AngularJS

04

Presented by
Subhash EP

Subhash

Subhash

Course Agenda – Introduction

- 01
  - AngularJS core concepts
  - Two-way binding
- 02
  - Routing basics
  - Templating basics
  - Validations introduction
  - Directives concepts
  - What is dependency injection
- 03
  - Development environment
  - Chrome
  - Chrome plugins for AngularJS – Batarang, ng-Inspetor
  - Sublime Text
  - Angular seed project
  - Anatomy of Angular Project
  - MVW, MVC, MVVM

Course Agenda – Structuring AngularJS

- 01
  - Modules
  - Built-in
  - Own modules
  - Best practices
- 02
  - Controllers
  - Scope
- 03
  - From a simple HTML page to a complex AngualarJS app
  - Ng-prefix
  - Expressions
  - Adding Logic
  - Properties
  - Functions
  - Nesting
  - Dependency Injection and controllers

Course Agenda – Client Side

- 04
  - Events
  - Services
- 05
  - Form element
  - Controls
  - Form Validations

Course Agenda – Routing

- 06
  - Views
  - Routing
  - Factories
  - Providers

Course Agenda – Let us go deeper

- 07
  - AJAX
  - $http
  - Data binding
  - Two-way Data Binding
  - Simple Page Applications
- 08
  - Directives
  - More on DI

Agenda
Validation

Subhash

Built-in Directives for Validation

- ng-maxlength: Validates the maximum allowable characters in a field.
- ng-minlength: Validates the minimum allowable characters in a field.
- ng-required: Marks the fields that are mandatory.
- ng-pattern: Validates a field against a given regex (regular expression).

Form Status - $invalid

- The $invalid property of the form
  - is set to true
  - if at least one field in the form is invalid

Agenda
Routing

Agenda
Routing

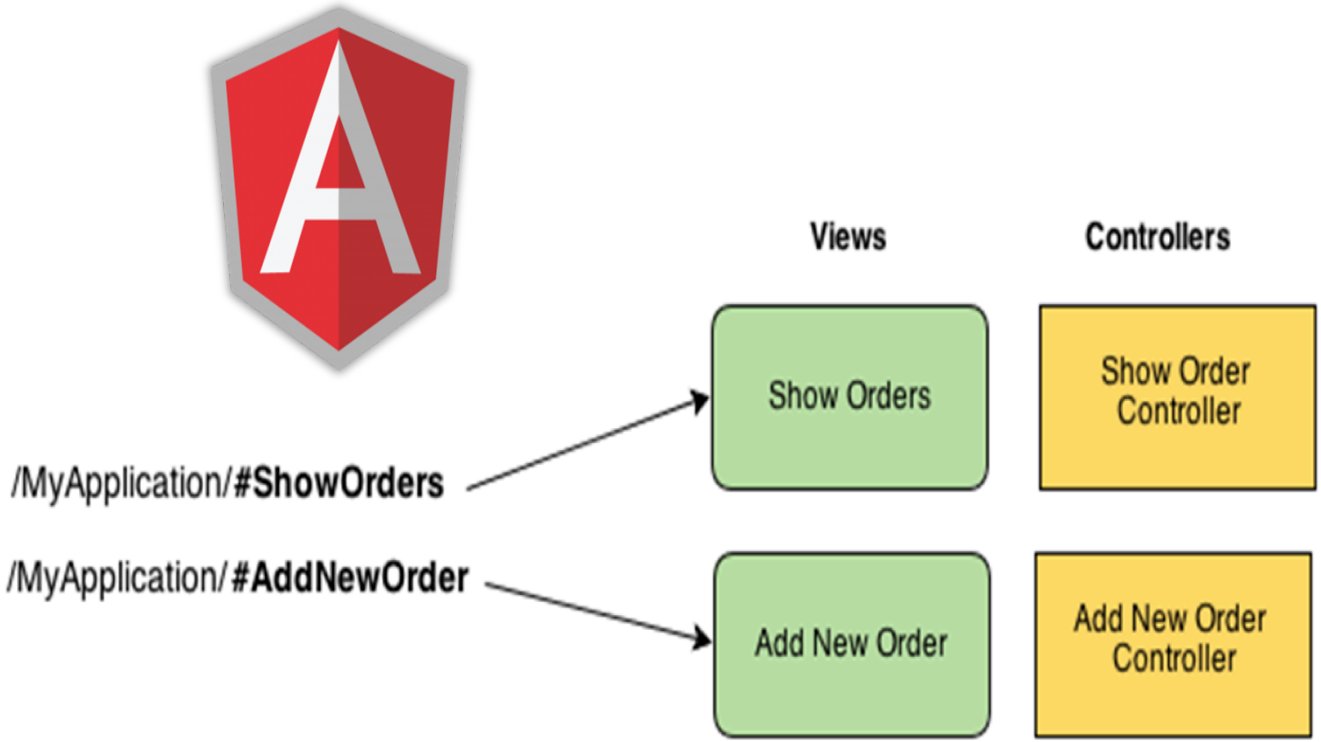
06

Views

Routing

Creating Routes in AngularJS

- Routes, Templates and Controllers



Agenda
Routing

06

Factories

Providers

Providers (1)

- 4 types
  - Factory
  - Service
  - Value
  - Provider

Providers - Factory

- We define a function that returns an object to which we have attached methods and properties that will be accessible by factory users later
- This object is available everywhere in the module in which the factory was defined via Dependency Injection

Factory example (1) – defining the factory

```
module.factory('factory_id', function() {
    return {
            functionname: function() {
                return "this is a function";
            },
            anotherfunction: function() {
            // make a request and get data from backend
                return data;
            }
        }
});
```

Factory example (2) – using the factory

```
Module.controller('ControllerName', function
    ControllerName($scope, factory_id) {
    $scope.methodname = function() {
                factory_id.functionname();
    }
});
```

Providers - Service

- You define a function in which additional functions and properties are defined via the this keyword

Service example (1) - defining a service

```
module.service('service_name', function() {
    this.function_name = function() {
                return "this is a function's result";
    };
    this.anotherfunction = function() {
                // make a request to backend
                // and fetch data
                return data;
});
```

Service example (2) - using a service in a controller

```
Module.controller('ControllerName', function
    ControllerName($scope, service_name) {
    $scope.methodname = function() {
                service_name.function_name();
    }
});
```

Providers - value

- Similar to constants
- Could be used to store configuration properties

Value example (1) - defining a value

```
module.value('value_name', 'value');
```

Value example (2) - using a value in a controller

```
Module.controller('ControllerName', function
    ControllerName($scope, value_name) {
    $scope.methodname = function() {
                if (value_name == '1') {
                        // do something based on
                        // specific value of the constant
                }
    }
});
```

Providers - provider

- Define $get method in a function that returns the object to be injected
- The object can have various properties and methods similar to the object returned by factory

Provider example (1) - defining a provider

```
module.provider('provider_name', function() {
    this.$get = function() {
                return {
                        function_name: function() {
                        },
                        another_function: function() {
                        }
    }
});
```

Provider example (2) - Using a provider

```
Module.controller('ControllerName', function
    ControllerName($scope, provider_name) {
    $scope.methodname = function() {
                provider_name.function_name();
    }
});
```

That's all

End of Session

©

subhash.ep@gmail.com
linkedin.com/in/subhashep