Marionette.RegionManager

RegionManager предоставляет собой внятный способ управления экземплярами объекта Marionette.Region в приложении. RegionManager предназначается для использования другими объектами приложения, для облегчения хранения, поиска и удаления регионов из них. Примеры использования можно найти в коде объектов Marionette.Application и Marionette.Layout.

Основы использования

RegionManagers инстанциируется явным образом, посредством ключевого слова new, и к нему могут быть добавлены регионы - с помощью нескольких очевидных методов:

```
var rm = new Marionette.RegionManager();
var region = rm.addRegion("foo", "#bar");
var regions = rm.addRegions({
  baz: "#baz",
  quux: "ul.quux"
});
regions.baz.show(myView);
rm.removeRegion("foo");
```

RegionManager.addRegion

Данный метод принимает два параметра - имя региона и его определение:

```
var rm = new Marionette.RegionManager();
var region = rm.addRegion("foo", "#bar");
```

В этом примере к экземпляру RegionManager-а будет добавлен регион с именем "foo", определенный через jQuery-селектор - в DOM ему будет соответствовать элемент `#bar`.

Существует много других способов добавления региона, включая объектные литералы с различными параметрами, и экземпляры Region-а (см. документацию по Marionette.Region)

RegionManager.addRegions

Множество регионов также может быть добавлено путем использования метода `addRegions`, который принимает в качестве параметра объектный литерал, содержащий ключи - имена регионов, и значения - их дифиниции.

```
var rm = new Marionette.RegionManager();
var regions = rm.addRegions({
  foo: "#bar",
  baz: {
    selector: "#quux",
    type: MyRegionType
  }
});
regions.foo; //=> экземпляр "foo"
regions.bar; //=> экземпляр "bar"
```

Этот метод возвращает объектный литерал, содержащий имена всех регионов.

addRegions параметры по умолчанию

При добавлении множества регионов в RegionManager может оказаться полезным предоставлять ряд значений по умолчанию, которые могут быть применены ко всем добавляемым регионам.
Это может быть сделано посредством параметра `defaults`, который представляет собой объектный литерал с парами ключ-значение - все они, как было сказано, имеют отношение к любому из добавляемых регионов.

```
var rm = new Marionette.RegionManager();
var defaults = {
  type: MyRegionType
};
var regions = {
  foo: "#bar",
  baz: "#quux"
};
rm.addRegions(regions, defaults);
```

В этом примере все регионы будут добавлены как экземпляры типа `MyRegionType`.

RegionManager.get

Получить экземпляр региона можно с помощью метода `get`, которому задано в качестве параметра имя данного региона.

```
var rm = new Marionette.RegionManager();
rm.addRegion("foo", "#bar");
var region = rm.get("foo");
```

Прежде чем регион будет уделен из экземпляра RegionManager-а, сработает его метод `close`.

RegionManager.removeRegions

Можно быстро удалить все имеющиеся в экземпляре RegionManager-а регионы с помощью команды `removeRegions`:

```
var rm = new Marionette.RegionManager();
rm.addRegions({
 foo: "#foo",
 bar: "#bar",
 baz: "#baz"
});
rm.removeRegions();
```

...которая, соответственно, сначала закроет регионы, а потом удалит их.

RegionManager.closeRegions

Можно также быстро закрыть все имеющиеся в экземпляре RegionManager-а регионы с помощью команды `closeRegions`:

```
var rm = new Marionette.RegionManager();
rm.addRegions({
 foo: "#foo",
 bar: "#bar",
 baz: "#baz"
});
rm.closeRegions();
```

Эта команда закроет все регионы без удаления их из экземпляра RegionManager-а.

RegionManager.close

Экземпляр RegionManager-а может напрямую быть закрыт посредством вызова его метода `close`. Таким образом будут закрыты и удалены все его регионы.

```
var rm = new Marionette.RegionManager();
rm.addRegions({
 foo: "#foo",
 bar: "#bar",
 baz: "#baz"
});
rm.close();
```

RegionManager Events

По ходу работы с экземпляром RegionManager'а им будет порождаться ряд определенных событий.

Событие region:add

Событие "region:add" будет порождено, когда регион добавлен. Это позволит немедленно пускать в дело экземпляр региона, или, например, прикреплять регион к объекту, для сохранения имени данного региона в качестве ссылки на него:

```
var rm = new Marionette.RegionManager();
rm.on("region:add", function(name, region){
 myObject[name] = region;
});
rm.addRegion("foo", "#bar");
```

Событие region:remove

Соответственно, событие "region:remove" произойдет при удалении региона:

```
var rm = new Marionette.RegionManager();
rm.on("region:remove", function(name, region){
 delete myObject[name];
});
rm.addRegion("foo", "#bar");
rm.removeRegion("foo");
```

RegionManager итераторы

RegionManager наследует от underscore.js ряд методов для перебора объектов. Они работают в том же ключе, что и методы Backbone.Collection, также импортированные из underscore.
Например, можно перебрать коллекцию экземпляров регионов с помощью метода `each`:

```
var rm = new Marionette.RegionManager();
rm.each(function(region){
  //тут что-то делаем с регионами
});
```

Список методов underscore:

* forEach
* each
* map
* find
* detect
* filter
* select
* reject
* every
* all
* some
* any
* include
* contains
* invoke
* toArray
* first
* initial
* rest
* last
* without
* isEmpty
* pluck