Building Movie-Finder Mobile App in Sencha Architect 2.x

Author: Bilal Soylu

September 2012, NCDevCon 2012

Based on Sencha Architect 2.0 documentation

Conventions

- Aliases are lower case
- Class names: Camel case starting with upper case
- Save often!
- Ids are: Camel case starting with lower

Pre-Requisites:

- You need an API key from Rotten Tomatoes website (http://developer.rottentomatoes.com/)

Application

Create a new Sencha Touch Application. In Config, set the following: - **name:** MovieFinder. This will become the name space for the app.

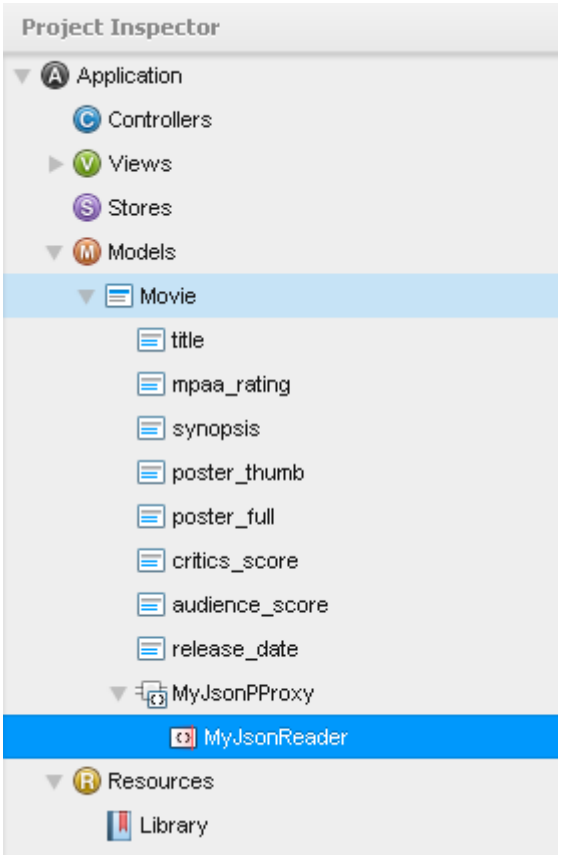
Save application to a destination.

The model

Add a Model, and set its **userClassName** config to `Movie`
Add 8 fields to the model and name (mapping,format) them as follows: (CSV list : title,mpaa_rating,synopsis,poster_thumb,poster_full,critics_score,audience_score,release_date)

1. `title`
2. `mpaa_rating`
3. `synopsis`
4. `poster_thumb (posters.thumbnail)`
5. `poster_full (posters.detailed)`
6. `critics_score (ratings.critics_score/sortDir=DESC/sortType=asInt/type=int)`
7. `audience_score (ratings.audience_score/type=int)`
8. `release_date (release_dates.theater, Y-m-d)`

Add ***JSONP Proxy*** / Add ***JSON Reader*** as child to JSONP Proxy:



In MyJsonPProxy configs:

1. **url:**

http://api.rottentomatoes.com/api/public/v1.0/lists/movies/box_office.json?apikey=[ADD YOUR API KEY HERE]

In MyJsonReader configs:

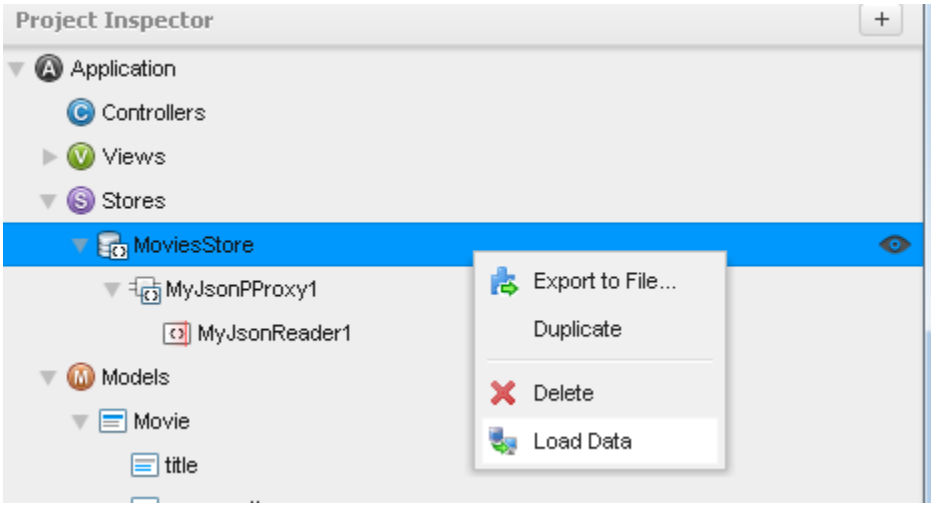
1. rootProperty: movies

The store

Add a ***JsonPStore*** and set its configs as follow:

- **userClassName:** `MoviesStore`
- **storeId:** `MoviesStore`
- **autoLoad:** true
- **model:** `Movie`. (Note that this associates the store with the model that was just added.)
- **url**: http://api.rottentomatoes.com/api/public/v1.0/lists/movies/box_office.json?apikey=[ADD YOUR API KEY HERE]
- **Sorters:** click (+) sign to add sorter

Select `MyJsonReader1` in the Inspector and set its **rootProperty** config to `movies`. Theoretically you should not need url, but adding it at store level allows architect to load sample data. See the Eye icon after load.



In ***MySorter*** config:

- direction: DESC
- property: critics_score

The Views

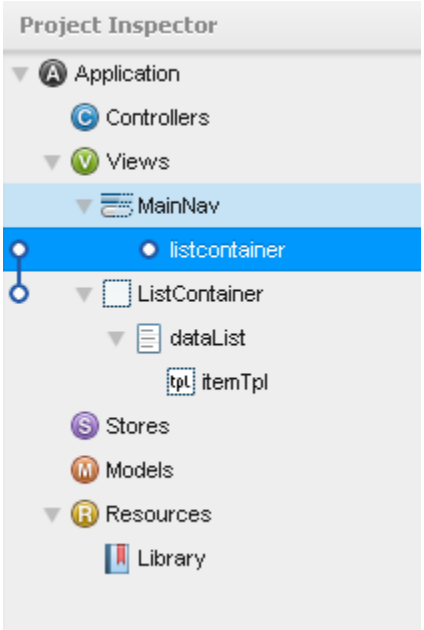
Main navigation view

Add a Navigation view to the project. In Config, set the following:

- **userAlias:** mainnav
- **userClassName:** MainNav

List view

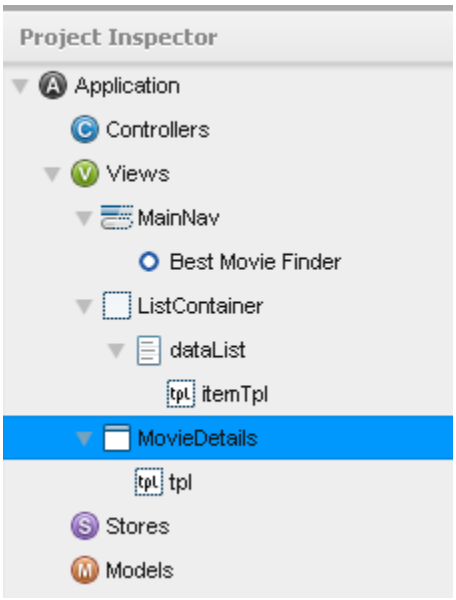
1. Add a ***Container*** to the project as a top-level component. Set the following configs:
2. **userAlias:** `listcontainer`
3. **userClassName:** `ListContainer`
4. **layout:** `fit`
5. Add a ***List*** as a child of the container (`ListContainer`) you just created. Set the following configs for List:
6. **id:** `dataList`
7. **itemTpl:** <section class="movieListItem"><img src="{poster_thumb}"><h1>{title}</h1> Critics: {critics_score}% Audience: {audience_score}% </section>
8. **store:** MoviesStore
9. **onItemDisclosure:** true
10. **loadingText:** Yo..hold on!
11. Create a linked instance of the container (`ListContainer`) by dragging it onto the MainNav view in the Inspector and selecting the Link option from the Copy Component dialog box that appears. The Inspector shows the container under MainNav as follows:



1. Select the linked instance of the container (`listcontainer`) under MainNav and set the following configs:
2. **title:** `Movie Finder`

Detail view

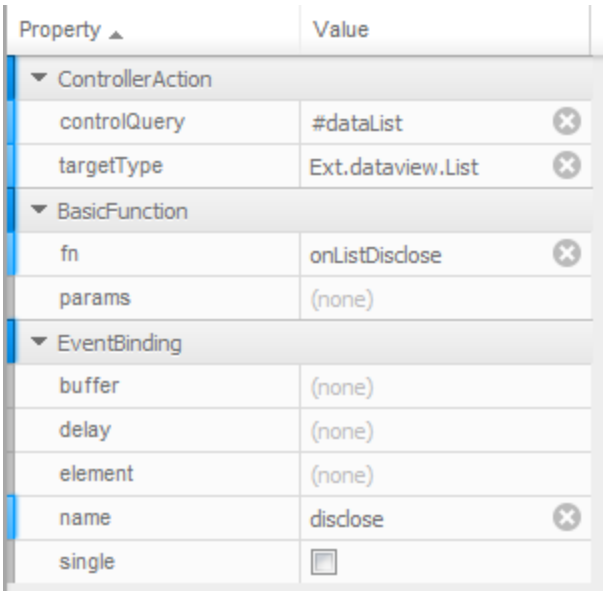
1. Add a ***Panel*** as a top-level component and set the following configs:
2. **userAlias:** `moviedetails`
3. **userClassName:** `MovieDetails`
4. **layout:** vbox
5. **scrollable:** `true`
6. **ui:** `light`
7. **tpl (click on plus to expand, then click on tpl node)**: <img src="{poster_full}"> {synopsis} Release Date: {release_date:date("M d Y")} Critics: {critics_score}% Audience: {audience_score}%



The controller

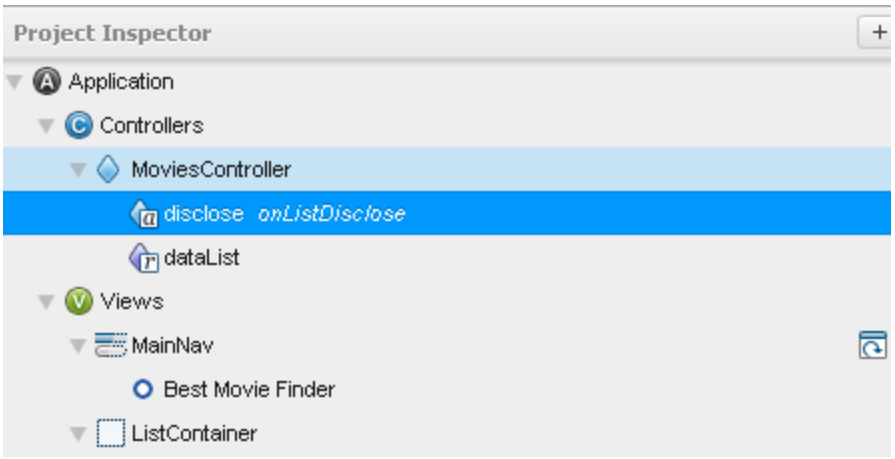
Add a controller and set its **userClassName** to `MoviesController`. Then add a launch function (click plus in Config), then, double-click it in the Inspector to open the code editor, and insert the following code: See the comments for an explanation of what the code does.

Next, add four controller references. In Config, find the References property. Click the add button on the right under Value ("+") to add the first controller reference. When prompted name it `dataList` and set its selector to `#dataList` (this is Id based query, leaving the pound (#) off makes it an alias based query).
.
Add a ***controller action*** to the controller and set its configs as follows:



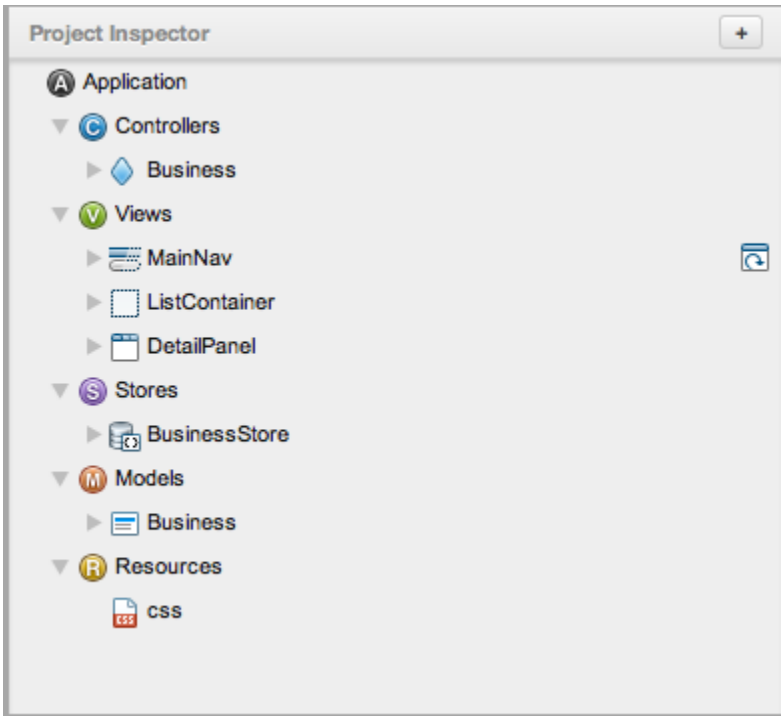
Double-click the controller action in the Inspector, and insert the following code into the code editor:

```
list.up('mainnav').push({
            xtype: 'moviedetails',
            data: record.data,
            title: record.get('title')
        });
```



Add a style sheet

To improve the look of the app, giving it an attractive grey theme, you need to add a style sheet. To do this, in the Inspector, click the add button ("+") and choose Resource-->CSS Resource. The Resources node will display a new css resource, as shown here:



Select it there and set its **url** config to `styles.css`.
Download and unzip the styles.css. Place `styles.css` file in the `MovieFinder` folder you set up at the beginning of the project to store your app. Optionally, you can create a styles directory within the folder

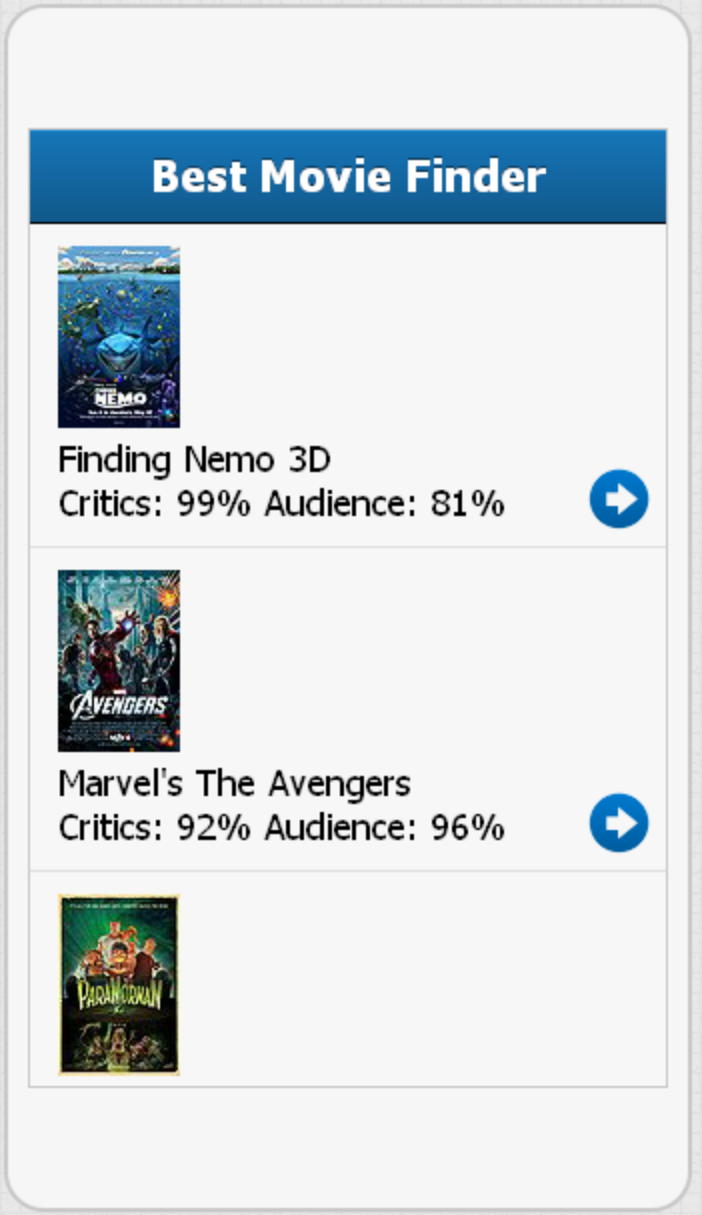
and set the url config to `styles/styles.css`.

Have a look at your hard work

View the application in the Inspector and be sure it now has the following:

- Models: Movie
- Store: MoviesStore
- Views: ListContainer, MovieDetails

Now save your project one last time. Point a web server to the application's location and open the app.html file for it. Here's what it should look like.



You can also deploy the project to a hosted location and view it there. You can then navigate using your browser to view the application. Chrome should work fine, or view it through a WebKit browser on a mobile device.