Marionette.Controller

Многоцелевой объект для использования в качестве контроллера для модулей и роутеров, а также в качестве медиатора для рабочего процесса приложения
и координации остальных объектов, view и т.д.

Базовое использование

Marionette.Controller может быть расширен (extended), как и другие Backbone и Marionette объекты. Он поддерживает стандартный `initialize` метод (см. применение), в нем присутствует
`EventBinder` (вследствие дополнения его прототипа методами Backbone.Events), и он сам может генерить события.

```
// определяем контроллер
var MyController = Marionette.Controller.extend({
  initialize: function(options){
    this.stuff = options.stuff;
  },
  doStuff: function(){
    this.trigger("stuff:done", this.stuff);
  }
});
// создаем экземпляр контроллера
var c = new MyController({
  stuff: "some stuff"
});
// используем встроенный EventBinder
c.listenTo(c, "stuff:done", function(stuff){
  console.log(stuff);
});
// делаем что-то
c.doStuff();
```

Закрытие контроллера

Любой экземпляр контроллера имеет встроенный `close`-метод, который отвязывает все события, которые были присоединены непосредственно к экземпляру контроллера,
а также те события, которые были привязаны посредством использования EventBinder-а контроллера.

Метод `close` генерит событие "close" и вызывает соответствующий `onClose`-метод:

```
// определяем контроллер с методом onClose:
var MyController = Marionette.Controller.extend({
  onClose: function(){
    // какой-то код, связанный с закрытием контроллера
  }
})

// создаем новый экземпляр контроллера
var contr = new MyController();

// и обработчики событий
contr.on("close", function(){ ... });
contr.listenTo(something, "bar", function(){...});

// закрываем контроллер - отвязываем все обработчики событий, вызываем событие "close" и метод onClose:
controller.close();
```

По поводу названия 'Controller'

Так уж повелось, что именование `Controller` является источником определенной путаницы, и может показаться неудачным, особенно учитывая тот факт, что его обязательно будут путать с контроллером из MVC-архитектуры. Надо сказать, что среди создателей Marionette неоднократно происходили дебаты и дискуссии по поводу того, что должно соответствовать данному объекту. В конце концов, мы все равно решили закрепить за ним название контроллера, поскольку типичный случай его использования - контролировать рабочий процесс приложения и (или) модулей (см. пример и связанную с архитектурным решением статью).

Но в действительности, это многоцелевой объект, который может исполнять совершенно разные роли в различных ситуациях.