305CDE Developing the Modern web 2 *Lecture 07*

Forms and services

Lecturer: Dr. Jianhua Yang
Module leader: Mr. Mark Tyers
12/11/2014

Coventry University

Contents

- Forms
- Services
- Dependency injection

Forms

form.FormController

- type in module ng

View Source Improve this Doc

`FormController` keeps track of all its controls and nested forms as well as the state of them, such as being valid/invalid or dirty/pristine.

Each form directive creates an instance of `FormController`.

Forms

Properties

`$pristine`

boolean True if user has not

`$dirty`

boolean True if user has alre

`$valid`

boolean True if all of the cont re valid.

`$invalid`

boolean True if at least one c invalid.

`$submitted`

boolean True if user has subr valid.

```
required

ng-required

ng-minlength
ng-maxlength
ng-pattern

type="email"
type="number"

type="date"

type="url"
```

- `ng-valid` : the model is valid
- `ng-invalid` : the model is invalid
- `ng-valid-[key]` : for each valid key added by `$setValidity`
- `ng-invalid-[key]` : for each invalid key added by `$setValidity`
- `ng-pristine` : the control hasn't been interacted with yet
- `ng-dirty` : the control has been interacted with
- `ng-touched` : the control has been blurred
- `ng-untouched` : the control hasn't been blurred
- `ng-pending` : any `$asyncValidators` are unfulfilled

Services

Controllers

For memory and performance purposes, controllers are instantiated only when they are needed and discarded when they are not. That means that every time we switch a route or reload a view, the current controller gets cleaned up by Angular.

Services

Services provide a method for us to keep data around for the lifetime of the app and communicate across controllers in a consistent manner.

Services

```
// Without Dependency Injection
function fetchDashboardData() {
  var $http  = new HttpService();
  return $http.get('my/url');
}

// With Dependency Injection
function fetchDashboardData($http) {
  return $http.get('my/url');
}
```

Services

Function	Defines
`provider(name, Object OR constructor() )`	A configurable service with complex creation logic. If you pass an Object, it should have a function named $get that returns an instance of the service. Otherwise, Angular assumes you've passed a constructor that, when called, creates the instance.
`factory(name, $get Function() )`	A non-configurable service with complex creation logic. You specify a function that, when called, returns the service instance. You could think of this as `provider(name, { $get: $getFunction() } )`.
`service(name, con structor() )`	A non-configurable service with simple creation logic. Like the constructor option with provider, Angular calls it to create the service instance.

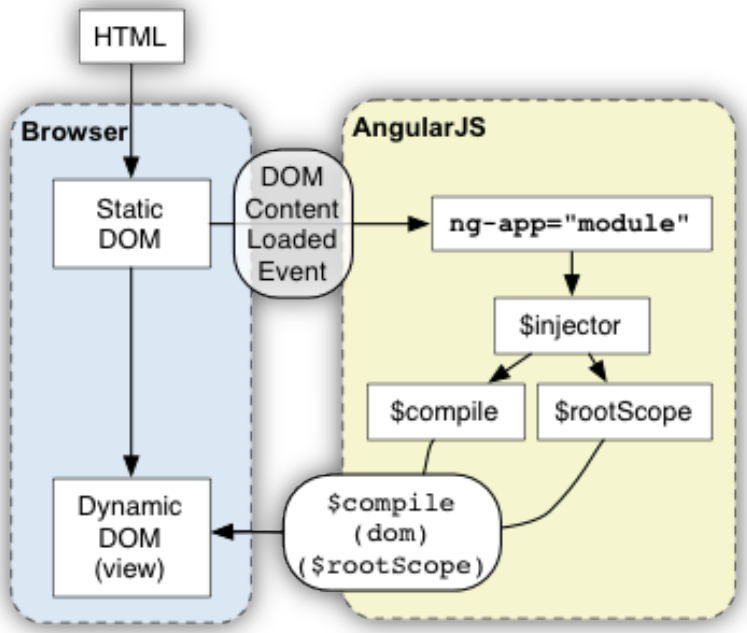
As a naming convention, Angular's built-in services, Scope methods and a few other Angular APIs have a $ prefix in front of the name.

Dependency injection

Initial	Stands for (acronym)	Concept
S	SRP [4]	**Single responsibility principle** a class should have only a single responsibility (i.e. only one potential change in the software's specification should be able to affect the specification of the class)
O	OCP [5]	**Open/closed principle** "software entities … should be open for extension, but closed for modification."
L	LSP [6]	**Liskov substitution principle** "objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program." See also design by contract.
I	ISP [7]	**Interface segregation principle** "many client-specific interfaces are better than one general-purpose interface."[8]
D	DIP [9]	**Dependency inversion principle** one should "Depend upon Abstractions. Do not depend upon concretions."[8] Dependency injection is one method of following this principle.

Dependency injection

```
1. <!doctype html>
2. <html ng-app>
3.   <head>
4.     <script src="http://code.angularjs.org/
   angular-1.0.2.min.js"></script>
5.   </head>
6.   <body>
7.     <p ng-init=" name='World' ">
8.       Hello {{name}}!
9.     </p>
10.  </body>
11.</html>
```



Dependency injection

1. Browser loads the HTML and parses it into a DOM
2. Browser loads angular.js script
3. Angular waits for DOMContentLoaded event
4. Angular looks for ng-app directive, which designates application boundary
5. Module specified in ng-app (if any) is used to configure the $injector
6. $injector is used to create the $compile service as well as $rootScope
7. $compile service is used to compile the DOM and link it with $rootScope
8. ng-init directive assigns World to the nameproperty on the scope
9. The {{name}} interpolates the expression to Hello World!