**View**

AngularJS supports Single Page Application via multiple views on a single page. To do this AngularJS has provided ng-view and ng-template directives and $routeProvider services.

ng-view

ng-view tag simply creates a place holder where a corresponding view (html or ng-template view) can be placed based on the configuration.

Usage

Define a div with ng-view within the main module.

<div ng-app = "mainApp">

...

<div ng-view></div>

</div>

ng-template

ng-template directive is used to create an html view using script tag. It contains "id" attribute which is used by $routeProvider to map a view with a controller.

Usage

Define a script block with type as ng-template within the main module.

<div ng-app = "mainApp">

...

<script type = "text/ng-template" id = "addStudent.htm">

<h2> Add Student </h2>

{{message}}

</script>

</div>

$routeProvider

$routeProvider is the key service which set the configuration of urls, map them with the corresponding html page or ng-template, and attach a controller with the same.

Usage

Define a script block with main module and set the routing configuration.

var mainApp = angular.module("mainApp", ['ngRoute']);

mainApp.config(['$routeProvider', function($routeProvider) {

$routeProvider.

when('/addStudent', {

templateUrl: 'addStudent.htm', controller: 'AddStudentController'

}).

when('/viewStudents', {

templateUrl: 'viewStudents.htm', controller: 'ViewStudentsController'

}).

otherwise({

redirectTo: '/addStudent'

});

}]);

Following are the important points to be considered in above example.

* $routeProvider is defined as a function under config of mainApp module using key as '$routeProvider'.
* $routeProvider.when defines a url "/addStudent" which then is mapped to "addStudent.htm". addStudent.htm should be present in the same path as main html page.If htm page is not defined then ng-template to be used with id="addStudent.htm". We've used ng-template.
* "otherwise" is used to set the default view.
* "controller" is used to set the corresponding controller for the view.

Example

Following example will showcase all the above mentioned directives.

*testAngularJS.htm*

<html>

<head>

<title>Angular JS Views</title>

<script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

<script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular-route.min.js"></script>

</head>

<body>

<h2>AngularJS Sample Application</h2>

<div ng-app = "mainApp">

<p><a href = "#addStudent">Add Student</a></p>

<p><a href = "#viewStudents">View Students</a></p>

<div ng-view></div>

<script type = "text/ng-template" id = "addStudent.htm">

<h2> Add Student </h2>

{{message}}

</script>

<script type = "text/ng-template" id = "viewStudents.htm">

<h2> View Students </h2>

{{message}}

</script>

</div>

<script>

var mainApp = angular.module("mainApp", ['ngRoute']);

mainApp.config(['$routeProvider', function($routeProvider) {

$routeProvider.

when('/addStudent', {

templateUrl: 'addStudent.htm',

controller: 'AddStudentController'

}).

when('/viewStudents', {

templateUrl: 'viewStudents.htm',

controller: 'ViewStudentsController'

}).

otherwise({

redirectTo: '/addStudent'

});

}]);

mainApp.controller('AddStudentController', function($scope) {

$scope.message = "This page will be used to display add student form";

});

mainApp.controller('ViewStudentsController', function($scope) {

$scope.message = "This page will be used to display all the students";

});

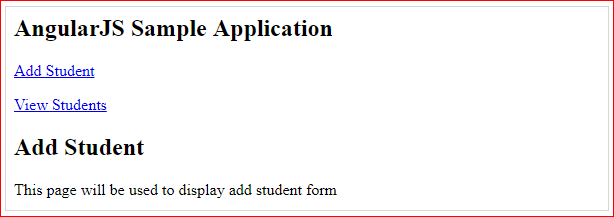
</script>

</body>

</html>

Result

Open textAngularJS.htm in a web browser. See the result



**Scope**

Scope is a special javascript object which plays the role of joining controller with the views. Scope contains the model data. In controllers, model data is accessed via $scope object.

<script>

var mainApp = angular.module("mainApp", []);

mainApp.controller("shapeController", function($scope) {

$scope.message = "In shape controller";

$scope.type = "Shape";

});

</script>

Following are the important points to be considered in above example.

* $scope is passed as first argument to controller during its constructor definition.
* $scope.message and $scope.type are the models which are to be used in the HTML page.
* We've set values to models which will be reflected in the application module whose controller is shapeController.
* We can define functions as well in $scope.

## Scope Inheritance

Scope are controllers specific. If we defines nested controllers then child controller will inherit the scope of its parent controller.

<script>

var mainApp = angular.module("mainApp", []);

mainApp.controller("shapeController", function($scope) {

$scope.message = "In shape controller";

$scope.type = "Shape";

});

mainApp.controller("circleController", function($scope) {

$scope.message = "In circle controller";

});

</script>

Following are the important points to be considered in above example.

* We've set values to models in shapeController.
* We've overridden message in child controller circleController. When "message" is used within module of controller circleController, the overridden message will be used.

## Example

Following example will showcase all the above mentioned directives.

*testAngularJS.htm*

<html>

<head>

<title>Angular JS Forms</title>

</head>

<body>

<h2>AngularJS Sample Application</h2>

<div ng-app = "mainApp" ng-controller = "shapeController">

<p>{{message}} <br/> {{type}} </p>

<div ng-controller = "circleController">

<p>{{message}} <br/> {{type}} </p>

</div>

<div ng-controller = "squareController">

<p>{{message}} <br/> {{type}} </p>

</div>

</div>

<script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

<script>

var mainApp = angular.module("mainApp", []);

mainApp.controller("shapeController", function($scope) {

$scope.message = "In shape controller";

$scope.type = "Shape";

});

mainApp.controller("circleController", function($scope) {

$scope.message = "In circle controller";

});

mainApp.controller("squareController", function($scope) {

$scope.message = "In square controller";

$scope.type = "Square";

});

</script>

</body>

</html>

## Widget Example (Scope Inheritance)

## \*\*\*\*\*HTML\*\*\*\*

## <html>

## <head>

## <title>Angular JS Forms</title>

## </head>

## 

## <body>

## <h2>AngularJS Sample Application</h2>

## 

## <div >

## <p >{{student.Shape()}} </p>

## </div>

## <div >

## <p >{{student.Circle()}} </p>

## </div>

## 

## </body>

## </html>

## \*\*\*\*\*\*\*\*\*Client Script\*\*\*\*\*\*\*\*\*\*

## function($scope){

## var c = this;

## $scope.student = {

## firstName: "Mahesh",

## lastName: "Parashar",

## fees:500,

## 

## subjects:[

## {name:'Test',marks:70}

## ],

## 

## Shape: function() {

## var studentObject;

## studentObject = $scope.student;

## return studentObject.firstName + " " + studentObject.lastName;

## },

## Circle: function() {

## var studentObject;

## studentObject = $scope.student;

## return studentObject.firstName + " " + studentObject.lastName;

## },

## Test: function() {

## var studentObject;

## studentObject = $scope.student;

## return studentObject.subjects.name + " " + studentObject.subjects.marks;

## }

## }

## }

## Result

Open textAngularJS.htm in a web browser. See the result.



**Services**

AngularJS supports the concepts of "Separation of Concerns" using services architecture. Services are javascript functions and are responsible to do a specific tasks only. This makes them an individual entity which is maintainable and testable. Controllers, filters can call them as on requirement basis. Services are normally injected using dependency injection mechanism of AngularJS.

AngularJS provides many inbuilt services for example, $https:, $route, $window, $location etc. Each service is responsible for a specific task for example, $https: is used to make ajax call to get the server data. $route is used to define the routing information and so on. Inbuilt services are always prefixed with $ symbol.

There are two ways to create a service.

* factory
* service

Using factory method

Using factory method, we first define a factory and then assign method to it.

var mainApp = angular.module("mainApp", []);

mainApp.factory('MathService', function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b

}

return factory;

});

Using service method

Using service method, we define a service and then assign method to it. We've also injected an already available service to it.

mainApp.service('CalcService', function(MathService){

this.square = function(a) {

return MathService.multiply(a,a);

}

});

Example

Following example will showcase all the above mentioned directives.

*testAngularJS.htm*

<html>

<head>

<title>Angular JS Services</title>

<script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

</head>

<body>

<h2>AngularJS Sample Application</h2>

<div ng-app = "mainApp" ng-controller = "CalcController">

<p>Enter a number: <input type = "number" ng-model = "number" /></p>

<button ng-click = "square()">X<sup>2</sup></button>

<p>Result: {{result}}</p>

</div>

<script>

var mainApp = angular.module("mainApp", []);

mainApp.factory('MathService', function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b

}

return factory;

});

mainApp.service('CalcService', function(MathService){

this.square = function(a) {

return MathService.multiply(a,a);

}

});

mainApp.controller('CalcController', function($scope, CalcService) {

$scope.square = function() {

$scope.result = CalcService.square($scope.number);

}

});

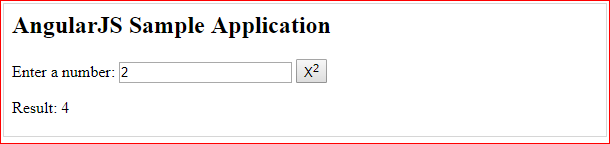
</script>

</body>

</html>

Result

Open textAngularJS.htm in a web browser. See the result.



**Dependency Injection**

Dependency Injection is a software design pattern in which components are given their dependencies instead of hard coding them within the component. This relieves a component from locating the dependency and makes dependencies configurable. This helps in making components reusable, maintainable and testable.

AngularJS provides a supreme Dependency Injection mechanism. It provides following core components which can be injected into each other as dependencies.

* value
* factory
* service
* provider
* constant

value

value is simple javascript object and it is used to pass values to controller during config phase.

//define a module

var mainApp = angular.module("mainApp", []);

//create a value object as "defaultInput" and pass it a data.

mainApp.value("defaultInput", 5);

...

//inject the value in the controller using its name "defaultInput"

mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {

$scope.number = defaultInput;

$scope.result = CalcService.square($scope.number);

$scope.square = function() {

$scope.result = CalcService.square($scope.number);

}

});

factory

factory is a function which is used to return value. It creates value on demand whenever a service or controller requires. It normally uses a factory function to calculate and return the value.

//define a module

var mainApp = angular.module("mainApp", []);

//create a factory "MathService" which provides a method multiply to return multiplication of two numbers

mainApp.factory('MathService', function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b

}

return factory;

});

//inject the factory "MathService" in a service to utilize the multiply method of factory.

mainApp.service('CalcService', function(MathService){

this.square = function(a) {

return MathService.multiply(a,a);

}

});

...

service

service is a singleton javascript object containing a set of functions to perform certain tasks. Services are defined using service() functions and then injected into controllers.

//define a module

var mainApp = angular.module("mainApp", []);

...

//create a service which defines a method square to return square of a number.

mainApp.service('CalcService', function(MathService){

this.square = function(a) {

return MathService.multiply(a,a);

}

});

//inject the service "CalcService" into the controller

mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {

$scope.number = defaultInput;

$scope.result = CalcService.square($scope.number);

$scope.square = function() {

$scope.result = CalcService.square($scope.number);

}

});

provider

provider is used by AngularJS internally to create services, factory etc. during config phase(phase during which AngularJS bootstraps itself). Below mention script can be used to create MathService that we've created earlier. Provider is a special factory method with a method get() which is used to return the value/service/factory.

//define a module

var mainApp = angular.module("mainApp", []);

...

//create a service using provider which defines a method square to return square of a number.

mainApp.config(function($provide) {

$provide.provider('MathService', function() {

this.$get = function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b;

}

return factory;

};

});

});

constant

constants are used to pass values at config phase considering the fact that value can not be used to be passed during config phase.

mainApp.constant("configParam", "constant value");

Example

Following example will showcase all the above mentioned directives.

*testAngularJS.htm*

<html>

<head>

<title>AngularJS Dependency Injection</title>

</head>

<body>

<h2>AngularJS Sample Application</h2>

<div ng-app = "mainApp" ng-controller = "CalcController">

<p>Enter a number: <input type = "number" ng-model = "number" /></p>

<button ng-click = "square()">X<sup>2</sup></button>

<p>Result: {{result}}</p>

</div>

<script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

<script>

var mainApp = angular.module("mainApp", []);

mainApp.config(function($provide) {

$provide.provider('MathService', function() {

this.$get = function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b;

}

return factory;

};

});

});

mainApp.value("defaultInput", 5);

mainApp.factory('MathService', function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b;

}

return factory;

});

mainApp.service('CalcService', function(MathService){

this.square = function(a) {

return MathService.multiply(a,a);

}

});

mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {

$scope.number = defaultInput;

$scope.result = CalcService.square($scope.number);

$scope.square = function() {

$scope.result = CalcService.square($scope.number);

}

});

</script>

</body>

</html>

Result

Open textAngularJS.htm in a web browser. See the result.

