from the relevant Listener. If no callback is specified, all event listeners for that event are removed.

**Parameters**

- **event** (*string*) – The name of the event
- **callback** (*callable*) – A callable function to be triggered

**Returns** A list of listeners attached to the event

**Return type** Listener

**trigger**(*event*)
Fire an event and return a list of results from all listeners.

Dispatches an event to all associated listeners and returns a list of results. If the event is stopped (Event.stopped) then the Result returned will only contain the response from the first listener in the stack.

**Parameters** **event** (*watson.events.types.Event*) – The event to trigger

**Returns** A list of all the responses

**Return type** Result

**class** watson.events.dispatcher.**EventDispatcherAware**
Provides an interface for event dispatchers to be injected.

**dispatcher**
Retrieve the event dispatcher. If no event dispatcher exists, create a default one.

**Returns** An EventDispatcher object

### 5.2.3 watson.events.types

**class** watson.events.types.**Event**(*name*, *target=None*, *params=None*)
A base event that can be subclassed for use with an EventDispatcher.

Example:

```
def my_listener(event):
    print(event.params['config'])

dispatcher.add('MyEvent', my_listener)

event = Event('MyEvent')
event.params['config'] = {'some': 'config'}
dispatcher.trigger(event)
```

**__init__**(*name*, *target=None*, *params=None*)
Initializes the event.

Initialize the Event based on an event name. The name will be used when the event is triggered from the event dispatcher.

**Parameters**

- **name** (*string*) – the name of the event
- **target** (*mixed*) – the originating target of the event
- **params** (*dict*) – the params associated with the event

**stop_propagation**()
Prevents the event from triggering any more event listeners.

This should be used within an event listener when you wish to halt any further listeners from being triggered.

**stopped**
Return whether or not the event has been stopped.

## W

## Symbols



## A



## C



## D



## E



## F



## H



## L



## R



## S



## T



## W