Softtek®

Business Applications Maximized

AngularJS

Curso Angular

Softtek®

Business Applications Maximized

AngularJS Introduccion

Softtek®

¿Qué es AngularJS?

AngularJS framework

Softtek®

AngularJS es un framework de JavaScript de código abierto, mantenido por Google, que ayuda con la gestión de lo que se conoce como aplicaciones de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

Características principales

Softtek®

2-way-binding: Una de las principales características de Angular el binding de datos, ya que AngularJS estara observando continuamente los cambios en el modelo.

Directivas (Directives): Las directivas son marcadores en el DOM (atributos, tags, comentarios o clases CSS) que le asignaran al compilador HTML de Angular un comportamiento específico, esto nos permitirá crear nuestros propios elementos HTML.

Expresiones: Los valores que se tengan que plasmar en la vista, deberán ser puestos entre dos corchetes, sin embargo, de esta forma también se pueden evaluar expresiones como {{1+3}}, {{modelo.cantidad+3}}, {{modelo.texto}}.

Vistas y rutas: Las rutas nos permitirán cambiar las vistas en nuestra aplicación, haciendo posible cambiar solo una parte de la página.

I**nyección de dependencias**: AngularJS cuenta con un contenedor de inversión de control, esto quiere decir que a la hora de definir un componente, angular se encargará de proporcionarnos el objeto por medio de su constructor o su factory.

Espacio de trabajo

Softtek®

Al ser AngularJS un framework de JavaScript, tendremos que importar la librería en nuestras aplicaciones web, dentro de los encabezados:

```
<script src="http://ajax.googleapis.
com/ajax/libs/angularjs/1.3.15/angular.min.js"></script>
```

Introduccion a AngularJS

Softtek®

- Extensión de HTML
  - Directivas
  - expresiones
- Aplicaciones AngularJS

Directivas de Angular

La directiva **ng-app** define una aplicación de AngularJS.
La directiva **ng-controller** liga el controlador con el elemento HTML
La directiva **ng-model** une los elementos HTML (input, select, textarea) al modelo de datos de la aplicación.
La directiva **ng-bind** une el modelo de datos de la aplicación a la vista.

```
<!DOCTYPE html>
    <html>
        <head>
            <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.
js"></script>
        </head>
    <body>
        <div ng-app="">
          <p>Name: <input type="text" ng-model="name"></p>
          <p ng-bind="name"></p>
        </div>
    </body>
</html>
```

AngularJS inicia automáticamente cuando la página es cargada.
La directiva **ng-app** le dice a AngularJS que el elemento es el “dueño” de la aplicación AngularJS.
La directiva **ng-model** une el valor del input a la variable “name” de la aplicación.
La directiva **ng-bind** une el contenido HTML del <p> a la variable “name” de la aplicación.
Más adelante entraremos en detalle con las directivas.

Expresiones

Las expresiones de AngularJS se escriben dentro de dos corchetes: {{ expresión }}.

AngularJS escribirá los datos exactamente donde se sitúe la expresión.

```
<div ng-app="">
    <p>Mi primer expresión: {{ 5 + 5 }}</p>
</div>
```

Filtros

Softtek®

Los filtros en las expresiones son utilizados para transformar los datos, por ejemplo, dándole un formato específico.

```
<div ng-app="">
    <p>Expresión angular: {{ cantidad | currency}}< /p>
</div>
```

```
<div ng-app="">
    <p>Expresión angular: {{ nombre | uppercase}}< /p>
</div>
```

También pueden ser usadas dentro de las directivas para generar algunos comportamientos, como filter y orderBy

```
<ul >
 <li ng-repeat="x in nombres | filter:test | orderBy:'nombres'" >
    Expresión angular: {{ x | uppercase}}
 </li>
</ul>
```

Aplicaciones Angular

Softtek®

Los **modules** definen las aplicaciones de AngularJS. Los **controllers** controlan las aplicaciones AngularJS.

La directiva **ng-app** define la aplicación en el HTML y la directiva **ng-controller** define el **controller**.

Ejemplo:

```
<div ng-app="myApp" ng-controller="myCtrl">
    Nombre: <input type="text" ng-model="firstName"><br>
    Apellido: <input type="text" ng-model="lastName"><br>
    <br>
    Nombre completo: {{firstName + " " + lastName}}

</div>

<script>
    var app = angular.module('myApp', []);
    app.controller('myCtrl', function($scope) {
        $scope.firstName= "Juan";
        $scope.lastName= "Pérez";
    });
</script>
```

Introduccion a AngularJS

Softtek®

- Binding de datos
  - one-way binding
  - two-way binding
- Modelos de binding

Binding de datos

El binding o enlazado de datos, permite ligar los datos de nuestro modelo con los elementos de la vista.

Existen dos formas de enlaze de datos:

1. One-way binding: el binding unidireccional nos permite ligar el modelo con los elementos DOM, donde el usuario no puede interactuar de forma dinamica.

```
<div id="mensajes">
  {{modelo}}
</div>

<div ng-bind="modelo"></div>
```

2. Two-way binding: El binding bidireccional, se da cuando el modelo se liga con un elemento que tambien permite la entrada de datos (INPUT), por ejemplo:

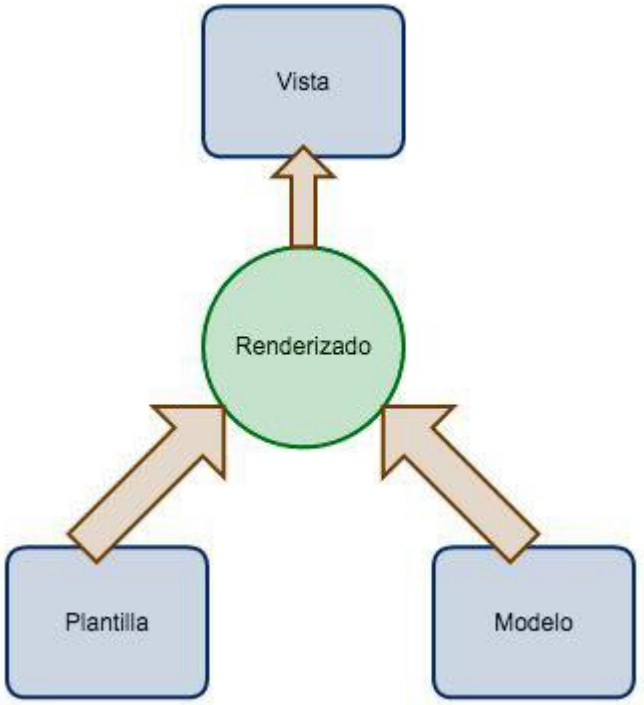
```
<select ng-model="opcion">
  <option value="1">opcion 1</option>
  <option value="2">opcion 2</option>
</select>

Mensaje:<input type="search" ng-model="message" > mundo!
```

Modelo tradicional de binding

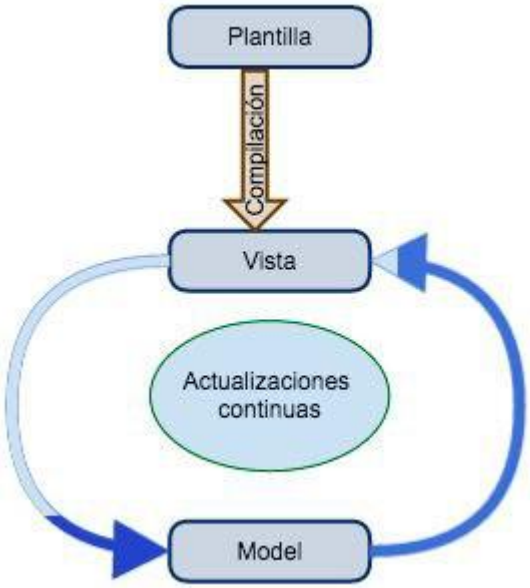
Softtek®

Por un lado tenemos el modelo tradicional, donde el motor de renderizado une la plantilla con el modelo de datos, pero esto obliga a hacer actualizaciones manuales del DOM al actualizar los datos del modelo.



Modelo de binding de AngularJS

AngularJS nos permite hacer un binding de datos dinámico, es decir, los datos que se actualicen en el modelo, actualizará la vista, aunque hay que tener cuidado con los listeners que se levanten por este binding.



AngularJS

Softtek®

- Modulos
- Controllers
  - Eventos (ng-click)

Modulos

Softtek®

Los modulos sirven para desacoplar la funcionalidad de nuestra aplicación, y estos nos permitiran declarar funcionalidad reutilizable, ya sea por una característica, por componente o funcionalidad.

la sintaxis para declarar un modulo en javascript sera:

```
angular.module('Nombre_del_modulo', [dependencias...]);
```

Las dependencias que inyectamos, pueden ser funcionalidades que el mismo angular nos brinda, o puede ser alguna funcionalidad nuestra, más adelante profundizaremos en este tema.

```
angular.module('Nombre_del_modulo', [ngRoute]);
```

Los módulos son ligados a nuestro HTML por medio de la directiva ng-app.

```
<div ng-app="myApp">
```

Controladores

Softtek®

Los controladores son objetos donde se nos permitirá desarrollar la lógica de nuestra aplicación, es a partir de aquí donde se enlaza el ámbito de $scope con la vista, permitiéndonos un control de todos los datos, también aquí podremos manejar todos nuestros eventos.

Los controladores se enlazan a la vista mediante la directiva ng-controller.

```
<div ng-controller="myCtrl">
```

La sintaxis para declarar un controller es la siguiente

```
var app = angular.module('MyApp', []);
   app.controller('controller', function($scope, dependencias, …,){
    //contenido
   });
```

Dentro de la declaración del controller inyectaremos el ámbito $scope, pero también podemos inyectar otras dependencias, como por ejemplo los servicios.

```
<body>
    <div ng-controller="mainController">
           <h1>Hola AngularJS mundo</h1>
    <div/>
</body>
```

Controladores

Softtek®

También es válido anidar controladores.

```
<div ng-controller="myCtrl">
   {{cuenta}}
   <button ng-click="aumentarCuentaHijo()">Aumentar</button>
   <div ng-controller="innerController">
       {{cuenta}}
       <button ng-click="aumentarCuentaPadre()">Aumentar</button>
   </div>
</div>
```

```
var app = angular.module('MyApp', []);
   app.controller(myCtrl, function($scope){
     $scope.cuenta=0;
     $scope.aumentarCuentaHijo =function(){}
   });
   app.controller(innerController, function($scope){
    $scope.cuenta=0;
    $scope.aumentarCuentaPadre =function(){}
   });
```

Eventos

Softtek®

Los eventos del mouse tienen sus propias directivas, algunas de ellas son ng-click. ng-dblclick, ng-mouseover … etc. Estas directivas pueden asociarse con codigo “inline” o con funciones creadas dentro de los controllers

```
<button ng-click="count = count + 1">count++</button>
<button ng-click="contar()">count++ controller</button>
```

La sintaxis para declarar una funcion dentro de un controller es la siguiente:

```
var app = angular.module('MyApp', []);
   app.controller('controller', function($scope){
    $scope.contar=function(){
        $scope.count += 1;
    }
   });
```

Propagación de eventos

Softtek®

Ya que es posible tener controllers anidados, es posible propagar los eventos hacia los padres o hacia los hijos (parents-childs) eso lo lograremos gracias a **$broadcast**, **$emit** y **$on**.
Existen dos direcciones para propagar el evento, hacia el padre (parent) o hacia los hijos (childs).

Los eventos eventos hacia el padre se transmiten por medio de $scope.$emit, emite los eventos que pueden ser capturados por los scopes de los padres, mientras que $scope.$broadcast emite los eventos hacia todo los scopes hijos.

```
var app = angular.module('MyApp', []);
app.controller('controllerPadre', function($scope) {
    $scope.count = 0;
    $scope.contarEnHijo = function() {
        $scope.$broadcast('to_childrens', 1);
    }
    $scope.$on('to_parent', function(event, data) {
        $scope.count += data;
    });
});

app.controller('controllerHijo', function($scope) {
    $scope.count = 0;
    $scope.$on('to_childrens', function(event, data) {
        $scope.count += data;
    });
    $scope.contarPadre = function() {
        $scope.$emit('to_parent', 1);
    }
});
```

AngularJS

Softtek®

- Creacion de filtros y directivas

Creación de filtros

Softtek®

Es posible crear nuestros propios filtros, los filtros no solo nos permiten dar formato a cadenas si no manipular todo tipo de objetos, un ejemplo muy usado es para manipular arrays.

```
angular.module('nombreModuloFiltros', []).filter('nombreFiltro', function(){
    return function (input, params...){
         return resultado;
    };
});

var app=angular.module('modulo', ['nombreModuloFiltros']);

Por ejemplo:

angular.module('filtros', []).filter('cortar', function(){
    return function (input, max){
         if(input==null) return;
        if(input.length>max)
             return input.substring(0,max)+'...';
    };
});

Mensaje: {{mensaje | cortar:5 | uppercase}}
```

Creación de directivas - Resumen

Softtek®

También nos es posible crear directivas, para esto haremos un resumen de lo que sabemos de las directivas y veremos algunas cosas nuevas.

Para invocar una directiva existen 4 métodos:

Como un atributo: `<span mi-directiva></span>`
Como una clase: `<span class="mi-directiva: expresion;"></span>`
Como un elemento: `<mi-directiva></my-directive>`
Como un comentario: `<!-- directive: mi-directiva expression -->`

Angular podrá reconocer las directivas en todas esas formas.

Los siguientes nombres de directiva son válidos para ngBind:

```
<div ng-controller="Controller">
  Hello <input ng-model='name'> <hr/>
  <span ng-bind="name"></span> <br/>
  <span ng:bind="name"></span> <br/>
  <span ng_bind="name"></span> <br/>
  <span data-ng-bind="name"></span> <br/>
  <span x-ng-bind="name"></span> <br/>
</div>
```

Esto es debido a que Angular normaliza los elementos y atributos, el proceso de normalizacion elimina las cadenas “x-” y “data-”, ademas convierte la siguiente letra despues de “:”, “-” o “_” para convertirlo en su formato camelCase, todas las formas anteriores son equivalentes para ngBind.

Creación de directivas - Sintaxis

Softtek®

Al igual que los controllers y los filtros, las directivas son registradas en los modulos, para registrar una directiva utilizaremos la función “module.directive”:

```
var app = angular.module('modulo', []);
app.directive('directiva', function() {
  return {
    template: 'template'
  };
});
```

También se puede utilizar templateUrl para importar plantillas más grandes.

Se puede restringir a que la directiva sea usada solo de una forma específica con la opción “restrict” con los valores:

‘A’ - solo se podra usar como atributo
‘E’ - solo se podra usar como element
‘C’ - solo se podra usar como una clase

Estas opciones se pueden mezclar para permitir cuantas opciones se necesiten: ‘AEC’

Se recomienda utilizar un elemento cuando se estan creando secciones especificas con una funcionalidad bien definida, y se utiliza un atributo cuando se va a decorar un elemento que ya existe.

Creación de directivas - scopes

Softtek®

Las directivas pueden acceder al scope del controlador donde se utilice, pero también contienen un scope al cual se les puede enviar datos, para esto usaremos la opción scope:

```
.directive('contacto', function() {
  return {
    restrict: 'E',
    scope: {
      contactInfo: '=info'
    },
    templateUrl: 'ficha.html'
  };
});
```

En este caso, se creará una variable en el scope aislado de la directiva llamado ‘contactInfo’, mientras que la cadena ‘=info’ avisa al compilador de angular ($compile) que se debe enlazar al atributo ‘info’, tambien estos nombres de atributo estan normalizados, de forma que si queremos crear un atributo con un nombre “atributo-de-directiva” tendremos que usar una cadena como esta ‘=atributoDeDirectiva’, pero si queremos que el atributo sea igual al nombre de la variable en nuestro scope aislado, solo usaremos ‘=’.

Una vez que creemos un scope aislado para nuestra directiva, está ya no tendrá acceso al scope del controller, por lo tanto no podremos acceder al sus variables, por lo que es una buena práctica utilizar el scope aislado solo cuando el componente se reutilizara.

Creación de directivas - wrap elements

Softtek®

Algunas veces necesitamos poder pasar cualquier contenido u objeto dentro de nuestras directivas, para lo cual no siempre nos servira el scope aislado, para eso podremos hacer wrap de otros elmentos y sera mediante la opcion transclude :

-- archivo .js

```
.directive('contacto', function() {
  return {
    restrict: 'E',
    transclude: true,
    templateUrl: 'ficha.html'
  };
});
```

--ficha.html

```
<div ng-transclude>
</div>
```

-- archivo.html

```
<div ng-controller="Controller">
  <contacto>datos de controller: {{name}}!</contacto>
</div>
```

La opcion transclude hara visible el scope exterior dentro de la directiva.

Creación de directivas - Link

La propiedad link nos ayudara a crear logica dentro de nuestra directiva, donde podremos manejar el scope de la directiva o incluso agregar eventos a nuestra directiva:

```
.directive('directiva', function() {
  return {
    restrict: 'E',
    link: function(scope, elemento, attributos,controllers){},
    templateUrl: 'ficha.html'
  };
});
```

```
scope - Sera el escope (scope aislado) de nuestra directiva.
elemento - es el elemento de nuestra directiva, envuelto en un objeto de jqLite
atributos - una tabla con llaves-valores, con los atributos mapeados de la directiva
```

Creación de directivas - Controller

la propiedad controller tiene el mismo funcionamiento que la propiedad link, sin embargo, con esta propiedad, podremos exponer funcionalidad hacia las directivas que estén dentro de la directiva.

```
.directive('directiva', function() {
  return {
    restrict: 'E',
    controller: function($scope,... servicios){
       this.funcion = function();
      },
    templateUrl: 'ficha.html'
  };
});

.directive('directivaHija', function() {
  return {
    restrict: 'E',
    requires: '^directiva',
    link: function(scope, element, attrs, directivaController){
         directivaController.funcion();
      }
    templateUrl: 'ficha.html'
  };
});
```

AngularJS

Softtek®

- Rutas

Rutas

Las rutas serán las herramientas que nos permitirán manejar nuestro proyecto como una aplicación de una sola página.

Podremos definir las rutas inyectando el servicio de ngRoute dentro de nuestro módulo, para esto debemos agregar también un nuevo JS.

<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.15/angular-route.js"></script>

-- archivo.js

```
var app = angular.module("modulo", ["ngRoute"]);
```

Las rutas serán configuradas dentro del config del módulo, no podrán ser configuradas en otro lugar y será con el servicio **$routeProvider**, el servicio $routeProvider nos esencialmente dos funciones: when y otherwise

```
app.config(function($routeProvider){
      $routeProvider.when('/ruta', {
            templateUrl:'template.html',
            controller:'controlador'
        }).otherwise({
           redirectTo: '/';
         });
});
```

Las rutas serán accesibles por la url agregando el carácter '#' seguido de la ruta,

ir a ruta

AngularJS

Softtek®

- Constantes y Servicios

Constantes

Softtek®

Declarar una constante es muy sencillo, lo haremos mediante dos funciones que nos proporciona angular.

```
module.value("nombre",valor);
module.constant("nombre", valor);
```

y para utilizarlo dentro de nuestro código, tendremos que inyectar la constante en nuestro controller, filtro o servicio.

```
var app = angular.controller("controller", function($scope, constante){
     console.log(constante.valor);
});
```

Servicios

Softtek®

Hasta ahora hemos estado creando toda la lógica de negocio dentro de nuestros controllers, pero lo más adecuado es que separemos el código en servicios, que aislaran la funcionalidad de nuestra lógica de negocio.

Tenemos tres formas de declarar un servicio, con las funciones “factory”, “service” y “provide”.

La función factory, nos regresara el objeto resultado de la invocación del mismo.

```
module.factory( 'nombreServicioFactory', function );

module.factory( 'nombreServicioFactory', function(){

   var funciones = {
         sumar: function(a,b){
              return a+b;
         },
         restar: function(a,b){
              return a-b;
         }
   }

   return funciones;

} );

module.controller('controller', function($scope, nombreServicioFactory){

$scope.suma=nombreServicioFactory.sumar(2,3);
$scope.restar=nombreServicioFactory.restar;

});
```

Servicios

La función service, invocará el constructor del objeto que estemos declarando, permitiéndonos usar nuestro código de una forma más orientada a objetos.

La sintaxis es la siguiente:

```
module.service( 'nombreServicio', function );
```

Por ejemplo:

```
module.service( 'nombreServicio', function(){
   this.sumar(a, b){
       return a+b;
   }
   this.restar(a,b){
       return a-b;
   }
} );

module.controller('controlador', function($scope, nombreServicio){
    $scope.suma=nombreServicio.suma(1+2);
    $scope.restar=nombreServicio.restar;
}
```

Servicios

La función provider es la opción más poderosa, pero también es la más complicada, esta función nos permitirá configurar nuestro servicio antes de crearlo, nuestro servicio sera lo que esté configurado en la opción $get del provider.

La sintaxis es la siguiente:

```
module.provider( 'nombreServicio', function );
```

Por ejemplo:

```
module.provider( 'nombreServicio', function($servicios){
   var idUsuario=null;
   this.setIdUsuario = function(id){
       idUsuario = id;
   }

   function Servicio(){
       this.saludar=function(){
          return 'hola '+idUsuario;
       }
   }

   this.$get = function(a) { return new Servicio(); };

} );

module.config(function(nombreServicioProvider){
      nombreServicioProvider.setIdUsuario('Hiram');
});

module.controller('controlador', function($scope, nombreServicio){
    console.log(nombreServicio.saludar()); //la salida seria hola Hiram
}
```