Events

One of the most obvious applications of delegates is implementation of design pattern "Observer". In the framework implementation of this pattern is represented by class **ClickBlocks\Core\Event**.

The class contains static methods that allow to bind some event with one or several delegates. Binding of the delegate to the event occurs by using method **listen()**:

```
Event::listen('myEvent', 'MyClass::method1', 1);
Event::listen('myEvent', [new MyClass(), 'method2'], 2);
```

The first parameter is event name and the second method is a delegate that associated with this event. The third parameter is an integer number which responsible for the priority of delegate call. Delegates are invoked in order of priority (first called delegates with a higher priority, then with less). If no priority is specified, it is assumed to be zero.

You can trigger event by using method **fire()**:

```
Event::fire('myEvent', ['arg1', 'arg2', ...]);
```

The second parameter of the method determines an array of arbitrary values which will be passed to each delegate of the event as arguments.

Delegates are sequentially called according to their priority. If some delegate returns FALSE then all subsequent delegates (have a lower priority) will not be invoked.

Once triggered event may be initiated again. This means that all event delegates will be called again. This behavior is not always necessary. Sometimes it is necessary to implement a single call of a delegate or delegates. You can easily do this by using the appropriate method:

```
Event::once('myEvent', 'MyClass->foo', 5);
```

Parameters of method **once()** correspond parameters of method **listen()**. The only difference is that after calling of the delegate that given by the method **once()** it will be deleted from the event.

You also can manually remove delegates from events by using method **remove()**:

```
Event::remove('myEvent', 'MyClass->foo');
```

The first parameter of the method is event name and the second - a delegate.

To find the total number of delegates signed at an event, or the total number of delegates in all events you should use the method **listeners()**:

```
// Gets number of delegates of event "myEvent".
echo Event::listeners('myEvent');
// Gets total number of delegates in all events.
echo Event::listeners();
```