

# Put Some Backbone.js in your apps

NCDevCon - 2013

## Introduction

Today, we'll look at the components used to construct a backbone.js application. These include models, collections, views, templates and routers. As we walk through the code each example will build upon the previous one until we have a fully functional app.

## Features

Backbone.js employs a MVC (Model View Controller) pattern. Data management is a key feature of the framework. Data is stored in models and collections and saved via a RESTful backend. For today, we'll use StackMob's JavaScript SDK to easily add a backend to our backbone.js app we are building. We will also use require.js to physically separate our code and then assemble it for production.

## What you'll need for the workshop

Your laptop with the following software

- **HTML Editor** of your choice (Sublime Text, TextMate, etc)
- **Modern Browser** (Chrome is a good choice)

Create your free **StackMob Account**

- <http://bit.ly/ncdevcon2013>

**Node.js** for compiling with require.js (You can download here <http://nodejs.org/download/>)

The node.js stuff can be tricky to install, so don't stress if you have issues. It's a very small part of the session and I will demonstrate on my laptop.

Download the **Workshop files**

<https://github.com/SidneyAllen/backbonejs-stackmob-workshop>

## Contact Me

Sidney Maestre - Platform Evangelist at StackMob

Twitter and Github - SidneyAllen

Email: [sid@stackmob.com](mailto:sid@stackmob.com)

## Agenda

- Module Pattern
- Models
- Collections
- Views
- Templates
- Routers
- Events
- StackMob Datastore
- Data Access Controls
- User Management
- Require.js

## Section 0 - Module Pattern

Open 00-module.html in your browser.

Open your browser Developer Tools

- Chrome or Safari - Right-click or Control-click anywhere in the browser window and select **Inspect Element**

Select the **Console** tab and enter the following commands. Hit the enter key after each command to see the results.

```
app.foo
app.name
app.get()
app.set("StackMob")
app.get()
app
```

## Section 1 - Models

**Open 01-model.html in your browser.**

Open your browser Dev Tools

- Chrome or Safari - Right-click or Control-click anywhere in the browser window and select **Inspect Element**

Select the **Console** tab and enter the following commands. Hit the enter key after each command to see the results.

```
firstWine = new Wine({name : 'Clos Pegase', year : '2008'})  
  
firstWine.get('name');  
  
firstWine.set('name', 'Cakebread');  
  
firstWine.toJSON()
```

## Section 2 - Collections

Open 02-collection.html in your browser.

Open your browser Dev Tools

- Chrome or Safari - Right-click or Control-click anywhere in the browser window and select **Inspect Element**

Select the **Console** tab and enter the following commands. Hit the enter key after each command to see the results.

```
wines = new Wines()
wines.toJSON()
w = new Wine({name : 'Clos Pegase'})
wines.add(w)
wines.toJSON()
```

**Handy tip!** use the up and down arrow keys to see previous commands you typed into the console.

## Section 3 - Home View

### Open 03-home-view.html in your browser.

Open your browser Dev Tools

- Chrome or Safari - Right-click or Control-click anywhere in the browser window and select **Inspect Element**

Select the **Console** tab and enter the following commands. Hit the enter key after each command to see the results.

```
view
view.render()
view.el
view.render().el
```

### Open 03-home-view.html in your HTML Editor.

Add the lines of code in **bold**.

```
HomeView = Backbone.View.extend({
  initialize : function() {
    this.render();
  },
  render: function() {
    this.empty();
    this.$el.append("<h1>Wine Cellar</h1>");
    return this;
  }
});

$(document).ready(function() {
  view = new HomeView();
  $('#body').append(view.el);
});
```

## Section 4 - List View

Open 04-list-view.html in your browser.

You'll see the HomeView displayed. Now let's add a list subview.

Open 04-list-view.html in your HTML Editor.

Add the lines of code in **bold**.

```
ListView = Backbone.View.extend({  
  tagName: 'ul',  
  render: function() {  
    this.$el.empty();  
    this.$el.append("<li>Wine 1</li>");  
    this.$el.append("<li>Wine 2</li>");  
  
    return this;  
  }  
});
```

Inside the HomeView render method

```
render: function() {  
  this.$el.empty();  
  this.$el.append("<h1>Wine Cellar</h1>");  
  
  this.listView = new ListView();  
  this.$el.append(this.listView.render().el);  
  
  return this;  
}
```



## Section 5 - Basic Template

Open **05-basic-template.html** in your browser.

You'll see the HomeView displayed. Now let's render our list with a template.

Open **05-basic-template.html** in your HTML Editor.

Our template looks like this.

```
<script type="text/template" id="listTemplate">
  <li><%= winery %></li>
</script>
```

Add the lines of code in **bold**.

```
ListView = Backbone.View.extend({
  tagName: 'ul',

  initialize: function() {
    this.template = _.template($('#listTemplate').html());
    this.render();
  },

  render: function() {
    this.$el.empty();
    this.$el.append(this.template({winery : "Cakebread"}));

    return this;
  }
});
```

## Section 6 - Collection & Template

**Open 06-collection-template.html in your browser.**

You'll see the HomeView and displayed, but our list is empty.

**Open 06-collection-template.html in your HTML Editor.**

In our **ListView** let's loop over our collection and render using a template

```
render: function() {
  var self = this;

  this.$el.empty();

  wines.each(function(model) {
    self.$el.append(self.template(model.toJSON()));
  });

  return this;
}
```

Let's listen for changes in our collection to trigger the render method.

```
initialize: function() {
  this.template = _.template($('#listTemplate').html());
  this.listenTo(wines, "all", this.render);
  this.render();
},
```

### Open your Developer Tools

Select the **Console** tab and enter the following commands. Hit the enter key after each command to see the results.

```
wines.add({winery : 'Clos Pegase'})
```

## Section 7 - Basic Router

**Open 07-basic-router.html in your browser.**

You'll see the HomeView and a link to Add View. Click the link to display the AddView.

**Open 07-basic-router.html in your HTML Editor.**

I've added **AppRouter** with 2 routes, home and add along with an AddView

```
var AppRouter = Backbone.Router.extend({
  routes:{
    "":"home",
    "add":"add"
  },
  home:function () {
    var view = new HomeView();
    $('body').empty();
    $('body').append(view.el);
  },
  add:function () {
    var view = new AddView();
    $('body').empty();
    $('body').append(view.el);
  }
});
```

At this point everything inside our module is var scoped and private. To kick off our app, we add an initialize method and expose it via a return. Then call the method on document ready.

```
var app = (function($){
  ...
  var initialize = function() {
    wineApp = new AppRouter();
    Backbone.history.start();
  }
  return { initialize : initialize};
}(jQuery));

$(document).ready(function() {
  app.initialize();
});
```

## Section 8 - Adv. Router

**Open 08-adv-router.html in your browser.**

You'll see the HomeView.

**Open 08-adv-router.html in your HTML Editor.**

Cut and paste the new Wines code from line 24 and paste into initialize method. Pass the wines collection to the AppRouter.

```
var initialize = function() {  
    var wines = new Wines([{"winery": "Cakebread"}]);  
  
    wineApp = new AppRouter({collection: wines});  
    Backbone.history.start();  
}
```

Add an initialize method on our AppRouter, then pass the collection into our views.

```
var AppRouter = Backbone.Router.extend({  
    routes:{  
        "":"home",  
        "add":"add"  
    },  
  
    initialize:function(options) {  
        this.collection = options.collection;  
    },  
  
    home:function () {  
        var view = new HomeView({collection: this.collection});  
        $('body').empty();  
        $('body').append(view.el);  
    },  
    add:function () {  
        var view = new AddView({collection: this.collection});  
        $('body').empty();  
        $('body').append(view.el);  
    }  
});
```

Now change the render method in our ListView to use the collection pass to it.

```
render: function() {  
    var self = this;  
  
    this.$el.empty();  
  
    this.collection.each(function(model) {  
        self.$el.append(self.template(model.toJSON()));  
    });  
  
    return this;  
}
```

## Section 9 - Events

**Open 09-events.html in your browser.**

You'll see the HomeView.

**Open 09-events.html in your HTML Editor.**

We need to listen for a click event on the save button and call the save method.

```
var AddView = Backbone.View.extend({
  events: {
    "click #saveBtn": "save"
  },

  initialize: function() {
    this.template = _.template($('#addTemplate').html());
    this.collection = this.options.collection;
    this.router = this.options.router;
    this.render();
  },

  render: function() {
    this.$el.empty();
    this.$el.append(this.template());

    return this;
  },

  save: function(e) {
    e.preventDefault();

    var wine = new Wine({winery:$('#winery').val() });
    this.collection.add(wine);

    this.router.navigate('#',{trigger: true});

    return this;
  }
});
```

## Section 10 - StackMob

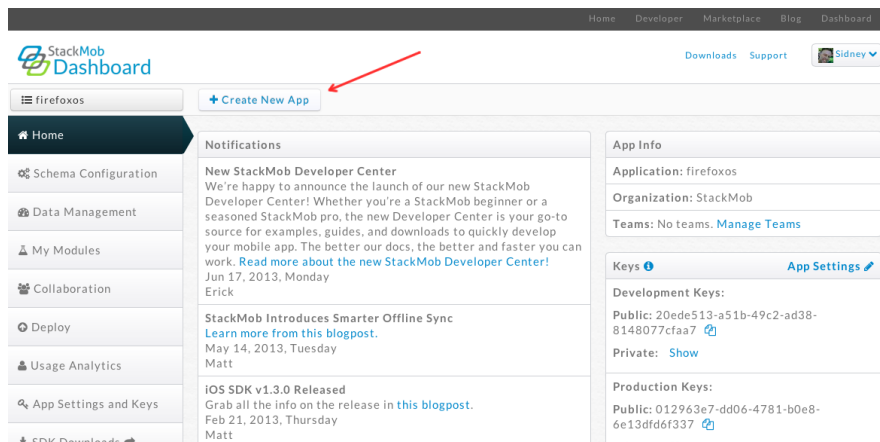
### Open stackmob-init.js in your HTML Editor.

You'll see a few lines of code that initializes the StackMob SDK.

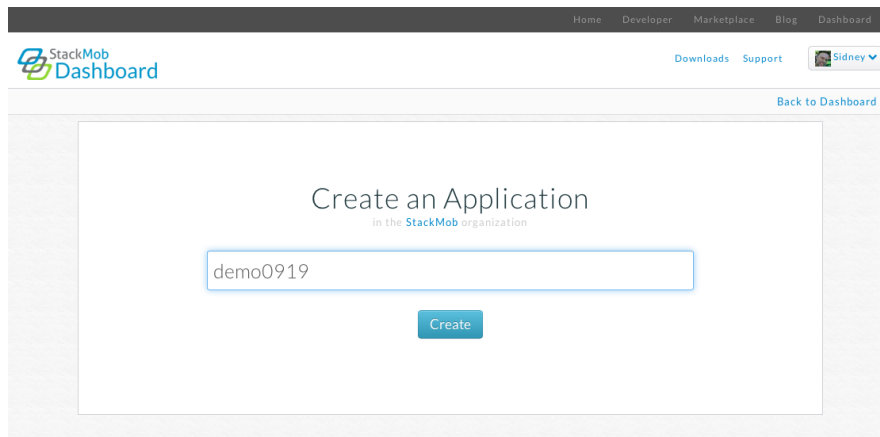
```
StackMob.init({  
    publicKey: "cb28c0a3-20b9-4a3c-9691-aa1b86ee558f",  
    apiVersion: 0  
});
```

### Open your browser to dashboard.stackmob.com

Click on Create New App.



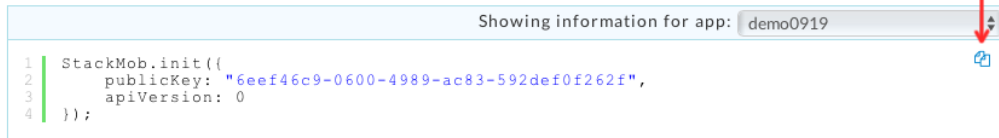
Enter the name of your new App



Select JavaScript as your platform and copy the StackMob.init method in step 2.

## Set up Starter Project

1. [Download JavaScript Starter Project v0.9.2](#)
2. Edit `app.js` and add your public key to the `StackMob.init(..)` block:



```
Showing information for app: demo0919
1 StackMob.init({
2   publicKey: "6eef46c9-0600-4989-ac83-592def0f262f",
3   apiVersion: 0
4 });
```

Paste the StackMob.init method into stackmob-init.js file.



## Open 10-stackmob.html in your browser.

You'll see the HomeView, but data is not persisted anywhere

## Open 10-stackmob.html in your HTML Editor.

Change your Model and Collection in **bold**.

```
var Wine = StackMob.Model.extend({  
  schemaName : "wine"  
});  
  
var Wines = StackMob.Collection.extend({  
  model: Wine  
});
```

Fetch your wines at the start of your app.

```
var initialize = function() {  
  var wines = new Wines();  
  wines.fetch({async:true});  
  
  wineApp = new AppRouter({collection : wines});  
  Backbone.history.start();  
}
```

In your save method, wrap your collection.add and router.navigate inside a success callback from the model.

```
wine.create({  
  success: function(model) {  
    self.collection.add(model);  
    self.router.navigate('#',{trigger: true});  
  }  
});
```

Refresh 10-stackmob.html in your browser, now try to add a new wine.

Go back to **StackMob Dashboard** and select the **Data Management** to see if your data was saved.