Marionette.CompositeView

`CompositeView` происходит от `CollectionView`, его стоит использовать в деревовидных сценариях, где нужно отобразить одновременно лист и ветвь, или там, где коллекция должна быть отрендерена с применением оборачивающего шаблона.

К примеру, если рендерится представление для компонента "tree", может понадобиться рендеринг collection view, где модель и шаблон устроены таким образом, чтобы компонент имел возможность отбражать все внутренние ветки (деревья).

Тут можно предложить реализовать `modelView` в качестве модели, но если в вашем проекте нет аналогичной структуры, по умолчанию в этом качестве выступит `Marionette.ItemView`.

```
CompositeView = Backbone.Marionette.CompositeView.extend({
 template: "#leaf-branch-template"
});
new CompositeView({
 model: someModel,
 collection: someCollection
});
```

Шаблон композиционной модели

В процессе отображения `CompositeView` на странице, модель будет отрендерена с применением шаблона (`template`), описанного в конфигурации.

```
new MyComp({
  template: "#some-template"
});
```

CompositeView: `itemViewContainer`

По умолчанию, composite view использует тот же самый метод `appendHtml`, который представлен в collection view.

Впрочем, пользы от этого немного, поскольку composite view обыкновенно использует в качестве контейнера для отображения DOM-элемент, в котором развернутся его встроенные item views.

К примеру, если вы проектируете представление табличного вида, и хотите добавлять элементы коллекции внутрь `<tbody>`, вы можете сделать в шаблоне следующее:

```
<script id="row-template" type="text/html">
  <td><%= someData %></td>
  <td><%= moreData %></td>
  <td><%= stuff %></td>
</script>
<script id="table-template" type="text/html">
  <table>
    <thead>
      <tr>
        <th>Some Column</th>
        <th>Another Column</th>
        <th>Still More</th>
      </tr>
    </thead>
    <tbody><!-- сюда мы хотим помещать элементы коллекции --></tbody>
    <tfoot>
      <tr>
        <td colspan="3">some footer information</td>
      </tr>
    </tfoot>
  </table>
</script>
```

Чтобы заставить экземпляры itemView отображаться внутри `<tbody>` имеющейся табличной структуры,
определите атрибут `itemViewContainer` в composite view, примерно так:

```
RowView = Backbone.Marionette.ItemView.extend({
  tagName: "tr",
  template: "#row-template"
});
TableView = Backbone.Marionette.CompositeView.extend({
  itemView: RowView,
  // определяем jQuery-селектор для размещения экземпляров itemView в соответствующем элементе:
  itemViewContainer: "tbody",
  template: "#table-template"
});
```

В качестве альтернативы, вы можете определить `itemViewContainer` как функцию.
Она должна возвращать либо jQuery-селектор в строковом представлении, либо jQuery-объект.

```
TableView = Backbone.Marionette.CompositeView.extend({
  // ...
  itemViewContainer: function(){
    return "#tbody"
  }
});
```

Если использовать функцию, можно поэкспериментировать с возвращаемым значением. Однако следует заметить, что значение будет возвращено только единожды: оно кешируется и все дальнейшие обращения к `itemViewContainer` в течение жизненного цикла представления будут иметь дело уже с этим закешированным значением.

Метод `appendHtml` CompositeView

Иногда конфигурации `itemViewContainer` недостаточно для определения, где должен быть размещен экземпляр itemView.
В этом случае стоит переопределить метод `appendHtml`.

Например:

```
TableView = Backbone.Marionette.CompositeView.extend({
  itemView: RowView,
  template: "#table-template",
  appendHtml: function(collectionView, itemView, index){
    collectionView.$("tbody").append(itemView.el);
  }
});
```

Рекурсия по умолчанию

В обычном режиме отображения, принятом по умолчанию, `CompositeView` предполагает иерархическую рекурсивную структуру. Если сконфигурировать composite view без специфического `itemView`, мы получим однотипное отображение composite view для всех элементов коллекции (чтобы понять, что имеется в виду, можно поэкспериментировать с GridView - прим.пер.).
Если необходимо переопределить данное поведение, нужно задать `itemView` в определении composite view:

```
var ItemView = Backbone.Marionette.ItemView.extend({});
var CompView = Backbone.Marionette.CompositeView.extend({
  itemView: ItemView
});
```

Отображение модели и коллекции

Модель и коллекция composite view заключает в себе следующие предпосылки для повторного отображения:

- Когда происходит событие "reset" коллекции, коллекция будет отображена заново внутри композиции, но не внутри шаблона-обертки (??? - нужен пример)
- Когда в коллекцию добавляется новая модель (и в коллекции порождается событие "add"), только один соответствующий модели элемент будет повторно отображен
- Когда модель удаляется из коллекции (порождается событие "remove"), только один соответствующий модели элемент будет удален из списка отображения

Вы также можете запускать рендеринг вручную:

- Если вы хотите заново отрендерить весь composite view, вызовите метод `.render()`
- Если вы хотите заново отрендерить представление, соответствующее обособленной модели, вызовите метод `.renderModel()`

События и коллбэки

В процессе рендеринга композиции, могут происходить некоторые события. Все они порождаются с помощью функции Marionette.trigger Method, которая пытается вызвать соответствующие методы "on{EventName}", если они определены в базовом представлении.

- событие "composite:model:rendered" (метод `onCompositeModelRendered`) - произойдет после того, как `modelView` отобразится
- событие "composite:collection:rendered" (метод `onCompositeCollectionRendered`) - произойдет после того, как отобразится коллекция моделей
- события "render" / (метод `onRender`) и "composite:rendered" (метод `onCompositeRendered`) - произойдет после того, как все будет отображено

Дополнительно, после того как composite view будет отображен, вызовется метод `onRender`. Вы можете реализовать его в представлении, например, чтобы взаимодействовать с объектом `el` представления, который к этому времени уже будет отображен:

```
Backbone.Marionette.CompositeView.extend({
  onRender: function(){
    // делаем что-то
  }
});
```

Организация UI элементов

Подобно ItemView, можно организовать UI элементы внутри CompositeView, определив их в `UI` хеше. Уточним, что элементы, которые доступны через данный хеш, это те самые элементы, которые напрямую внедряются в страницу в шаблоне CompositeView, а не те, что принадлежат коллекции.

UI элементы будут доступны сразу, как только шаблон composite view отобразится на странице (и прежде чем отрендерится коллекция).

modelEvents и collectionEvents

CompositeViews также поддерживает привязку методов к событиям модели и коллекции в декларативной манере:

```
Marionette.CompositeView.extend({
  modelEvents: {
    "change": "modelChanged"
  },
  collectionEvents: {
    "add": "modelAdded"
  }
});
```

(упоминание о происхождении CompositeView - прим. пер)