v1.5.6-build.4825 (snapshot

/ Tutorial (tutorial)
/ 5 - Filtering Repeaters (tutorial/step_05)

ocs/content/tutorial/step_05.ngdoc?message=docs(tutorial%2F5 - Filtering Repeaters)%3A%20describe%20your%20change...)

Previous (tutorial/step_04) ▶ Live Demo (http://angular.github.io/angular-phonecat/step-5/app) Code Diff (https://github.com/angular/angular-phonecat/compare/step-4...step-5)
Next (tutorial/step_06)

We did a lot of work in laying a foundation for the app in the previous steps, so now we'll do something simple; we will add full-text search (yes, it will be simple!). We will also write an end-to-end (E2E) test, because a good E2E test is a good friend. It stays with your app, keeps an eye on it, and quickly detects regressions.

- The app now has a search box. Notice that the phone list on the page changes depending on what a user types into the search box.

Workspace Reset Instructions ➤

Reset the workspace to step 5.

```
git checkout -f step-5
```

Refresh your browser or check out this step online: Step 5 Live Demo (http://angular.github.io/angular-phonecat/step-5/app).

The most important changes are listed below. You can see the full diff on GitHub (https://github.com/angular/angular-phonecat/compare/step-4...step-5).

Component Controller

We made no changes to the component's controller.

Component Template

`app/phone-list/phone-list.template.html`:

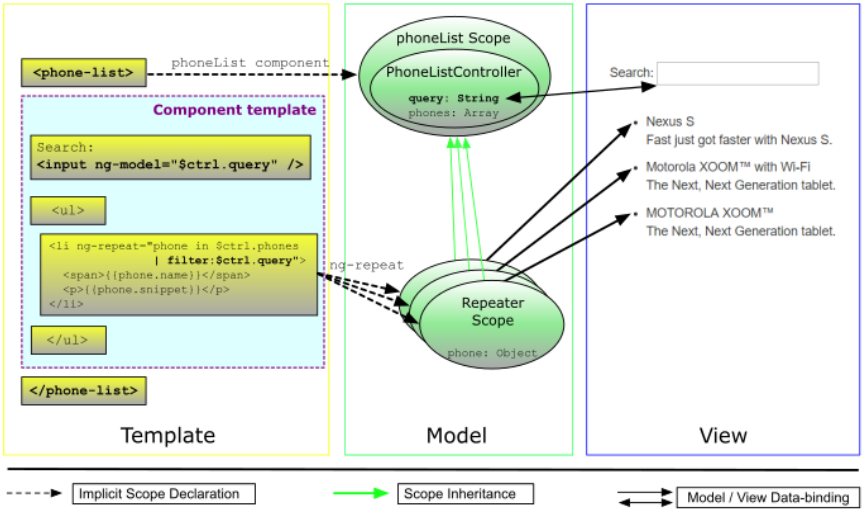
```
<div class="container-fluid">
  <div class="row">
    <div class="col-md-2">
      <!--Sidebar content-->

      Search: <input ng-model="$ctrl.query" />

    </div>
    <div class="col-md-10">
      <!--Body content-->

      <ul class="phones">
        <li ng-repeat="phone in $ctrl.phones |
filter:$ctrl.query">
          <span>{{phone.name}}</span>
          <p>{{phone.snippet}}</p>
        </li>
      </ul>

    </div>
  </div>
</div>
```

We added a standard HTML `<input>` tag and used Angular's filter (api/ng/filter/filter) function to process the input for the ngRepeat (api/ng/directive/ngRepeat) directive.

By virtue of the ngModel (api/ng/directive/ngModel) directive, this lets a user enter search criteria and immediately see the effects of their search on the phone list. This new code demonstrates the following:

- Data-binding: This is one of the core features in Angular. When the page loads, Angular binds the value of the input box to the data model variable specified with `ngModel` and keeps the two in sync.

In this code, the data that a user types into the input box (bound to **`$ctrl.query`**) is immediately available as a filter input in the list repeater (`phone` **`in`** `$ctrl.phones | filter:` **`$ctrl.query`**). When changes to the data model cause the repeater's input to change, the repeater efficiently updates the DOM to reflect the current state of the model.



- Use of the `filter` filter: The filter (api/ng/filter/filter) function uses the `$ctrl.query` value to create a new array that contains only those records that match the query.

  `ngRepeat` automatically updates the view in response to the changing number of phones returned by the `filter` filter. The process is completely transparent to the developer.

Testing

In previous steps, we learned how to write and run unit tests. Unit tests are perfect for testing controllers and other parts of our application written in JavaScript, but they can't easily test templates, DOM

manipulation or interoperability of components and services. For these, an end-to-end (E2E) test is a much better choice.

The search feature was fully implemented via templates and data-binding, so we'll write our first E2E test, to verify that the feature works.

e2e-tests/scenarios.js :

```
describe('PhoneCat Application', function() {

  describe('phoneList', function() {

    beforeEach(function() {
      browser.get('index.html');
    });

    it('should filter the phone list as a user types into
the search box', function() {
      var phoneList = element.all(by.repeater('phone in
$ctrl.phones'));
      var query = element(by.model('$ctrl.query'));

      expect(phoneList.count()).toBe(3);

      query.sendKeys('nexus');
      expect(phoneList.count()).toBe(1);

      query.clear();
      query.sendKeys('motorola');
      expect(phoneList.count()).toBe(2);
    });

  });

});
```

This test verifies that the search box and the repeater are correctly wired together. Notice how easy it is to write E2E tests in Angular. Although this example is for a simple test, it really is that easy to set up any functional, readable, E2E test.

Running E2E Tests with Protractor

Even though the syntax of this test looks very much like our controller unit test written with Jasmine, the E2E test uses APIs of Protractor (https://github.com/angular/protractor). Read about the Protractor APIs in the Protractor API Docs (https://angular.github.io/protractor/#/api).

Much like Karma is the test runner for unit tests, we use Protractor to run E2E tests. Try it with `npm run protractor`. E2E tests take time, so unlike with unit tests, Protractor will exit after the tests run and will not automatically rerun the test suite on every file change. To rerun the test suite, execute `npm run protractor` again.

> **Note:** In order for protractor to access and run tests against your application, it must be served via a web server. In a different terminal/command line window, run `npm start` to fire up the web server.

Experiments

- Display the current value of the `query` model by adding a `{{$ctrl.query}}` binding into the `phone-list.template.html` template and see how it changes, when you type in the input

box.

You might also try to add the `{{$ctrl.query}}` binding to `index.html`. However, when you reload the page, you won't see the expected result. This is because the `query` model lives in the scope defined by the `<phone-list>` component. Component isolation at work!

Summary

We have now added full-text search and included a test to verify that it works! Now let's go on to step 6 (tutorial/step_06) to learn how to add sorting capabilities to the PhoneCat application.

Previous (tutorial/step_04) **Live Demo** (http://angular.github.io/angular-phonecat/step-5/app) **Code Diff** (https://github.com/angular/angular-phonecat/compare/step-4...step-5)