Programación Frontend en JavaScript

José Sánchez Moreno

January 21, 2014

Iniciación al desarrollo de frontends en javascript

- Básicamente veremos como desarrollar en javascript usando como back-end para los datos servidores REST.

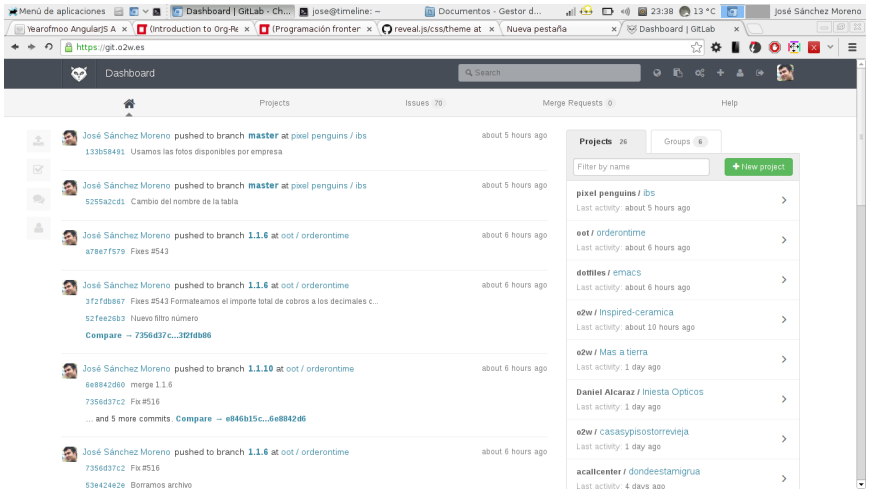
Herramientas que usaremos

- **Git** para el control de código
- **Sass** para la generación de estilos css
- **Python** / **Django** para la creación del servidor rest
- **AngularJS** como plataforma de desarrollo javascript

Git: Gitlab

- La encontraremos en `https://git.o2w.es`
- Lo básico:
- Descargamos el código con "git pull"
- Modificamos lo que queramos y después
- Realizamos un commit "git commit -am 'Descripción de los cambios'"
- Subimos los cambios con "git push"

Interfaz de colaboración



Sass

- Es un metalenguage sobre css que nos permite generar código css de manera más eficiente, fácil y ponente.
- Usaremos compass específicamente sobre sass

Código sass

```
.navbar .container-fluid.main-nav .nav-collapse .nav.menubar {
    p {
        @include respond-to(mobile) {
            display: inline-block;
        }
    }

}
```

- El Resultado "transpilado"

```
@media only screen and (max-width: 640px) {
  /* line 38, ../scss/dondeestamigrua.scss */
  .navbar .container-fluid.main-nav .nav-collapse .nav.menubar p {
    display: inline-block;
```

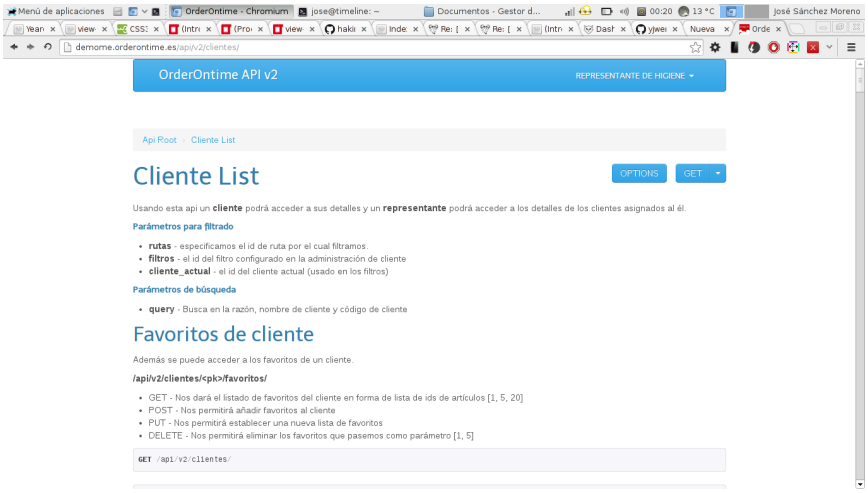
```
  }
}
```

Python y django

- Django nos permite crear servidores REST de manera muy rápida, potente y sencilla.

```
class FacturasViewSet(viewsets.GenericViewSet):

    def retrieve(self, request, pk=None):
        result = Tvh().get_factura(pk)
        return Response(result)

    @link()
    def lineas(self, request, pk=None):
        result = Tvh().get_lineas_de_factura(pk)
        return Response(result)

    @link()
    def consumos(self, request, pk=None):
        result = Tvh().get_consumos_de_factura(pk)
        return Response(result)


  class StockalmViewSet(BaseView):
     model = models.Stockalm
     serializer_class = serializers.StockalmSerializer
```

API



Rest

- Nos permite interactuar con los modelos de datos
- Descentralizado: los datos pueden estar en varios servidores
- Sencillo: basado en javascript (en realidad son objetos javascript), aunque es posible usar otros: xml, yaml, plist, etc
- Recursos: Colecciones y vista detalle.
- *clientes* -> devuelve la lista de clientes
- *clientes/1* -> devuelve los datos del cliente1
- Operaciones: Implementados sobre métodos HTTP
  - GET (consulta)
  - POST (creación)
  - PUT (modificación)
  - DELETE (borrado)

Ejemplo


```
[
    {
        "id": 1,
        "nomruta": "MURCIA",
        "ruta": "       1"
    },
    {
        "id": 2,
        "nomruta": "LEVANTE",
        "ruta": "       2"
    },
    {
        "id": 3,
        "nomruta": "CASTILLA-LAMANCHA",
        "ruta": "       3"
    }
]
```


Angular JS

- **Hay otros**:
  - Backone: Mucha más bajo nivel.
  - Ember: Más o menos a la par.
- Pero, **AngularJS** tiene una gran comunidad y está por Google

Características de AngularJS

- Módulos
- Vistas
- Directivas
- Servicios
- Filtros
- Controladores

- Data binding
- Enrutamiento
- Injección de dependencias
- Código testeable

Módulos

- Unidades de código para agrupar funcionalidades
- Tiene dos básicamente dos métodos "config" y "run"

```
angular.module('miModulo', []).
    config(function(injectables) { // provider-injector
     // Aquí se configura el módulo
   }).
     run(function(injectables) { // instance-injector
     // Este es el código de inicación
});
```

Rutas

- En la configuración del módulo podemos configurar nuestras rutas de la aplicación.
- Al cambiar la ruta, carga la plantilla en la directiva <ng-view> y le da el control al controlador asignado.

```
angular.module('miModulo', []).
config(function($routeProvider) {
    $routeProvider
            .when('/', {
                templateUrl: 'main.html',
                controller: 'MainCtrl'
            })
            .when('/clientes', {
                templateUrl: 'clientes.html',
                controller: 'ClientesCtrl'

            })
            .otherwise({redirectTo: '/'});
   });
```

Directivas

- Son etiquetas de html
- Angular tiene las suyas
  - ng-click
  - ng-href
  - select
  - input
  - ng-repeat
  - ng-view
  - ng-template
  - ng-switch
  - ng-if
  - Muchas más
- Además podemos crear las nuestras

Directivas

- Ejemplos de uso

```
<div ng-repeat="servicio in servicios | orderBy:'demora'" class="servicio">
{{ servicio.matricula }} ·
{{servicio.marca}} {{ servicio.modelo}} ·
{{servicio.cliente}} ·
</div>

<td ng-show="documento.tieneEcoraee()">{{ documento.ecoraee|moneda }}</td>

<ul>
  <li ng-repeat="(porcentaje, iva) in documento.ivas">
    {{ iva|moneda }}
     <span class="porcentaje">({{ porcentaje }}%)</span>
  </li>
</ul>
```

Directivas

- Nos sirve para abstraer agrupando vistas/ controlador y modelo en una directiva.

```
angular.module('cliente', [])
.directive('cliente', function() {
    return {
          restrict: 'E',
          scope: { clienteInfo: '=info' },
          templateUrl: 'cliente.html'
        };
  });
```

```
Nombre: {{ cliente.nombre }}
Dirección: {{ cliente.direccion }}
```

```
<cliente info="datosDeCliente"></cliente>
```

Filtros

- Modifican la presentación de los datos del modelo en la vista
  - Por ejemplo: Formateando valores, Filtrando listas, etc
- Escalares:
  - number
  - uppercase/lowercase
  - date
  - curreny
- Arrays
  - orderBy
  - limitTo
  - filter

  {{ expression | filter }}

Servicios

- **Son objetos o funciones que realizan una tarea específica.**
- Se definen y quedan disponibles para que sean injectados por medio de la injección de dependencias
- Son singletons

```
angular.module(
    "tvh.services",
    []
).factory("clientes",
    ["Restangular",
     function(Restangular) {
      return Restangular.all("clientes");
    }]
);
```

Controladores

- Son los que preparan los modelos de datos y dan vida a las vistas.

```
function MainCtrl($scope, clientes) {
    $scope.texto = 'Bienvenido al área de clientes';
    $scope.templateMenuUrl = utils.tpl('ui/menu');
    clientes.getList()
        .then(function(data) {
            $scope.clientes = data;
        });

    $scope.borrarCliente = function(cliente) {
        clientes.remove({id: cliente.id });
    }
}

angular.module('tvh.controllers.main', [])
        .controller('MainCtrl', [ '$scope', 'clientes', MainCtrl ]);
```

Vistas

- Son plantillas en html que son interpretadas por AngularJS

```
<ng-include src="templateMenuUrl"></ng-include>
<h1>{{ texto }}</h1>
<table class="table table-striped">
    <tr>
        <th>Nombre</th>
        <th>Provincia</th>
        <th>Dirección</th>
        <th>Población</th>
    </tr>
    <tr ng-repeat="cliente in clientes">
        <td>{{ cliente.titular }}</td>
        <td>{{ cliente.provincia }}</td>
        <td>{{ cliente.direccion }}</td>
        <td>{{ cliente.poblacion }}</td>
    </tr>
</table>
```

Módulos de terceros

- Restangular: Para crear clientes Rest fácilmente.
- AngularUI: Directivas extra, bootstrap
- AngularGM: Directivas para controlar google Maps
- ngmodules.org

Documentación y referencias

- `http://www.angularjs.org`
- `https://github.com/mgonto/restangular`
- `http://sass-lang.com/`
- `http://www.django-rest-framework.org/`