

Capítulo 11

ASP NET Core

Objetivo

Al finalizar el capítulo, el alumno:

- Crear las aplicaciones con ASP.NET Core 2.0.

Temas

1. Introducción a ASP.NET Core
2. Creando aplicación con ASP.NET Core
3. ASP.NET Core MVC
 - Model
 - View
 - Controller
 - Routing
4. ASP.NET Core Web API

1. Conceptos de SPA y AngularJS

Conceptos de SPA y AngularJS

- Un single-page application (SPA), o aplicación de página única es una aplicación web o es un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios como una aplicación de escritorio. En un SPA todos los códigos de HTML, JavaScript, y CSS se carga de una vez o los recursos necesarios se cargan dinámicamente como lo requiera la página y se van agregando, normalmente como respuesta de las acciones del usuario.
- AngularJS, o simplemente Angular, es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web 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.

12 - 4

Copyright © Todos los Derechos Reservados - Cibertec Perú SAC.



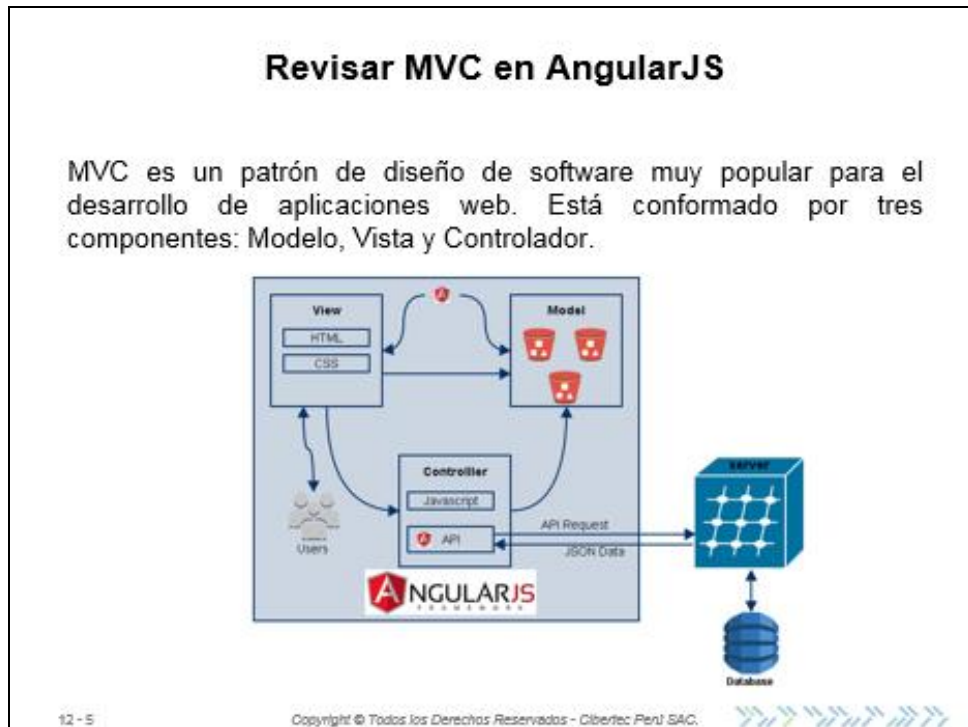
Un single-page application (SPA), o aplicación de página única es una aplicación web o es un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios como una aplicación de escritorio. En un SPA todos los códigos de HTML, JavaScript, y CSS se cargan de una vez o los recursos necesarios se cargan dinámicamente como lo requiera la página y se van agregando, normalmente como respuesta de las acciones del usuario. La página no tiene que cargar otra vez en ningún punto del proceso tampoco se transfiere a otra página, aunque las tecnologías modernas (como el `pushState()` API del HTML5) pueden permitir la navegabilidad en páginas lógicas dentro de la aplicación. La interacción con las aplicaciones de página única puede involucrar comunicaciones dinámicas con el servidor web que está detrás.

AngularJS, o simplemente Angular, es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web 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.

La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript. Los valores de las variables de JavaScript se pueden configurar manualmente, o recuperados de los recursos JSON estáticos o dinámicos.

Angular sigue el patrón MVC de ingeniería de software y alienta la articulación flexible entre la presentación, datos y componentes lógicos. Con el uso de la inyección de dependencias, Angular lleva servicios tradicionales del lado del servidor, tales como controladores dependientes de la vista, a las aplicaciones web del lado del cliente. En consecuencia, gran parte de la carga en el backend se reduce, lo que conlleva a aplicaciones web mucho más ligeras.

2. Revisar MVC en AngularJS



MVC es un patrón de diseño de software muy popular para el desarrollo de aplicaciones web. Está conformado por tres componentes: Modelo, Vista y Controlador.

- **Modelo:** Representa el estado de la aplicación.

En AngularJS un modelo está representado por un objeto en JavaScript. Pueden ser tipos primitivos como boolean, cadenas, números o tipo complejos como objetos.

- **Vista:** Representa las vistas (template) que se muestran al usuario.

El usuario debería ver vistas que tienen la apariencia de aplicaciones de escritorio (desktops), Tablets y Smartphones, en angular la vista es el DOM (Document Object Model). Con ayuda de angular expression se puede formatear y mostrar información.

Ejemplo de un template en AngularJS

```

<style>
.alto30
{
    height: 30px;
}
</style>
<div ng-controller="RptFichaSeguimientoController">
    <div class="row">
        <!--IZQUIERDA-->
        <div class="col-md-6">
            
        </div>
        <!--DERECHA -->
        <div class="col-md-6">
            <div class="row alto30">
                <div class="col-md-3"> <b>FILE N°</b> </div>
                <div class="col-md-3">: {{data.nroExpediente}}</div>
            </div>
            <div class="row alto30">
                <div class="col-md-3"><b>RAMO</b></div>
                <div class="col-md-3">: {{data.ramoDescripcion}}</div>
            </div>
            <div class="row alto30">
                <div class="col-md-3"><b>EJECUTIVO</b></div>
                <div class="col-md-3">: {{data.ejecutivoNombre}}</div>
            </div>
        </div>
    </div>
    <br>
    <div class="row alto30">
        <div class="col-md-3"><b>ASEGURADO</b></div>
        <div class="col-md-6">: {{data.nombreAsegurado}}</div>
    </div>

    <div class="row alto30">
        <div class="col-md-3"><b>CONTACTO</b></div>
        <div class="col-md-3">: {{data.contactoNombre}}</div>

        <div class="col-md-3"><b>TELEFONO</b></div>
        <div class="col-md-3">: {{data.contactoTelefono}}</div>
    </div>

    <div class="row alto30">
        <div class="col-md-3"><b>CIA ASEGURADORA</b></div>
        <div class="col-md-3">: {{data.ciaAseguradora}}</div>

        <div class="col-md-3"><b>N° SINIESTRO</b></div>
        <div class="col-md-3">: {{data.nroSiniestro}}</div>
    </div>

    <div class="row alto30">
        <div class="col-md-3"><b>POLIZA</b></div>
        <div class="col-md-3">: </div>

        <div class="col-md-3"><b>N° POLIZA</b></div>
        <div class="col-md-3">: {{data.numPoliza}}</div>
    </div>
</div>

```

```

<div class="row alto30">
  <div class="col-md-3"><b>COBERTURA</b></div>
  <div class="col-md-3" >:</div>

  <div class="col-md-3"><b>VIGENCIA</b></div>
  <div class="col-md-3" >:</div>
</div>

<div class="row alto30">
  <div class="col-md-3"><b>DETALLE OCURRENCIA</b></div>
  <div class="col-md-3" >: {{data.expedienteReferencia}}</div>

  <div class="col-md-3"><b>LUGAR</b></div>
  <div class="col-md-3" >: {{data.polizaDireccion}}</div>
</div>
<div class="row alto30">
  <div class="col-md-3"><b>FECHA / HORA OCURR</b></div>
  <div class="col-md-3" >: {{data.fechaOcurriencia}}</div>
</div>

<div class="row alto30">
  <div class="col-md-3"><b>MONTO DE LA PERDIDA</b></div>
  <div class="col-md-3" >: {{data.montoPerdida}}</div>

  <div class="col-md-3"><b>TIPO DE CAMBIO</b></div>
  <div class="col-md-3" >: {{data.tipoCambio}}</div>
</div>

<div class="row alto30">
  <div class="col-md-3"><b>RESERVA</b></div>
  <div class="col-md-9" >: {{data.reserva}}</div>
</div>
<div class="row alto30">
  <div class="col-md-3"><b>CORREDOR DE SEGUROS</b></div>
  <div class="col-md-9" >: {{data.corredorSeguro}}</div>
</div>

<div class="row alto30">
  <div class="col-md-3"><b>CONTACTO</b></div>
  <div class="col-md-3" >: {{data.corredorContacto}}</div>

  <div class="col-md-3"><b>TELEFONO</b></div>
  <div class="col-md-3" >: {{data.contactoTelefono}} , {{data.contactoCelular}}</div>
</div>

<div class="row alto30">
  <div class="col-md-3"><b>F. ENCARGO</b></div>
  <div class="col-md-3" >: {{data.expedienteFechaCreacion}}</div>

  <div class="col-md-3"><b>F. CONFIRMACION</b></div>
  <div class="col-md-3" >: {{data.aviso_cia}}</div>
</div>
<div class="row alto30">
  <div class="col-md-3"><b>F.I.BAS</b></div>
  <div class="col-md-3" >:</div>

  <div class="col-md-3"><b>F. I. PRE</b></div>
  <div class="col-md-3" >:</div>
</div>
</div>

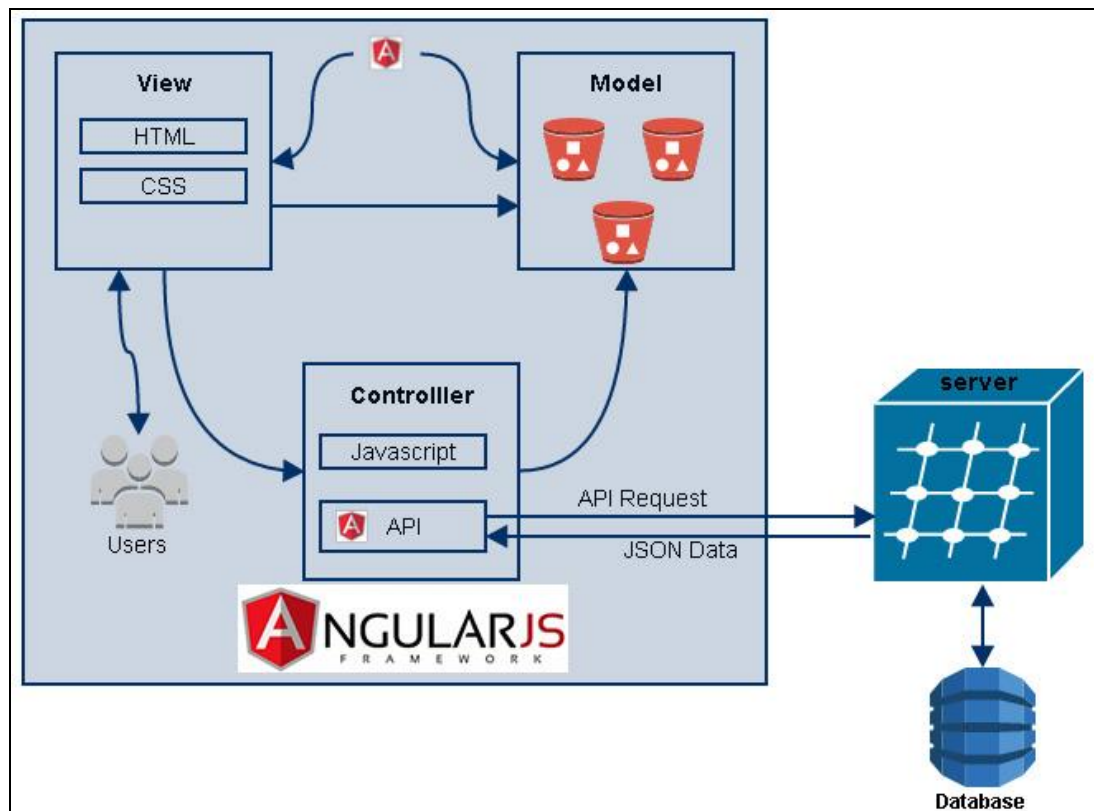
```

- **Controlador:** Es el componente que recibe solicitudes del usuario e interactúa con el modelo y la vista para mostrarle un resultado mediante una vista con datos.

En AngularJS, el controlador es una clase en JavaScript. Ejemplo de controlador en AngularJS:

```
(function() {  
    var injectParams = ['$scope', 'reportService'];  
    var RptFichaSeguimientoController = function($scope,reportService) {  
        $scope.data = {  
        };  
        init();  
        function init(){  
  
reportService.getFichaSeguimientoPorExpediente(10).then(function(dataExpediente)  
{  
  
            if(dataExpediente){  
  
                $scope.data = dataExpediente;  
            }  
            else{  
                $scope.errorMessage='Usuario no valido';  
            }  
        });  
    }  
  
};  
  
    RptFichaSeguimientoController.$inject = injectParams;  
    angular.module('segurosApp').controller('RptFichaSeguimientoController',  
RptFichaSeguimientoController);  
  
})();
```

En el ejemplo se define un controlador utilizando el método **controller** que se encuentra disponible en la definición del módulo (aplicación).



- **Configuración de AngularJS**

Para el desarrollo de una aplicación SPA basada en angular se debe descargar las librerías de angular.

<https://angularjs.org/>

En el proyecto web en la página index.html, hacer referencia al archivo javascript angular.js o angular.min.js

```
<script src="js/angular.js"></script>
```


3. Routing en AngularJS

AngularJS routes permite crear URLs para los diferentes contenidos o vistas de la aplicación. Cada Url representa una ruta (route)

AngularJS routes muestra un contenido o vista dependiendo de qué ruta se ha seleccionado. Una ruta se define con el signo **#<nombre de la ruta>** después del url de la aplicación.

Ejemplos:

```
http://myangularjsapp.com/index.html#books
http://myangularjsapp.com/index.html#albums
http://myangularjsapp.com/index.html#games
http://myangularjsapp.com/index.html#apps
```

Cuando desde el browser se invoque a esas rutas, el archivo index se cargará (http://myangularjsapp.com/index.html), pero AngularJS buscará la ruta (la parte del URL después de #) y decide qué template html va a mostrar.

Pasos para implementar routing en AngularJS

1. Para poder implementar el mecanismo de routing se tiene que descargar una librería angular-route.js o angular-route.min.js y hacer referencia en el proyecto.

```
<script src="js/angular-route.js"></script>
```

2. Como segundo paso se debe agregar la dependencia **ngRoute** a nivel de la aplicación o modulo.

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

3. Agregar la directiva **ngView** en la página index.html del proyecto web.

```
<div ng-view></div>
```

4. Definir las rutas utilizando \$routeProvider.

```
<script>
  module.config(['$routeProvider',
    function($routeProvider) {
      $routeProvider.
        when('/route1', {
          templateUrl: 'angular-route-template-1.html',
          controller: 'RouteController'
        }).
        when('/route2', {
          templateUrl: 'angular-route-template-2.html',
          controller: 'RouteController'
        }).
        otherwise({
          redirectTo: '/'
        });
    });
</script>
```

Donde:

When: se define la siguiente información:

- Ruta: representa la ruta para una vista determinada.
- templateUrl: es la ruta donde se ubica el template o la vista.
- controller: hace referencia a la clase en javascript que representa al controlador con el cual desea trabajar el template o la vista.

otherwise: cualquier otra ruta que no esté definida dentro de las funciones When.

5. Las rutas pueden ser invocadas de la siguiente manera:

```
<a href="#/route1">Route 1</a><br/>
<a href="#/route2">Route 2</a><br/>
```

Rutas con parámetros

```
#/books/12345
```

En este caso es necesario definir el \$routeProvider de la siguiente manera:

```
<script>
  module.config(['$routeProvider',
    function($routeProvider) {
      $routeProvider.
        when('/route1/:param', {
          templateUrl: 'angular-route-template-1.html',
          controller: 'RouteController'
        }).
        when('/route2/:param', {
          templateUrl: 'angular-route-template-2.html',
          controller: 'RouteController'
        }).
        otherwise({
          redirectTo: '/'
        });
    });
</script>
```

En la función **When**, al momento de definir la ruta se debe indicar el nombre del parámetro con el formato **:<nombre del parámetro>**

Para tener acceso a estos parámetros dentro de los controladores se debe utilizar como parámetro **\$routeParams** y con el nombre definido en \$routeProvider acceder al parámetro definido.

```
module.controller("RouteController", function($scope, $routeParams) {
  $scope.param = $routeParams.param;
})
```

4. Services en AngularJS

Services en AngularJS

Son objetos que se conectan al resto de componentes de la aplicación como los controladores usando Inyección de Dependencia (DI). Puede ser usado para organizar y compartir código en toda la aplicación.

```
(function () {  
    var injectParams = ['$http', '$q'];  
    var factory = function ($http, $q) {  
        var service = {};  
  
        service.GetProductos = function (nombre) {  
            var url = "Product/BusquedaService";  
            return $http.get(url).then(function (results) {  
                var data = results.data;  
                return data;  
            });  
        };  
  
        return service;  
    };  
    factory.$inject = injectParams;  
    angular.module('awApp').factory('ProductosService', factory);  
})();
```

12 - 13

Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C.

Son objetos que se conectan al resto de componentes de la aplicación como los controladores usando Inyección de Dependencia (DI). Puede ser usado para organizar y compartir código en toda la aplicación.

Los servicios son usados comúnmente para crear componentes que van a permitir conectar los Rest Services con la Aplicación Web SPA

Registrando un servicio:

```
var myModule = angular.module('myModule', []);  
myModule.factory('serviceld', function() {  
    var shinyNewServiceInstance;  
    // factory function body that constructs shinyNewServiceInstance  
    return shinyNewServiceInstance;  
});
```

Donde myModule, representa la aplicación. Luego se utiliza el método **factory** para crear el servicio con un nombre determinado. En el ejemplo se ha indicado que el servicio va a tener como nombre **serviceld**, luego del nombre se indica el comportamiento del servicio.

Ejemplo práctico de un servicio:

```
(function () {  
  
    var injectParams = ['$http', '$q'];  
  
    var factory = function ($http, $q) {  
        var serviceBase = 'rest/documento/',  
            service = {};  
  
        service.GetListaDocumentos = function () {  
            return $http.get(serviceBase).then(function (results) {  
                var data = results.data;  
                return data;  
            });  
        };  
  
        service.AddDocumento = function (documento) {  
            return $http.post(serviceBase, documento).then(function (results) {  
                var data = results.data;  
                return data;  
            });  
        };  
  
        return service;  
    };  
  
    factory.$inject = injectParams;  
  
    angular.module('segurosApp').factory('documentoService', factory);  
  
})();
```


5. Inyección de dependencia en AngularJS

Inyección de dependencia en AngularJS

Inyección de Dependencia (DI) es un patrón de diseño de software que permite resolver las dependencias entre los componentes de la aplicación.

```
someModule.controller('MyController', ['$scope', 'dep1', 'dep2', function($scope, dep1, dep2) {  
    ...  
    $scope.aMethod = function() {  
        ...  
    }  
    ...  
}]);
```

```
var MyController = function($scope, greeter) {  
    // ...  
}  
MyController.$inject = ['$scope', 'greeter'];  
someModule.controller('MyController', MyController);
```

12 - 14 Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C. 

Inyección de Dependencia (DI) es un patrón de diseño de software que permite resolver las dependencias entre los componentes de la aplicación.

DI en angular está a cargo de la creación de componentes, la solución de sus dependencias y proporcionar a otros componentes cuando se soliciten.

Se puede aplicar DI en componentes tales como Servicios, Directivas y Filtros, en los controladores.

En los controladores se especifican los componentes que serán inyectados por dependencia después de definir el nombre del controlador:

```
someModule.controller('MyController', ['$scope', 'dep1', 'dep2', function($scope, dep1, dep2) {  
    ...  
    $scope.aMethod = function() {  
        ...  
    }  
    ...  
}]);
```

Otra manera más adecuada de indicar DI es utilizando la palabra la propiedad **\$inject** de componente (controlador o servicio) y luego indicar los componentes que se desean inyectar.

```
var MyController = function($scope, greeter) {  
  // ...  
}  
MyController.$inject = ['$scope', 'greeter'];  
someModule.controller('MyController', MyController);
```

6. Directivas en AngularJS

Directivas en AngularJS


Las directivas en AngularJS permiten la manipulación y reutilización de código HTML.

- Primera tipología

```
<div class="container" ng-app="myApp" ng-controller="MainCtrl">  
  <div class="btn-group">  
    <label class="btn btn-primary" ng-model="radioModel" btnradio="Auto">Auto</label>  
    <label class="btn btn-primary" ng-model="radioModel" btn-  
      radio="Manual">Manual</label>  
  </div>  
</div>
```

- Segunda tipología

```
<customize-it></customize-it>
```

12 - 15 Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C. 

Las directivas en AngularJS permiten la manipulación y reutilización de código HTML. Siguiendo la propia documentación de AngularJS, las directivas son marcas en los elementos del árbol DOM, en los nodos del HTML, que indican al compilador de Angular que debe asignar cierto comportamiento a dichos elementos o transformarlos según corresponda.

Podríamos decir que las directivas nos permiten añadir comportamiento dinámico al árbol DOM haciendo uso de las nativas del propio AngularJS o extender la funcionalidad hasta donde necesitemos creando las nuestras propias. De hecho, la recomendación, es que sea en las directivas en el único sitio donde manipulemos el árbol DOM, para que entre dentro del ciclo de vida de compilación, binding y renderización del HTML.

Las directivas son las técnicas que nos van a permitir crear nuestros propios componentes visuales encapsulando las posibles complejidades en la implementación, normalizando y parametrizándolos según nuestras necesidades.

Tipos

La primera tipología de directivas son las nativas del propio Angular y cualquiera puede crear sus propias directivas o hacer uso de directivas de terceros.

En el siguiente código podemos encontrar las siguientes directivas:

```
<div class="container" ng-app="myApp" ng-controller="MainCtrl">
  <div class="btn-group">
    <label class="btn btn-primary" ng-model="radioModel" btnradio="Auto">Auto</label>
    <label class="btn btn-primary" ng-model="radioModel" btn-radio="Manual">Manual</label>
  </div>
</div>
```

- **ng-app (ngApp)**: es la directiva nativa de Angular que arranca nuestra aplicación estableciendo el elemento raíz de la misma.
- **ng-controller (ngController)**: es la directiva nativa que asocia a una vista el controlador que mantiene la vinculación del modelo y gestiona los eventos de control de la misma.
- **ng-model (ngModel)**: es la directiva nativa que vincula una propiedad en concreto del modelo declarado en el controlador a un componente de la vista; normalmente a un componente de formulario para obtener información.
- **btn-radio (btnRadio)**: es una directiva de angular-bootstrap que transforma una etiqueta en un componente de tipo radio button de bootstrap
- Lo normal es que las directivas que comienzan por **ng*** sean de Angular y cada módulo o aplicación tenga su propio prefijo, pero no tiene porque, de hecho las directivas de bootstrap no lo tienen.

La segunda tipología que podemos encontrar en las directivas es su ámbito de aplicación a los nodos del árbol DOM, de modo tal que se pueden asociar:

- Mediante la declaración de un atributo 'A' en cualquier elemento del DOM

```
<div customize-it></div>
```

- Mediante la declaración de un clase css 'C' en cualquier atributo de tipo class de un elemento del DOM

```
<div class="customize-it"></div>
```

- Mediante la asignación a un comentario 'M' en cualquier bloque de código comentado

```
<!-- directive: customize-it -->
```

- Mediante la declaración de un elemento 'E' del propio árbol DOM

```
<customize-it></customize-it>
```

También se puede combinar la declaración de un elemento con la asignación de un atributo 'EA' o añadirle también la declaración en un comentario 'EAC'

Lo normal es disponer y declarar directivas a nivel de elemento y atributo.

Directivas propias

A la hora de crear directivas, la recomendación es que se nombren con camelCase y para hacer referencia a las mismas se separan con guiones; aunque se podrían utilizar otras técnicas al usar guiones evitamos errores si usamos herramientas que validen nuestro HTML.

En el siguiente código podemos ver cómo declarar una directiva en un módulo ya existente:

```
(function() {  
  
    angular.module('tnt.ui.components')  
  
    .directive('userInfo', [function() {  
        return {  
            restrict: 'E',  
            template: 'Nombre: {{user.name}}, email: <a  
href="mailto:{{user.email}}">{{user.email}}</a>' ,  
            };  
        }]);  
    }());
```

Para hacer uso de la misma bastaría con el siguiente código:

```
<div ng-app="tnt.ui.components" ng-controller="DemoDirectivesCtrl">  
    <user-info></user-info>  
</div>
```

La directiva da por hecho que en el ámbito del controlador existe una propiedad “user” con la información a renderizar del usuario:

```
(function() {  
  
    angular.module('tnt.ui.components').controller('DemoDirectivesCtrl',  
    function($scope){  
        $scope.user = {  
            name: 'Jose',  
            email: 'jmsanchez@autentia.com'  
        };  
    }());
```

7. Crear la estructura de una SPA con AngularJS

Crear la estructura de una SPA con AngularJS

- Crear un archivo index.html y hacer referencia a la librerías de AngularJS y Bootstrap.



```
Schema: http://json.schemastore.org/bower
{
  "name": "ASP.NET",
  "private": true,
  "dependencies": {
    "bootstrap": "3.3.5",
    "jquery": "2.1.4",
    "jquery-validation": "1.14.0",
    "jquery-validation-unobtrusive": "3.2.4",
    "log4javascript": "1.4.9",
    "angular": "~1.5.6",
    "angular-route": "~1.5.6"
  }
}
```

12 - 17

Copyright © Todos los Derechos Reservados - Cibertec Perú SAC.



Realizar los siguientes pasos:

1. Descargar las librerías de AngularJS mediante Bower.



```
Schema: http://json.schemastore.org/bower
{
  "name": "ASP.NET",
  "private": true,
  "dependencies": {
    "bootstrap": "3.3.5",
    "jquery": "2.1.4",
    "jquery-validation": "1.14.0",
    "jquery-validation-unobtrusive": "3.2.4",
    "log4javascript": "1.4.9",
    "angular": "~1.5.6",
    "angular-route": "~1.5.6"
  }
}
```

2. Crear un archivo index.html y hacer referencia a la librerías de AngularJS y Bootstrap.

```
<link rel="stylesheet" href="lib/bootstrap/dist/css/bootstrap.min.css" />
<script type="text/javascript" src="lib/angular/angular.min.js"></script>
<script type="text/javascript" src="lib/angular-route/angular-route.min.js"></script>
```

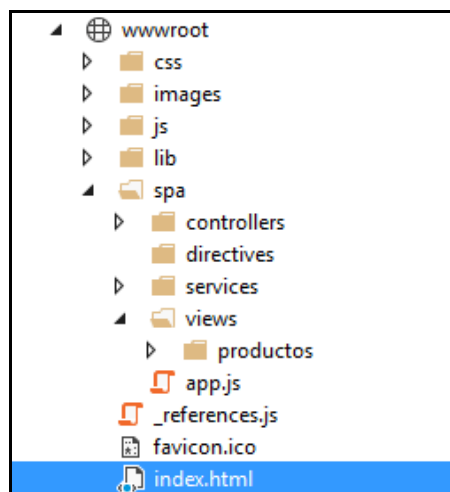
3. El archivo index.html debe tener la siguiente estructura:

```
<!DOCTYPE html>
<html ng-app="awApp">
<head>
  <meta charset="utf-8" />
  <title></title>
  <link rel="stylesheet" href="lib/bootstrap/dist/css/bootstrap.min.css" />
  <script type="text/javascript" src="lib/angular/angular.min.js"></script>
  <script type="text/javascript" src="lib/angular-route/angular-route.min.js"></script>
</head>
<body>
  <div ng-view></div>
</body>
</html>
```

En el elemento html se debe indicar el nombre del módulo en angular mediante la directiva **ng-app**

También se debe indicar mediante la directiva **ng-view** el lugar donde se van a renderizar los templates o vistas.

4. Crear una estructura de carpetas como en la imagen:



En el archivo app.js se debe crear el nombre del módulo (aplicación) y se deben indicar la rutas.

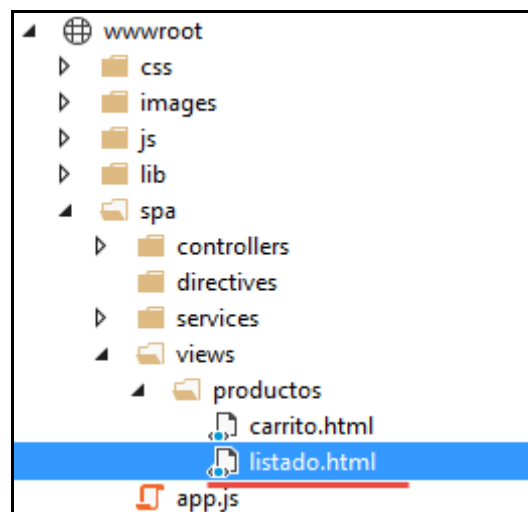
```
var app = angular.module('awApp', ['ngRoute']);

app.config(['$routeProvider', function ($routeProvider) {
  var viewBase = '/app/segurosApp/views/';
  $routeProvider
    .when('/busqueda', {
      controller: 'ProductosController',
      templateUrl: 'spa/views/productos/listado.html'
    })
    .when('/carrito', {
      controller: 'ProductosController',
      templateUrl: 'spa/views/productos/carrito.html'
    })
    .otherwise({ redirectTo: '/' });
}]);
```

Con la palabra clave **angular.module** se debe crear el módulo que representa a la aplicación.

Una vez creado el módulo con su respectivo nombre("awApp") se deben crear las rutas, controladores, servicios en función a este nombre.

5. Crear el template de productos.



```
<a href="#carrito">Ir a Carrito</a>
<table class="table">
  <tr>
    <th>Nombre</th>
    <th>Precio</th>
  </tr>
  <tr ng-repeat="model in listaProductos">
    <td>
      {{model.Name}}
    </td>
    <td>
      {{model.ListPrice}}
    </td>
  </tr>
</table>
```

```
</tr>  
</table>
```

6. Crear el controlador de productos con el nombre **ProductosController** y asociarlo al módulo **awApp**.

```
(function () {  
  
    var injectParams = ['$scope', '$location', '$routeParams', 'ProductosService'];  
  
    var ProductosController = function ($scope, $location, $routeParams, ProductosService) {  
  
        $scope.listaProductos = {};  
  
        function init()  
        {  
            ProductosService.GetProductos('').then(function (data) {  
                $scope.listaProductos = data;  
            }, function (data) {  
                $scope.listaProductos = [];  
            });  
        }  
  
        init();  
  
    };  
  
    ProductosController.$inject = injectParams;  
  
    angular.module('awApp').controller('ProductosController', ProductosController);  
  
})();
```

Tener en cuenta que el controlador agrega por DI algunos componentes de Angular, pero el componente **ProductosServices** es un servicio creado para la aplicación. El código para la implementación del servicio es el siguiente:

```
(function () {  
  
    var injectParams = ['$http', '$q'];  
  
    var factory = function ($http, $q) {  
        var service = {};  
  
        service.GetProductos = function (nombre) {  
            var url = "Product/BusquedaService";  
            return $http.get(url).then(function (results) {  
                var data = results.data;  
                return data;  
            });  
        };  
  
        return service;  
    };  
  
    factory.$inject = injectParams;
```

```
angular.module('awApp').factory('ProductosService', factory);  
});
```

El servicio está llamando a un servicio Rest implementado con ASP.NET MVC

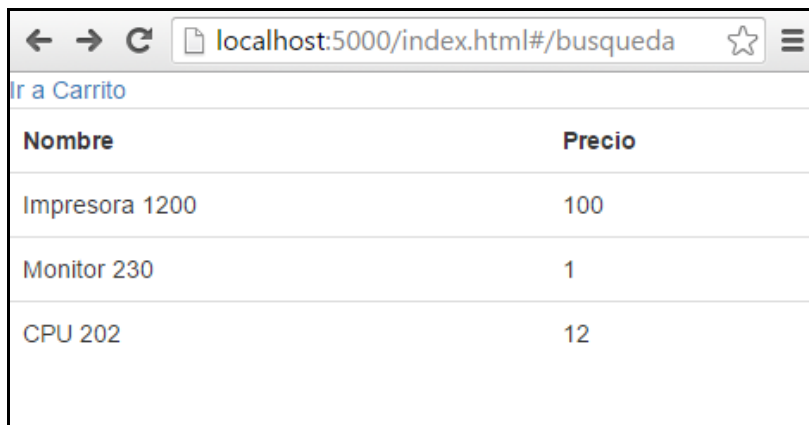
7. Tanto el controlador como el servicio son archivos, se deben referenciar en el archivo index.html.

```
<!DOCTYPE html>  
<html ng-app="awApp">  
<head>  
  <meta charset="utf-8" />  
  <title></title>  
  <link rel="stylesheet" href="lib/bootstrap/dist/css/bootstrap.min.css" />  
  <script type="text/javascript" src="lib/angular/angular.min.js"></script>  
  <script type="text/javascript" src="lib/angular-route/angular-route.min.js"></script>  
  <script type="text/javascript" src="spa/app.js"></script>  
  <script type="text/javascript" src="spa/controllers/ProductosController.js"></script>  
  <script type="text/javascript" src="spa/services/ProductosService.js"></script>  
</head>  
<body>  
  <div ng-view></div>  
</body>  
</html>
```

8. Ejecutar el proyecto y llamar al siguiente url.

<http://localhost:5000/index.html#/busqueda>

Con lo cual vamos a obtener el siguiente resultado:




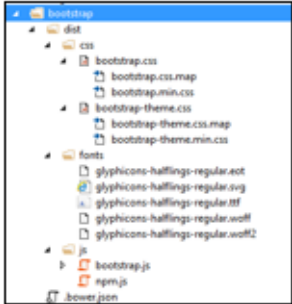
The screenshot shows a web browser window with the address bar displaying 'localhost:5000/index.html#/busqueda'. The page content includes a link 'Ir a Carrito' and a table with two columns: 'Nombre' and 'Precio'. The table lists three items: 'Impresora 1200' with a price of 100, 'Monitor 230' with a price of 1, and 'CPU 202' with a price of 12.

Nombre	Precio
Impresora 1200	100
Monitor 230	1
CPU 202	12

8. Estilizar la aplicación haciendo uso de Bootstrap

Estilizar la aplicación haciendo uso de Bootstrap

Bootstrap es un framework de estilos y de componentes en javascript para crear aplicaciones web responsive.
Url: <http://getbootstrap.com/>



```
EXAMPLE
Optional table caption.


| # | First Name | Last Name | Username |
|---|------------|-----------|----------|
| 1 | Mark       | Otto      | @mdo     |
| 2 | Jacob      | Thornton  | @fat     |
| 3 | Larry      | the Bird  | @twitter |

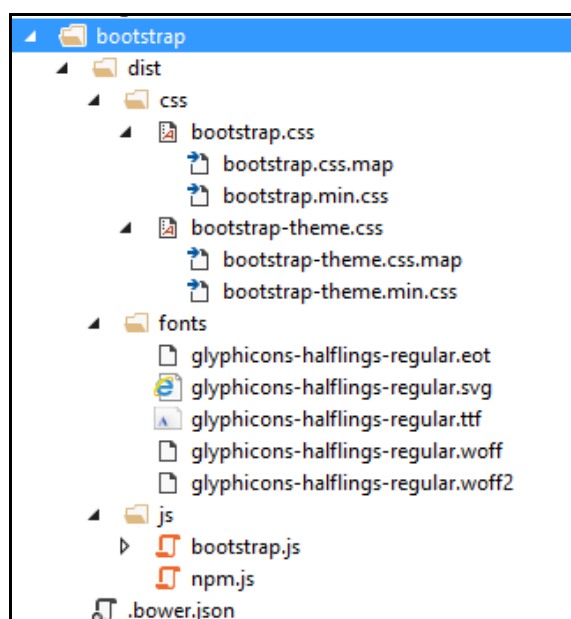

<table class="table">
  <tr>
    <th>#</th>
    <th>First Name</th>
    <th>Last Name</th>
    <th>Username</th>
  </tr>
  <tr>
    <td>1</td>
    <td>Mark</td>
    <td>Otto</td>
    <td>@mdo</td>
  </tr>
  <tr>
    <td>2</td>
    <td>Jacob</td>
    <td>Thornton</td>
    <td>@fat</td>
  </tr>
  <tr>
    <td>3</td>
    <td>Larry</td>
    <td>the Bird</td>
    <td>@twitter</td>
  </tr>
</table>
```

12 - 27 Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C.

Bootstrap es un framework de estilos y de componentes en javascript para crear aplicaciones web responsive.

Url: <http://getbootstrap.com/>

Para empezar a utilizar bootstrap solo es necesario referenciar las hojas de estilos, archivo javascript y fonts.



Por ejemplo, si deseo estilizar una tabla debo de realizar lo siguiente en el código:

EXAMPLE

Optional table caption.

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

```
<table class="table">  
...  
</table>
```

Copy

Como se puede observar para estilizar la tabla debo utilizar la CSS class con el nombre "table".