
GooCalendar Documentation

Release 0.1

Samuel Abels, Cédric Krier, Antoine Smolders

February 19, 2013

CONTENTS

The `goolendar` module supplies a calendar widget drawn with GooCanvas that can display a month view and a week view. It also supplies classes to manage events you can add to the calendar.

CALENDAR OBJECTS

A `Calendar` is a calendar widget using GooCanvas that can display a month view and a week view. It holds an `EventStore` which contains events displayed in the calendar.

```
class goocalendar.Calendar ([event_store[, view[, time_format[, firstweekday]]]])
```

Creates a `Calendar` object. All arguments are optional. `event_store` should be an `EventStore`. `view` should be either 'month' or 'week'. Default value is 'month'. `time_format` determines the format of displayed time in the calendar. Default value is '%H:%M'. `firstweekday` is a constant of module `calendar` specifying the first day of the week. Default value is `calendar.SUNDAY`.

Instance attributes:

`goocalendar.event_store`

`EventStore` currently plugged. Setting a new event store will automatically redraw the canvas to display the events of the new event store.

`goocalendar.view`

The current view: 'month' or 'week'.

`goocalendar.selected_date`

`datetime.date` which determines the current selected day in the calendar.

`goocalendar.firstweekday`

Determines the first day of the week (0 is Monday).

Instance methods:

`goocalendar.select (date)`

Select the given date in the calendar. Date should be a `datetime.date`.

`goocalendar.previous_page ()`

Go to the previous page of the calendar.

`goocalendar.next_page ()`

Go to the next page of the calendar.

`goocalendar.set_view (view)`

Change calendar's view. Possible values: 'month' or 'week'.

`goocalendar.draw_events ()`

Redraws events.

`goocalendar.update ()`

Redraws calendar and events.

Instance signals:

`event-pressed`

The `event-pressed` signal is emitted when an `Event` is pressed with the button 1 of the mouse.

```
def callback(calendar, event, user_param1, ...)
```

calendar The `Calendar` that received the signal.

event The pressed `Event` object.

user_param1 the first user parameter (if any) specified with the `connect()` method.

... additional user parameters (if any).

`event-activated`

The `event-activated` signal is emitted when an `Event` is double-clicked with the button 1 of the mouse.

```
def callback(calendar, event, user_param1, ...)
```

calendar The `Calendar` that received the signal.

event The double-clicked `Event` object.

user_param1 the first user parameter (if any) specified with the `connect()` method.

... additional user parameters (if any).

`event-released`

The `event-released` signal is emitted when the button 1 of the mouse is released on an event.

```
def callback(calendar, event, user_param1, ...)
```

calendar The `Calendar` that received the signal.

event The double-clicked `Event` object.

user_param1 the first user parameter (if any) specified with the `connect()` method.

... additional user parameters (if any).

`day-pressed`

The `day-pressed` signal is emitted when a day is pressed with the mouse button 1.

```
def callback(calendar, date, user_param1, ...)
```

calendar The `Calendar` that received the signal.

date `datetime.date` corresponding to the day pressed.

user_param1 the first user parameter (if any) specified with the `connect()` method.

... additional user parameters (if any).

`day-activated`

The `day-activated` signal is emitted when the day is double-clicked with the mouse button 1.

```
def callback(calendar, date, user_param1, ...)
```

calendar The `Calendar` that received the signal.

date `datetime.date` corresponding to the activated day.

user_param1 the first user parameter (if any) specified with the `connect()` method

... additional user parameters (if any).

`day-selected`

The day-selected signal is emitted when the selected day changes.

```
def callback(calendar, date, user_param1, ...)
```

calendar The `Calendar` that received the signal.

date `datetime.date` corresponding to the new selected day.

user_param1 the first user parameter (if any) specified with the `connect()` method.

... additional user parameters (if any).

view-changed

The view-changed signal is emitted when the view changes

```
def callback(calendar, view, user_param1, ...)
```

calendar The `Calendar` that received the signal.

view 'month' or 'week'

user_param1 the first user parameter (if any) specified with the `connect()` method

... additional user parameters (if any).

page-changed

The page-changed signal is emitted when the page currently showed in the calendar is changed.

```
def callback(calendar, date, user_param1, ...)
```

calendar The `Calendar` that received the signal.

date `datetime.date` corresponding to the selected day in the calendar.

user_param1 the first user parameter (if any) specified with the `connect()` method.

... additional user parameters (if any).

EVENTSTORE OBJECTS

An `EventStore` is the store of `Event` that can be plugged to a `Calendar`.

class `goocalendar.EventStore`
There is no arguments for this class.

Instance methods:

`goocalendar.add(event)`
Add the given event to the event store.

`goocalendar.remove(event)`
Remove the given event from the event store.

`goocalendar.clear()`
Remove all events from the event store and restore it to initial state.

`goocalendar.get_events(start, end)`
Returns a list of all events that intersect with the given start and end datetime. If no start time nor end time are given, the method returns a list containing all events.

Instance signals:

event-added

The event-added signal is emitted when an `Event` is added to the event store.

`def callback(event_store, event, user_param1, ...)`

event_store The `EventStore` that received the signal.

event The added `Event`.

user_param1 the first user parameter (if any) specified with the `connect()` method.

... additional user parameters (if any).

event-removed

The event-removed signal is emitted when an `Event` is removed from the event store.

`def callback(event_store, event, user_param1, ...)`

event_store The `EventStore` that received the signal.

event The removed `Event`.

user_param1 the first user parameter (if any) specified with the `connect()` method.

... additional user parameters (if any).

events-cleared

The `events-cleared` signal is emitted when the event store is cleared.

```
def callback(event_store, user_param1, ...)
```

event_store The `EventStore` that received the signal.

user_param1 the first user parameter (if any) specified with the `connect()` method.

... additional user parameters (if any).

EVENT OBJECTS

An `Event` represents an event in a `Calendar`.

```
class goocalendar.Event (caption, start[, end[, all_day[, text_color[, bg_color]]]])
```

caption argument is mandatory and will be the string displayed on the event. *start* argument is mandatory and determines the starting time of the event. It should be a `datetime`. All other arguments are optional. *end* argument may be a `datetime`, *all_day* a boolean value. An event will be considered as all day event if no *end* argument is supplied. *text_color* and *bg_color* arguments are supposed to be color strings.

Instance attributes:

```
goocalendar.id
```

Unique identification integer.

```
goocalendar.caption
```

Caption to display on the event in the calendar.

```
goocalendar.start
```

`datetime` determining event start time.

```
goocalendar.end
```

`datetime` determining event end time.

```
goocalendar.all_day
```

Boolean determining if the day is an all day event or a normal event.

```
goocalendar.text_color
```

String determining caption text color.

```
goocalendar.bg_color
```

String determining background color.

```
goocalendar.multidays
```

Boolean property determining if the event is longer than one day.

Supported operations:

All comparisons operations are supported.

`event1` is considered less than `event2` if it starts before `event2`. If two events start at the same time, the event which ends the first one is considered smaller.

Example usage:

```
>>> import datetime
>>> import goocalendar
>>> event_store = goocalendar.EventStore()
>>> calendar = goocalendar.Calendar(event_store)
```

```
>>> event = goocalendar.Event('Event number 1',
...     datetime.datetime(2012, 8, 21, 14),
...     datetime.datetime(2012, 8, 21, 17),
...     bg_color='lightgreen')
>>> event_store.add(event)
```

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

g

goocalendar, ??