ANGULARJS

Home

Learn ▾

Develop ▾

Discuss ▾

Click or press / to search

v1.3.0 ▾

/ Tutorial / 8 - More Templating

Show / Hide Table of Contents

Previous | Live Demo | Code Diff | Next | Improve this Doc

In this step, you will implement the phone details view, which is displayed when a user clicks on a phone in the phone list.

- When you click on a phone on the list, the phone details page with phone-specific information is displayed.

To implement the phone details view we used $http to fetch our data, and we fleshed out the `phone-detail.html` view template.

Workspace Reset Instructions ➤

The most important changes are listed below. You can see the full diff on GitHub

Data

In addition to `phones.json`, the `app/phones/` directory also contains one json file for each phone:

app/phones/nexus-s.json : (sample snippet)

```
{
"additionalFeatures": "Contour Display, Near Field Communications (NFC),...",
"android": {
  "os": "Android 2.3",
  "ui": "Android"
},
...
"images": [
  "img/phones/nexus-s.0.jpg",
  "img/phones/nexus-s.1.jpg",
  "img/phones/nexus-s.2.jpg",
  "img/phones/nexus-s.3.jpg"
],
"storage": {
  "flash": "16384MB",
  "ram": "512MB"
}
}
```

Each of these files describes various properties of the phone using the same data structure. We'll show this data in the phone detail view.

Controller

We'll expand the `PhoneDetailCtrl` by using the `$http` service to fetch the json files. This works the same way as the phone list controller.

`app/js/controllers.js` **:**

```
var phonecatControllers = angular.module('phonecatControllers',[]);

phonecatControllers.controller('PhoneDetailCtrl', ['$scope', '$routeParams', '$http',
  function($scope, $routeParams, $http) {
    $http.get('phones/' + $routeParams.phoneId + '.json').success(function(data) {
      $scope.phone = data;
    });
  }]);
```

To construct the URL for the HTTP request, we use `$routeParams.phoneId` extracted from the current route by the `$route` service.

Template

The TBD placeholder line has been replaced with lists and bindings that comprise the phone details. Note where we use the Angular `{{expression}}` markup and `ngRepeat` to project phone data from our model into the view.

`app/partials/phone-detail.html` **:**

```
<img ng-src="{{phone.images[0]}}" class="phone">

<h1>{{phone.name}}</h1>

<p>{{phone.description}}</p>

<ul class="phone-thumbs">
  <li ng-repeat="img in phone.images">
    <img ng-src="{{img}}">
  </li>
</ul>

<ul class="specs">
  <li>
    <span>Availability and Networks</span>
    <dl>
      <dt>Availability</dt>
      <dd ng-repeat="availability in phone.availability">{{availability}}</dd>
    </dl>
  </li>
    ...
  <li>
    <span>Additional Features</span>
    <dd>{{phone.additionalFeatures}}</dd>
  </li>
</ul>
```

Test

We wrote a new unit test that is similar to the one we wrote for the `PhoneListCtrl` controller in step 5.

`test/unit/controllersSpec.js` :

```
beforeEach(module('phonecatApp'));

...

describe('PhoneDetailCtrl', function(){
  var scope, $httpBackend, ctrl;

  beforeEach(inject(function(_$httpBackend_, $rootScope, $routeParams, $controller) {
    $httpBackend = _$httpBackend_;
    $httpBackend.expectGET('phones/xyz.json').respond({name:'phone xyz'});

    $routeParams.phoneId = 'xyz';
    scope = $rootScope.$new();
    ctrl = $controller('PhoneDetailCtrl', {$scope: scope});
  }));


  it('should fetch phone detail', function() {
    expect(scope.phone).toBeUndefined();
    $httpBackend.flush();

    expect(scope.phone).toEqual({name:'phone xyz'});
  });
});
...
```

You should now see the following output in the Karma tab:

```
Chrome 22.0: Executed 3 of 3 SUCCESS (0.039 secs / 0.012 secs)
```

We also added a new end-to-end test that navigates to the Nexus S detail page and verifies that the heading on the page is "Nexus S".

test/e2e/scenarios.js :

```
...
describe('Phone detail view', function() {

  beforeEach(function() {
    browser.get('app/index.html#/phones/nexus-s');
  });


  it('should display nexus-s page', function() {
    expect(element(by.binding('phone.name')).getText()).toBe('Nexus S');
  });
});
...
```

You can now rerun `npm run protractor` to see the tests run.

Experiments

Using the Protractor API, write a test that verifies that we display 4 thumbnail images on the Nexus S details page.

Summary

Now that the phone details view is in place, proceed to step 9 to learn how to write your own custom display filter.

⏮ Previous ▶ Live Demo 🔍 Code Diff Next ⏭