Devgirl's Weblog
Web, Cloud, Mobile Application Development

- PhoneGap Plugin Dev't for Android Tutorial
- PhoneGap Push Notifications Tutorial

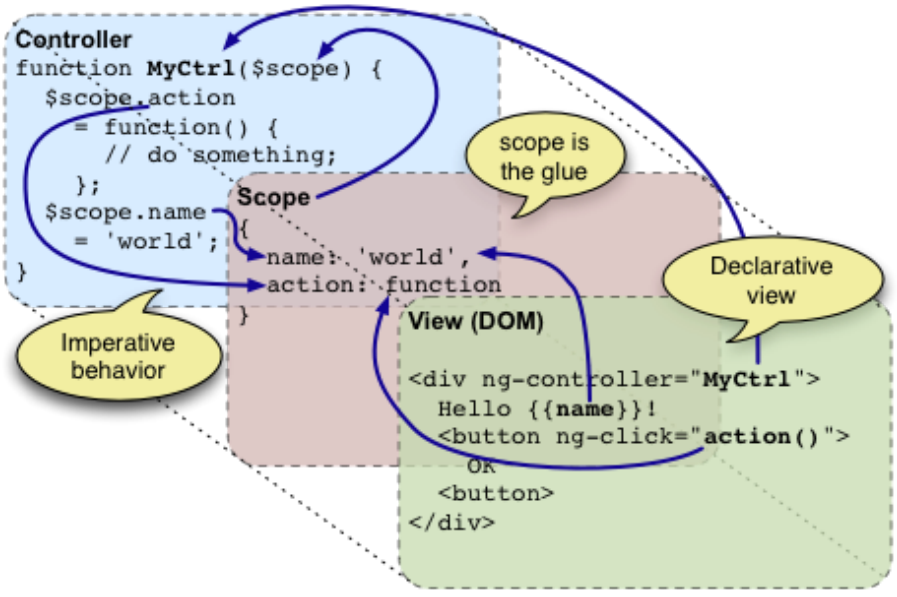
Fun with AngularJS!

March 21, 2013By Holly Schinsky25 Comments

Recently AngularJS, an MVC style JavaScript framework from Google, has been gaining more momentum. I've been curious about it for awhile and when I read this article recently, I decided it was finally time to give it a whirl. I was pleasantly surprised with what I found. I first watched these two videos by John Lindquist which I highly recommend as a starting point to get your feet wet. They are short and totally worth the time:

1. Simple ToDo's App
2. Simple Twitter Search App

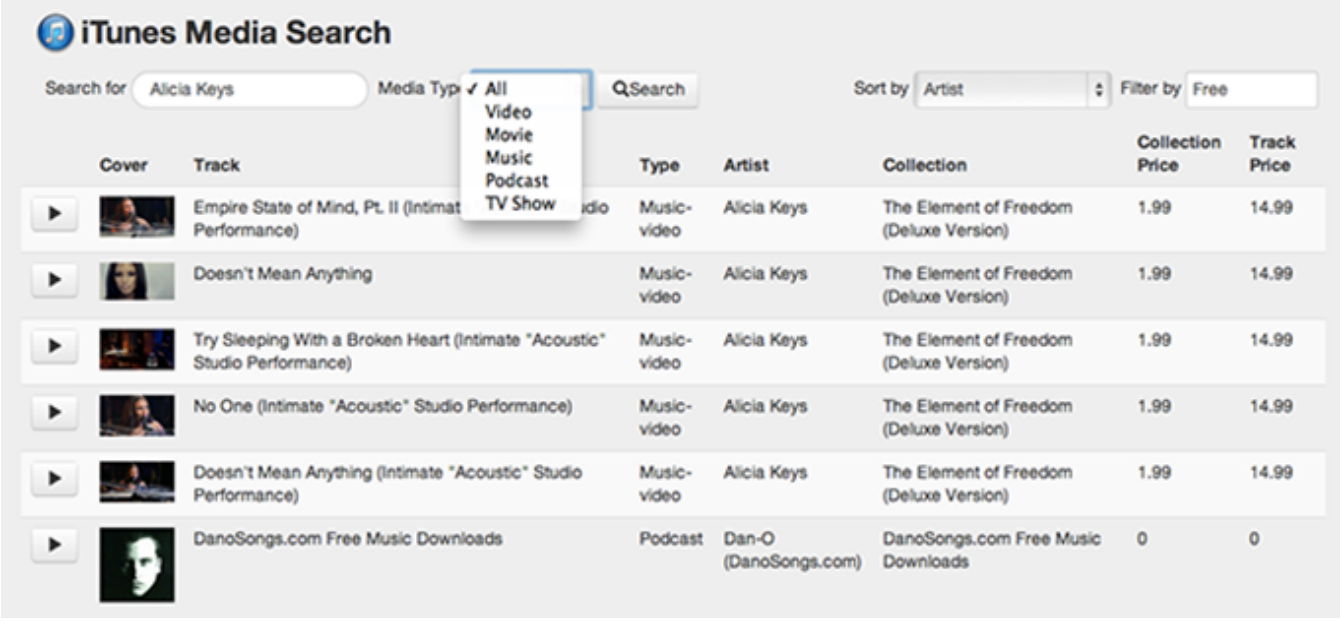
Part of the reason I personally may have been hooked initially could be due to my Adobe Flex background. There's a reminiscent feel with AngularJS to Flex in the two-way binding, HTML extension etc so I could see how a lot of Flex developers might feel at home with this framework. The framework is built on the idea that declarative programming should be used for building the UI and imperative programming for your business logic. With AngularJS you can do quite a bit with a small amount of code due to the built-in features that will be explained further in this post. It also relies on dependency injection throughout as well as the separation of the model/view/controller. Here's a nice illustration from the AngularJS site that helps explain more:



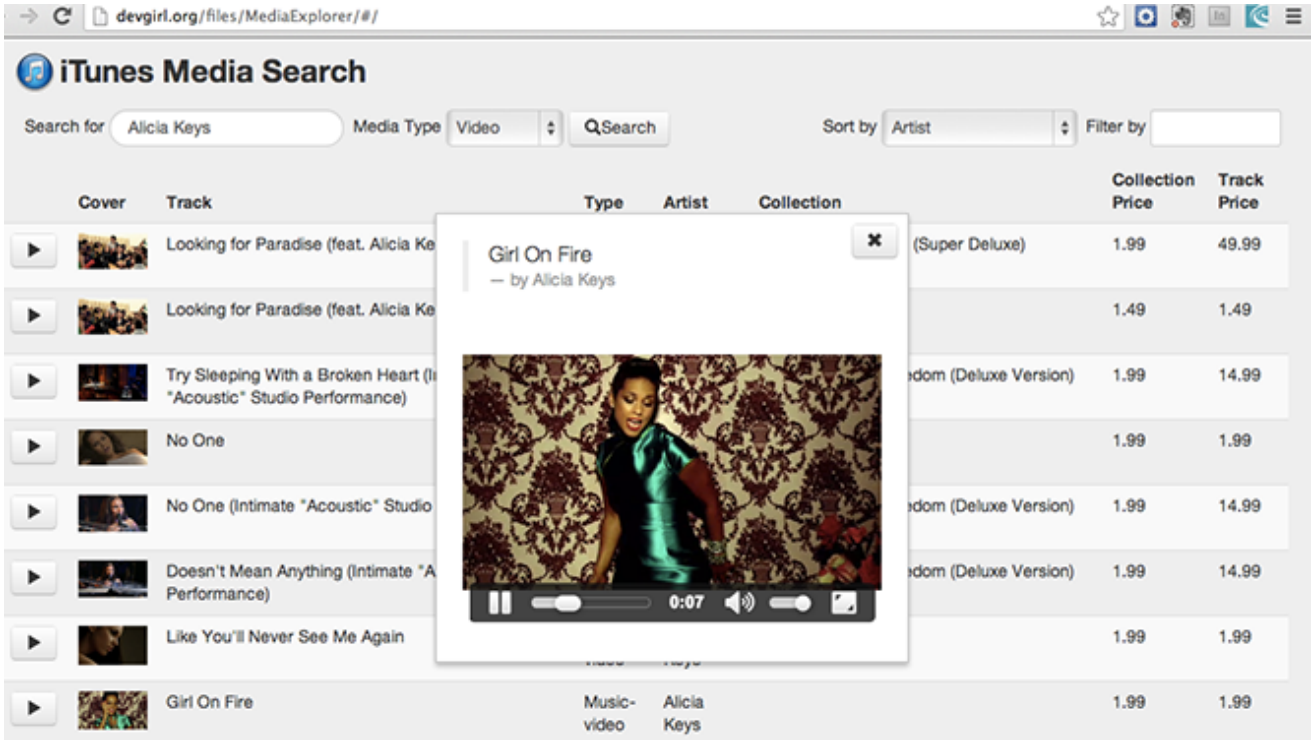
> My intention is not to incite a frameworks war here. I know there are a myriad of JavaScript frameworks available that people are passionate about. I'm simply sharing my thoughts and observations ;).

After watching the videos above, I moved on to the tutorial, which was very easy to follow and I thought it was especially cool how they introduced testing your AngularJS app while building it using Jasmine and Testacular. I had previously only heard of those but got a much better idea of how to use them from the tutorial. It's obvious they kept testing in mind while creating this framework.

I then decided to build myself a web application with what I had learned. I'm a big music fan, always downloading from iTunes for my running playlists etc, so I thought I'd build a simple web app that interfaced with the iTunes Search API. The resulting application is here. You can grab the project from my github account here. This was my first newbie attempt so I'm open to critique and feedback on how I could make it better. I noticed there are often multiple ways to do the same thing using AngularJS, so I'm still in the learning process of when to use what. Overall though I was impressed by how fast I was able to make it all work and how much the framework provides for you. From my Media Explorer application you can enter a search term for anything media related (artist, song name or part, video etc). Once you get the results back, try out the sort and filter to see the returned results sort and filter accordingly. The filter field will filter results as you type into it. This is all built into the array filtering mechanism in AngularJS. Currently the max number of results returned defaults to 50.



Hitting the play button on the far left of any row will bring up an overlay to play the sample media as shown below:

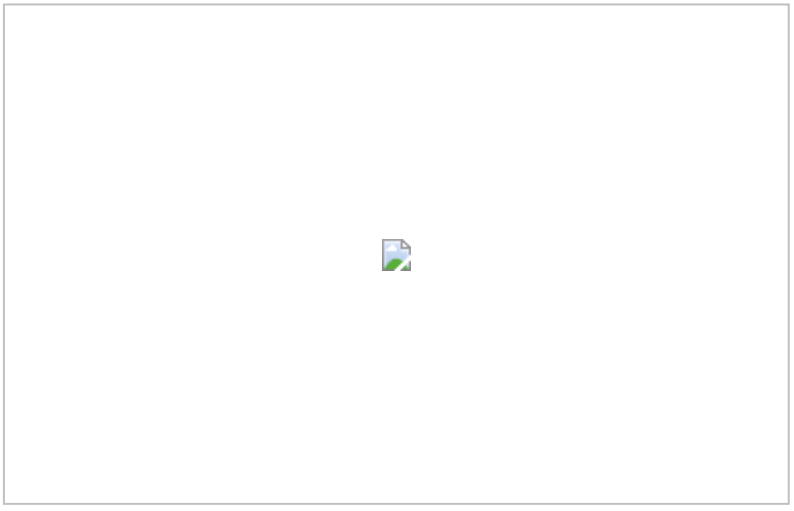


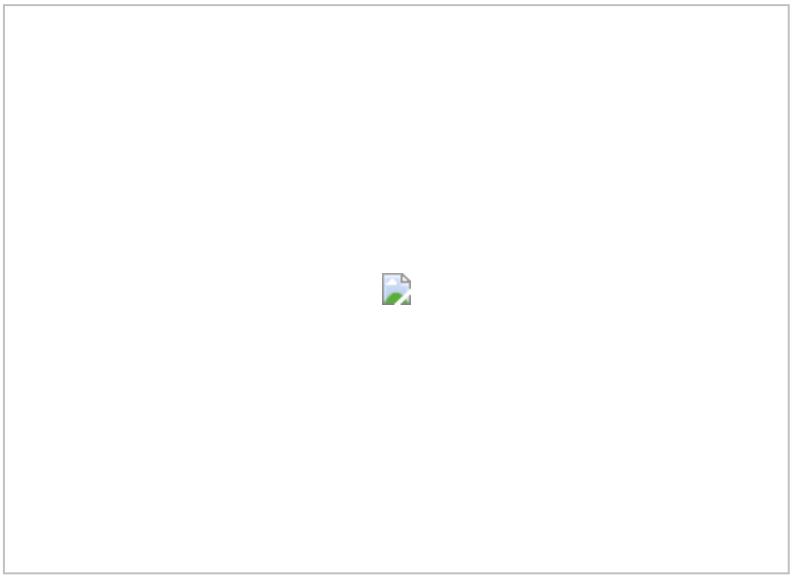
> I used Twitter Bootstrap in my MediaExplorer, but subsequently found AngularUI and Angular Bootstrap which you should also check out for help with your UI specific to Angular.

In the next sections I'll cover some of the main AngularJS concepts that seem important to understand.

Data Binding

One of the first things I was most excited to see with AngularJS is that it has built-in support for two-way data binding similar to Flex (yay)! This is a big deal for data-driven applications in particular and can save you a TON of time coding. Below are two images from the AngularJS site that I think illustrate this point so well:





If you want to try a very basic example of the two-way data binding in action, I found this live sample here. If you look at the source code you'll see how simple it is:

```
<div>
    Binding the value in the text field to the model name.
    <br/><br/>
    <input type="text" ng-model="name" placeholder="Enter a name here"/>
    <span ng-show="name"><h4>Hello {{name}}></h4></span>
</div>
```

As you type into the input field it will show up next to the 'Hello' string since the name field is referenced there and is also the bound model object on the HTML input (with ng-model=" name").

Model

The model is simply a plain old JavaScript object, does not use getter/setter methods or have any special framework-specific needs. Any changes are immediately reflected in the view via the two-way binding feature. There's a concept of scope in the AngularJS framework, where all model objects stem from. I recommend seeing the AngularJS Scope documentation for more information on this topic. Typically your model objects are initialized in your controller code with syntax like:

```
$scope.searchTerm = "Alicia Keys";
```

But in the HTML template, that model variable would be referenced in curly braces such as: {{searchTerm}} without the $scope prefix.

View

The view is the rendered HTML that contains the transformed DOM resulting from a combination of the HTML template along with the AngularJS model and controller. The AngularJS View is aware of the controller and model, and accesses it through annotated HTML directives specific to AngularJS. For example, an ng-click AngularJS 'directive' can be applied to a button element to call a specific method in the controller when clicked. More on directives below…

Controller

The controller is used to provide the business logic behind the view and construct and value the model. The goal is to not manipulate the DOM at all in the controller, which is

something to get used to if you've been using other frameworks.

Filters

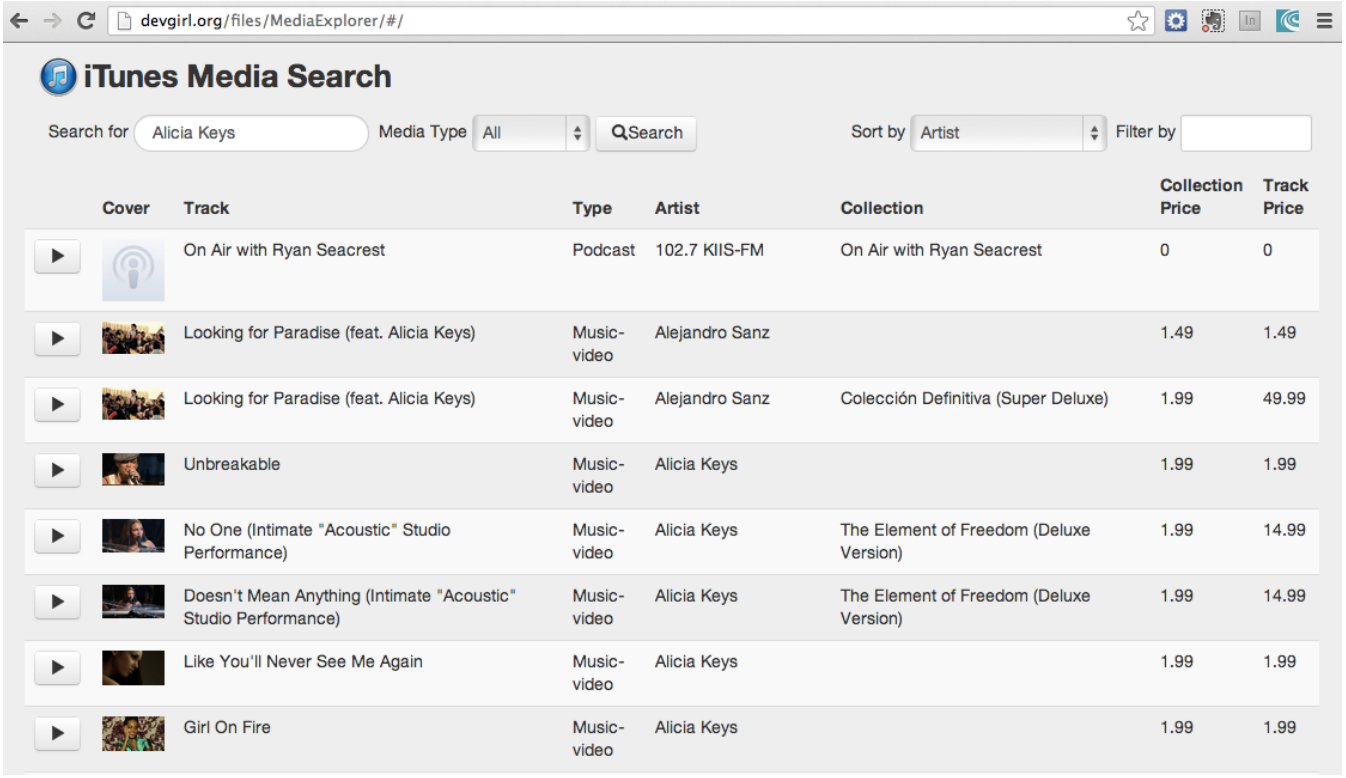
AngularJS filters are used to modify data in some way, similar to formatters in Flex. For instance, a built-in filter called uppercase would look like this when applied to the name string from the model in order to uppercase it upon display.

```
<td>{{name|uppercase}}</td>
```

You're welcome to write your own filters as well. I included a custom one for fun in my application called capitalize that will simply capitalize the first letter of a string.

```
mediaApp.filter('capitalize', function() {
    return function(input, scope) {
        if (input!=null)
            return input.substring(0,1).toUpperCase()+input.substring(1);
    }
});
```

You can see it applied here:



There's also an array filtering feature which is really cool, and you can see an example of it in the Media Explorer app. When you specify a filter string on an array in the Angular ng-repeat directive it will automatically filter the matching results into a new array.

For instance, here's the code from the Media Explorer that specifies the filterTerm (the model object bound to the filter input HTML element) to your results list as you type (automatically applied do to the two-way data binding on the model data):

```
<label>Filter by
    <input type="text" ng-model="filterTerm" class="input-small"/>
</label>
...
<tr ng-repeat="item in mediaResults | filter:filterTerm | orderBy:sortProp'
    <td><button id="playBtn" class="btn" ng-click="playVideo(item)"><i cl
```

```
        <td><a href="{{item.previewUrl}}"><img ng-src="{{item.artworkUrl60}}"
        <td>{{(item.trackName != null) item.trackName || item.collectionName}
        <td>{{item.kind | capitalize}}</td>
        <td>{{item.artistName}}</td>
        <td>{{item.collectionName}}</td>
        <td>{{item.trackPrice}}</td>
        <td>{{item.collectionPrice}}</td>
        ...
```

Directives

Directives are ways to transform the DOM through extended HTML. The add additional behavior. If you happen to have done Flex, it's similar to adding a property like itemRenderer to an MXML component, or specifying click to call a function. In AngularJS there are a bunch of built-in ones you can use, most with a prefix of 'ng' (ng-click, ng-show, ng-change, ng-app etc). They're actually defined with camelCase in the JavaScript, but applied with a dash to the HTML. You can also define your own directives and apply them in a similar fashion. In my Media Explorer application I wrote a couple of directives you could see for more examples. I used them to manipulate DOM elements since it's not good practice to do so in the controller code. One called videoLoader is used to play and stop the video in the modal when the play button is clicked:

```
mediaApp.directive('videoLoader', function(){
    return function (scope, element, attrs){
        console.log(scope.url);
        scope.$watch("url",  function(newValue, oldValue){ //watching the s
            element[0].children[0].attributes[3].value=newValue; //set the
            element[0].load();
            element[0].play();
        }, true);
        scope.$watch("showFlag",  function(newValue, oldValue){
            if (!newValue) // if the showFlag is false, stop playing the v
                element[0].pause();
        }, true);
    }
});
```

Then to apply this videoLoader directive to my video control, I add it as an attribute with a dash, such as:

```
<video id="vid" width="320" height="240" video-loader="item.previewUrl" aut
    {{url}}
    <source id="vidsrc" ng-src="url" type="video/mp4">
</video>
```

Services

Services are provided in AngularJS to help you do common task. For instance, there's a built-in $http service that gives you access to make http or JSON requests etc, there's also a $location service that gives you access to the browser location URL and helps you parse it etc. There are a bunch of others built-in and once again you can also write your own. You may write your own in the case where there's something you know you're going to use more than once in your application. In my application I did write a service just to try it out and use it to call the iTunes API with the $resource service, which allows you to interact with RESTful services and provides higher level behavior over the $http service (but is dependent on the $http service). My service looks like this:

```
mediaApp.factory('MediaService', function($resource){
    return $resource('https://itunes.apple.com/:action',
        {action: "search", callback: 'JSON_CALLBACK'},
        {get:  {method: 'JSONP'}
    });
});
```

Routes

You typically configure routes in your main or app.js file for an AngularJS app. For instance, mine is very basic since I'm not using multiple views and looks like this:

```
mediaApp.config(['$routeProvider', function($routeProvider) {
    $routeProvider.when('/', {templateUrl: 'views/media-list.html',   contr
}]);
```

You can specify multiple paths for different URLs as well as a fallback. More on route configuration can be found here.

Quick Start

You can use the AngularJS seed project, or if you're familiar with Yeoman, there's a whole option to generate an AngularJS-based project as well as additional commands that can be run to add new routes, views, controllers etc. It's actually really cool and easy to use, check out this article for an awesome walkthrough of it. It will even generate your stubbed out tests for testing with Jasmine / Testacular. They can't make it much easier for us than that :)! I didn't discover this until after I created my application so my project structure doesn't follow the Yeoman structure, but instead mirrors the AngularJS seed project structure. There are also seed projects available for using AngularJS on the front-end and Express+Node on the backend here, as well as AngularJS on the front-end, and Socket.io+Express+Node here if that fits your needs.

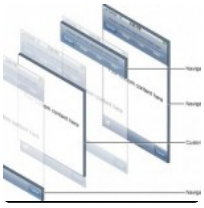
I've only scratched the surface of what this framework is about. Below is a list of resources that can be used to follow-up with more in-depth information for those interested. The AngularJS documentation is good and the community support as well from my experience. Next I plan to turn my Media Explorer web application into a mobile application using PhoneGap and will post it here so check back soon!

Useful Resources

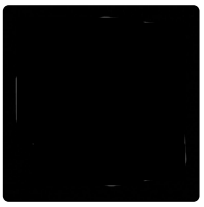
- 5 Awesome AngularJS Features
- @briantford on Building huuuuuuge Angular apps
- AngularJS Batarang - Chrome Extension for Debugging
- AngularJS Forum
- AngularJS on Google+
- AngularJS Cheat Sheet
- AngularJS Docs
- AngularJS Blog
- @briantford on Brian Ford on AngularJS with PhoneGap
- @simpulton on The AngularJS Scope LifeCycle
- AngularJS for jQuery Developers
- @simpulton blog - lots of useful articles on AngularJS
- Dean Sofer - AngularJS Tips and Tricks

Related Posts

Animating with AngularJS



Introducing BackStack.js - View Navigation JavaScript Framework



Three hooks your Cordova/PhoneGap project needs

Totally Rad Topcoat!

My Development Workflow for PhoneGap iOS Apps

Filed in: AngularJS Tags: angularJS • angularJS itunes • angularJS overview • angularjs phonegap • angularJS sample

About the Author (Author Profile)

Subscribe

If you enjoyed this article, subscribe to receive more just like it.


Comments (25)

Trackback URL | Comments RSS Feed

1.  Gavin Foley (@GFoley83) says:
   March 22, 2013 at 9:20 pm

   Nice article and demo Holly. Looking forward to seeing how well your AngularJS demo plays with PhoneGap.

   Reply

   -  Joao Grassi says:
     November 12, 2013 at 1:27 pm

     Great article! Also waiting to see AngularJs with phonegap. Thanks!

     Reply

2.  browniefed says:
   March 23, 2013 at 9:57 am

Wow this is pretty amazing stuff. Was looking to get into angular w/ node and this was the perfect intro. Thank you.

Reply

3.  Cody says:
March 23, 2013 at 4:21 pm

Hi Holly,

Thanks for the article. Was very handy to send around to all my coworkers wanting to get to know AngularJS more.

On a side note, I saw in the GitHub repo that you are still using PhpStorm 4.*. You should really upgrade! It's got a ton more features now :).

Reply

-  Holly Schinsky says:
March 25, 2013 at 11:36 am

  Thanks Cody, I'm so glad it was helpful :)! I will definitely upgrade my PHPStorm ;), thanks for the heads up! Holly

  Reply

4.  tim says:
March 23, 2013 at 7:06 pm

check out John Lindquist's awesome http://egghead.io/ for more screencasts on AngularJS.

Reply

5.  Ciro Nunes says:
March 24, 2013 at 2:01 pm

Wow! You covered a lot of topics in few words!
Pretty good stuff, congratz!

PS.: The image about MVC in AngularJS is not from @PaschalPrecht, but from official docs (http://docs.angularjs.org/guide/concepts)

Reply

-  Holly Schinsky says:
March 25, 2013 at 3:37 pm

  Hi Ciro, thanks!! I didn't realize that image was originally from the angularjs site, so thanks! Want to give proper credit :). Updated the link now…

  Reply

6.  Rawley says:
April 4, 2013 at 2:03 am

Yeah, I definitely know what you mean about Angular feeling like Flex. The two way binding really blew me away. It doesn't surprise me that this library has taken off so quickly. Fairly complex apps can be whipped up in minutes.

Reply

7.  Michiel says:
April 12, 2013 at 4:15 pm

As a Flex developer I am pretty excited about Angular and the Model->View data binding it provides. However, what I miss in Angular so far is Model->Model binding. In Flex I would use:

DataObject and DataProcessor are hypothetical classes. DataProcessor would have two properties: inData and outData. Every change to inData would be processed and the result written to outData.

How can I use Angular to accomplish a similar structure with data binding between models?

Reply

-  Michiel says:
April 17, 2013 at 12:35 am

Here is how I solved data binding between models.

In order for data objects (or: models) to be "bindable", they have to be added to a scope ($rootScope if it concerns application wide data). Next, the $watch function of that scope can be used for the binding. A change to the "bindable" object will now execute the desired function (e.g. a setter on another data object).

In my example I would add:

```
$rootScope.do = … /* assign DataObject */
$rootScope.$watch( 'do.data', function(newValue, oldValue) {
dp.processData(do.data); });
```

to the run function (initializer) of the main module.

Too bad that only scopes have this $watch function. If services had a similar function, there would be no need to "pollute" scopes with data models.

Reply

  -  Michiel says:
November 13, 2013 at 5:11 am

Actually, I found a solution for that as well. Services can be scopes at

the same time.

```
angular.module( 'my-app' ).
factory( 'myService' , [ '$rootScope' , function($rootScope) {
var scope = $rootScope.$new();
scope.blabla = addStuff();
return scope;
}]);
```

This way, the service "myService" will have a $watch() function for easy data binding.

Reply

8.  Stan says:
April 23, 2013 at 3:39 pm

For the case you want to start with generated app using Angular with Node and MongoDB - https://npmjs.org/package/amigo - will generates example CRUD app

Reply

9.  Anish Sukumaran says:
May 10, 2013 at 7:46 am

Hello,

My name is Anish and I am an Author Relationship Executive at Packt Publishing. Packt is a rapidly growing, dedicated IT book Publishing firm and has rolled out more than thousand books on various titles till date.

Packt is now planning to publish a book titled as 'AngularJS Web development Cookbook ' which would be a 300 page book and is in the process of seeking potential authors to work on it. Do let me know if anyone is interested in this project or if you have any queries, I will be happy to answer them.

Kind regards,
Anish Sukumaran
Author Relationship Executive
PACKT Publishing
http://www.packtpub.com
MSN: anishs@packtpub.com

Reply

10.  Chris Cotton says:
June 1, 2013 at 3:56 am

Definitely one of the best angular posts I have read (and I have read many). Thanks!

Reply

11.  Vishal says:

June 5, 2013 at 8:53 am

How we can access the models in multiple pages?

Reply

12.  Freddy May says:
June 21, 2013 at 1:18 am

Holly - there is no license file in the Github repo for your MediaApp. Am I able to get my dirty hands on it and hack it around / use it etc?

Thanks - Freddy

Reply

13.  Guillaume Balaine says:
June 24, 2013 at 4:49 pm

Took me about 20 minutes to remake the whole thing on my own.
I guess it's easier to code with a goal, nice example there miss.

Reply

14. Marco says:
July 15, 2013 at 8:45 pm

Hi Holly, thanks a lot, it really helped me to do my little application.
i'm working with AngularJS almost 2 month and building apps.

i have a question about your iTunes Media Search Application.

```
mediaApp.factory('MediaService', function($resource){
return $resource('https://itunes.apple.com/:action',
{action: "search", callback: 'JSON_CALLBACK'},
{get: {method: 'JSONP'}
});
});
```

i changed your code to like that:

```
mediaApp.factory('MediaService', function($resource){
return $resource('https://gdata.youtube.com/feeds/api/:action',
{action: "videos", q:'adele', label:'music', alt:'json', callback:
'JSON_CALLBACK'},
{get: {method: 'JSONP'}
});
});
```

i wanted get json file from youtube, but it showed nothing. Do you have any idea why it's not working.

Reply

15.  Frank Wang says:
August 12, 2013 at 11:08 pm

This is a very helpful article!
Help me know more about how to use AngularJS!

Reply

16.  nicolsc says:
October 16, 2013 at 2:57 am

Nice article, i’m (almost) tempted to try Angular now 🙂 But these dom extensions still bother me

Your example app is great, as it’s close to real-life implementations and not a plain-old todo list.

In your example, how could you handle the URLs using Angular ? Something like http://devgirl.org/files/MediaExplorer/#/Alicia+Keys/Music

Reply

17.  littleiffel says:
October 16, 2013 at 3:25 am

Hi,

really good article. I am just wondering, how many Items do you display in your ng-repeat directive. I had some trouble with performance when more than 1000 items were displayed. So I had to “tune” it using build in techniques. Like limitTo,…

Reply

18.  James Morris says:
October 20, 2013 at 10:30 pm

I found a similar article and example that was also very helpful on a coding blog:

Searching the iTunes API asynchronously with Angular JS
http://blog.jpamorgan.com/searching-the-itunes-api-asynchronously-with-angular-js/

Reply

19.  Gloria W says:
January 7, 2014 at 11:15 pm

Awesome blog! Thank you for this! It cleared up some questions I had about the use of directives.

Reply

20.  Rohan says:
May 8, 2014 at 4:06 am

Really thanx for this article. It was so helpful for newbie like me. I am trying to get into this. Doing nice job. Expecting more from U.

Reply

Leave a Reply

Name (required)

Email (required)

Website

Submit Comment

Notify me of followup comments via e-mail

« ADC Web Standards Monthly - FREE Newsletter
JavaScript Frameworks Session Resources - Adobe MAX 2013 »

Holly Schinsky

RSS Feed

Recent Posts

- Apache Cordova Newsflash!
- Interactive Config Guide for PhoneGap/Cordova
- Sync Data Using PouchDB In Your Ionic Framework App
- Push Notifications Sample App with Ionic and ngCordova
- PhoneGap CLI 3.6.3 Released with Significant Updates

Enter Search Terms Search

Categories

Categories Select Category

Blogroll

- Andy Trice
- Apache Cordova News
- Brian Rinaldi
- Christophe Coenraets
- Joe Bowser's Blog - (PhoneGap Android Info)
- PhoneGap Blog
- PhoneGap Build Blog
- Ray Camden
- Shazron's Blog - (PhoneGap iOS Info)

Connect

Subscribe

Enter your email address below to receive updates each time we publish new content.