How do I add a new view ? (on the frontend)

Adding a new view to the frontend is probably one of the first things you'll want to do when building anything new into Ushahidi 3.x. By adding a view, I really mean adding a new URL on the frontend of the application.

Technically this means adding a route, controller and view.. so lets walk through the process:

The (Backbone) view and template

All the HTML in the Ushahidi 3.x UI is rendered by a backbone view. We're using backbone marionette's layouts and regions to nest views within each other. For most purposes you'll only need to modify the HTML that goes into the main region, that is the contents of div#main-region shown below.

Base app view html (AppLayout.html)

```
<div class="content-wrapper  js-content-wrapper">
   <h1 class="visually-hidden">Main Section</h1>
   <section id="main" role="main">
      <div id="header-region"></div>
      <div id="main-region"></div>
   </section>
   <aside id="workspace-panel" class="workspace-panel" role="complementary">
   </aside>
</div>
<div id="footer-region"></div>
```

Creating a new view

Lets create a simple ExampleView. We'll add this to the media in the UshahidiUI module, so the path ends up like this: modules/Ushahidi/media/js/app/views/ExampleView.js

The example code below creates a super simple view, that just loads templates/Example.html and compiles it with Handlebars.

modules/Ushahidi/media/js/app/views/ExampleView.js

```
// Define required modules: Marionette, handlebars, and our template Example.html
// Add text! before the requirement loads the file as text, rather than evaluating it as JS.
define([ 'marionette', 'handlebars', 'text!templates/Example.html'],
   function( Marionette, Handlebars, exampleTemplate)
   {
  // Create an Item view
      return Marionette.ItemView.extend( {
  // Override template to use our template (loaded above)
         template: Handlebars.compile(exampleTemplate)
      });
   });
```

The HTML for this can be anything we need really.. here's a quick example

modules/UshahidiUI/media/js/app/templates/Example.html

```
<div class="body-wrapper">
    <div class="row">
        <p>My Awesome Example Page!</p>
    </div> <!-- end .row -->
</div> <!-- end .body-wrapper -->
```

Displaying the view

Now lets add an example controller and route to display our new view. Most of the existing controllers are a bit more complicated.. using layouts, and multiple views.. but we just want to get something on the screen.

Add this to controllers/Controller.js

modules/UshahidiUI/media/js/app/controllers/Controller.js

```
// At the top of Controller.js define the file path and add the class name to the function
define([..., 'views/ExampleView'],
 function(..., ExampleView)

Backbone.Marionette.Controller.extend(
{
 ...
 example : function()
 {
     this.layout.mainRegion.show(new ExampleView());
 }
 ...
}
```

Then add the following to routes/AppRouter.js

modules/UshahidiUI/media/js/app/routers/AppRouter.js

```
Marionette.AppRouter.extend(
{
    appRoutes :
    {
        '' : 'index',
        'views/list' : 'viewsList',
        'views/map' : 'viewsMap',
        'posts/:id' : 'postDetail',
  // Add your new view before the *path rule
  'example' : 'example',
  '*path' : 'index'
 }
});
```

Now if you load your V3 site, and got to #example you should see your new view! \o/

Extra: how routing works..

When you hit a particular url, say "posts/1", this is matched to a route, in this case 'posts/:id'. The router directs that request to a matching function on the Controller class.

routers/AppRouter.js

```
Marionette.AppRouter.extend(
{
    appRoutes :
    {
        '' : 'index',
        'views/list' : 'viewsList',
        'views/map' : 'viewsMap',
        'posts/:id' : 'postDetail',
        '*path' : 'index'
    }
});
```

Extract of controllers/Controller.js

```
Backbone.Marionette.Controller.extend(
{
    initialize : function()
    {
    ...
    },
    postDetail : function(id)
    {
    ...
    }
});
```

In our example: the URL '/posts/1' is matched to the 'posts/:id' route. This triggers a call to Controller.postDetail(1)