A controller is a JavaScript object containing attributes/properties and functions

Filters are used to change modify the data and can be clubbed in expression or directives using pipe character. Following is the list of commonly used filters.

Modules are used to separate logics say services, controllers, application etc.

In this example we're going to create two modules.

* **Application Module** − used to initialize an application with controller(s).
* **Controller Module** − used to define the controller.
* ng-click
* ng-dbl-click
* ng-mousedown
* ng-mouseup
* ng-mouseenter
* ng-mouseleave
* ng-mousemove
* ng-mouseover
* ng-keydown
* ng-keyup
* ng-keypress
* ng-change

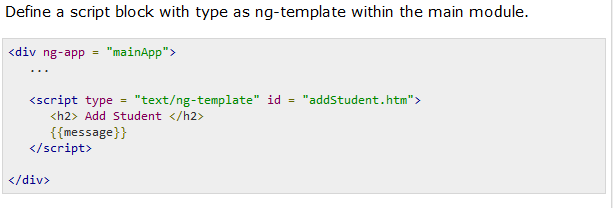
**Validate data**

* **$dirty** − states that value has been changed.
* **$invalid** − states that value entered is invalid.
* **$error** − states the exact error.

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

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.



$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.

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

<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>

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";

});

AngularJS supports the concepts of "Separation of Concerns" using services architecture. Services are javascript functions and are responsible to do a specific tasks only.

Services are normally injected using dependency injection mechanism of AngularJS.

There are two ways to create a service.

* factory
* service

<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>

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.

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);

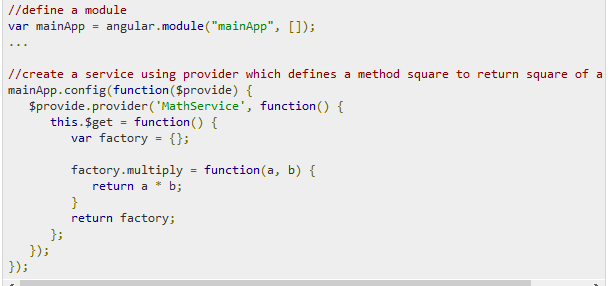
$scope.square = function() {

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

}

});

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.



## 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");

我们建议你为服务、指令和过滤器分别定义不同的模块。然后你的主模块可以声明依赖这些模块。

**1.Config代码块**

        在这一阶段里面，AngularJS会连接并注册好所有数据源。因此，只有数据源和常量可以注入到Config代码块中。那些不确定是否已经初始化好的服务不能注入进来。

   Run代码块用来启动你的应用，并且在注射器创建完成之后开始执行。为了避免在这一点开始之后再对系统进行配置操作，只有实例和常量可以被注入到Run代码块中。你会发现，在AngularJS中，Run代码块是与main方法最类似的东西。

The ng serve command builds the app, starts the development server, watches the source files, and rebuilds the app as you make changes to those files.

The --open flag opens a browser to http://localhost:4200/.

**Angular components**

The page you see is the *application shell*. The shell is controlled by an Angular **component** named AppComponent.

*Components* are the fundamental building blocks of Angular applications. They display data on the screen, listen for user input, and take action based on that input.

Open the project in your favorite editor or IDE and navigate to the src/app folder.

You'll find the implementation of the shell AppComponent distributed over three files:

1. app.component.ts— the component class code, written in TypeScript.
2. app.component.html— the component template, written in HTML.
3. app.component.css— the component's private CSS styles.

When building modules, we like to write them in their own directory named after the module. We’ll implement the module initialization in a file by the name of the module. For instance, we’re building a game module, so we’ll build our game module inside the app/scripts/game directory in a file named game.js. This methodology has provided to be scalable and logical in production.