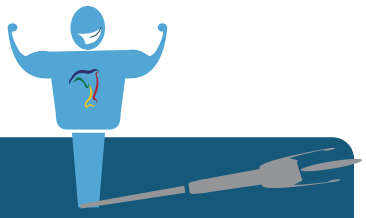
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

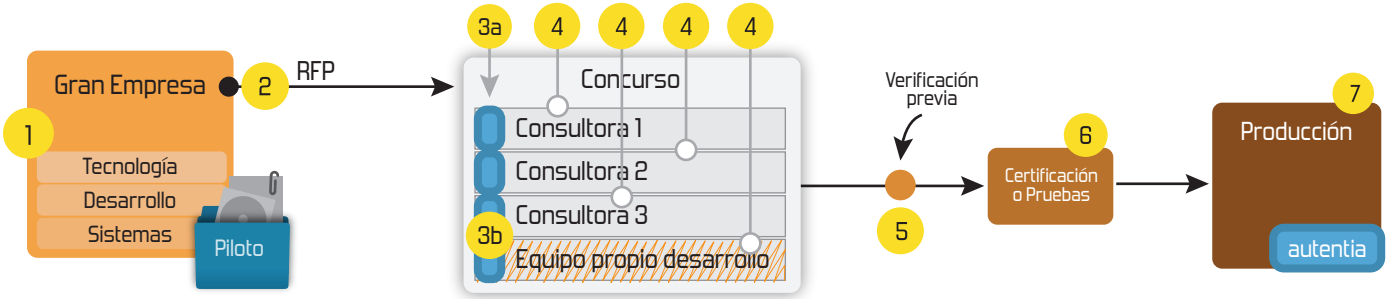
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.

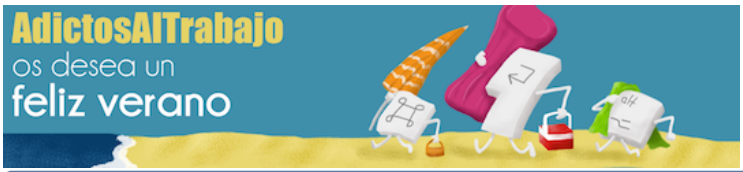


4. Cursos de formación (impartidos por desarrolladores en activo)

- Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery
- Gestor portales (Liferay)
- Gestor de contenidos (Alfresco)
- Aplicaciones híbridas
- Control de autenticación y acceso (Spring Security)
- UDDI
- Web Services
- Rest Services
- Social SSO
- SSO (Cas)
- JPA-Hibernate, MyBatis
- Motor de búsqueda empresarial (Solr)
- ETL (Talend)
- Dirección de Proyectos Informáticos.
- Metodologías ágiles
- Patrones de diseño
- TDD
- Tareas programadas (Quartz)
- Gestor documental (Alfresco)
- Inversión de control (Spring)
- BPM (jBPM o Bonita)
- Generación de informes (JasperReport)
- ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com



Entra en Adictos a través de
E-mail
Contraseña
Entrar
Registrarme
Olvidé mi contraseña

Inicio | Quiénes somos | Formación | Comparador de salarios | Nuestros libros | Más

» Estás en: Inicio » Tutoriales » Jugando con AngularJS y Google Maps

Miguel Arlandy Rodríguez

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/JEE

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2014-10-15

Tutorial visitado 28 veces Descargar en PDF

Jugando con AngularJS y Google Maps.

0. Índice de contenidos.

1. Introducción

Ya tuvimos la ocasión de hablar de AngularJS en anteriores tutoriales. Pese a ser un framework relativamente joven, el impacto que ha tenido en la comunidad de desarrollo web ha sido tremendo hasta el punto de que, a día de hoy, es probablemente uno de los temas que más interesan a los desarrolladores.

El API de Google para mapas es algo más veterano que Angular pero siempre es un recurso muy recurrido en determinados tipos de aplicativos. Con el API de Geolocalización de HTML5 o los dispositivos móviles sus posibilidades son incluso mayores.

En este tutorial aprenderemos a crear y manipular mapas de una manera muy sencilla en nuestros desarrollos web basados en AngularJS gracias a la librería de directivas angular-google-maps.

2. Entorno.

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 15' (2.2 Ghz Intel Core I7, 8GB DDR3).
- Sistema Operativo: Mac OS X Mavericks 10.9
- NetBeans IDE 8.0
- AngularJS 1.2.26
- angular-google-maps 1.2.2
- Google Chrome 37
- Mozilla Firefox 32
- Opera Browser 12.16
- Safari Browser 7

3. Angular Google Maps.

Angular Google Maps es una excelente **librería de directivas** AngularJS que nos permite integrar de una manera relativamente sencilla Google Maps en nuestros desarrollos AngularJS.

Aunque no es la única librería de este tipo, probablemente si que sea la que nos ofrezca una mayor funcionalidad a la hora de tratar los mapas. Será la librería que utilizaremos en este tutorial.

Catálogo de servicios Autentia

Síguenos a través de:

Últimas Noticias

» Curso JBoss de Red Hat

» Si eres el responsable o líder técnico, considérate desafortunado. No puedes culpar a nadie por ser gris

» Portales, gestores de contenidos documentales y desarrollos a medida

» Comentando el libro Start-up Nation, La historia del milagro económico de Israel, de Dan Senor & Salu Singer

» Screencasts de programación narrados en Español

Histórico de noticias

Últimos Tutoriales

» Extendiendo las reglas de SonarQube con Xpath

» Primeros pasos con Apache Kafka

» Solución de problemas comunes con la integración de maven en Eclipse Luna.

» Integración de SonarQube en Eclipse.

» Monitorización de Apache

4. Configuración.

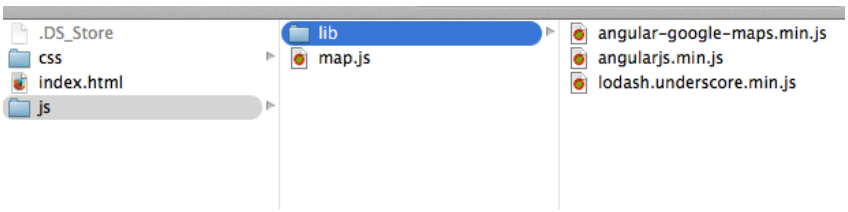
Existen dos formas de configurar nuestro proyecto: de manera manual o automática haciendo uso de bower. En nuestro caso lo haremos de forma manual.

Para ello, en la raíz de nuestro proyecto crearemos un directorio **js/** y en su interior un subdirectorio **lib/** donde meteremos las **librerías que necesitará nuestro proyecto**. Son las siguientes:

- AngularJS: en nuestro caso utilizaremos la versión 1.2.26
- angular-google-maps: versión 1.2.2
- lodash.underscore.min.js: librería de la que hace uso angular-google-maps. Versión 2.4.1.

Además, necesitaremos hacer uso del API Javascript para Google Maps.

Por último, crearemos en el directorio js/ un fichero **map.js** donde crearemos nuestro módulo para controlar el mapa (AngularJS) y, a la altura del directorio js/, un fichero index.html.



Para poder empezar a trabajar, nuestro fichero index.html quedaría así:

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8">
5          <title>AngularJS / Google Maps Tutorial
6
7          <!-- AngularJS -->
8          <script src="js/lib/angularjs.min.js?v=1.2.26"></script>
9
10         <!-- Google Maps Javascript API -->
11         <script src="http://maps.google.com/maps/api/js?sensor=false"></script>
12
13         <!-- angular-google-maps -->
14         <script src="js/lib/lodash.underscore.min.js?v=2.4.1"></script>
15         <script src="js/lib/angular-google-maps.min.js?v=1.2.2"></script>
16
17         <!-- Custom angular module -->
18         <script src="js/map.js?v=1.0"></script>
19     </head>
20     <body>
21
22     </body>
23 </html>
```

5. Creando nuestro mapa.

Una vez que ya tenemos todo lo necesario para empezar a trabajar, vamos a crear un sencillo mapa. Nuestro mapa tendrá un único marker (un marcador en un punto de coordenadas concreto del mapa), un zoom por defecto y estará centrado en las coordenadas de dicho marker.

Para ello creamos en nuestra vista (fichero .html) el mapa gracias a la directiva: **google-map** y añadimos en su interior el marcador mediante **marker**.

```
1  <!DOCTYPE html>
2  <html ng-app="AngularGoogleMap" >
3      <head>
4          <meta charset="utf-8"/>
5          <title>AngularJS / Google Maps Tutorial</title>
6          <!-- AngularJS -->
7          <script src="js/lib/angularjs.min.js?v=1.2.26"></script>
8
9          <!-- Google Maps -->
10         <script src="http://maps.google.com/maps/api/js?sensor=false"></script>
11
12         <!-- angular-google-maps -->
13         <script src="js/lib/lodash.underscore.min.js?v=2.4.1"></script>
14         <script src="js/lib/angular-google-maps.min.js?v=1.2.2"></script>
15
16         <!-- Custom angular module -->
17         <script src="js/map-module.js"></script>
18         <style>
19             .angular-google-map-container {
20                 position: absolute;
21                 top: 0;
22                 bottom: 0;
23                 right: 0;
24                 left: 0;
25             }
26         </style>
27     </head>
28     <body ng-controller="MapCtrl">
29     <google-map center="map.center"
30                 zoom="map.zoom"
31                 draggable="true"
32                 options="map.options"
33                 control="map.control">
34         <marker coords="marker.coords" options="marker.options" idkey="marker.id" ></mar
35     </google-map>
36 </body>
37 </html>
```

Como podemos observar en las líneas 29 y 34, hemos añadido un elemento **google-map** que contiene otro elemento **marker**, que no son más que **directivas** de la librería angular-google-maps que hemos añadido en la línea 14.

Tomcat con psi-probe.

Últimos Tutoriales del Autor

» Phonegap/Cordova y las Notificaciones Push

» Configurando Notificaciones Push para desarrollos Android con Google Cloud Messaging.

» Introducción a WSO2 API Manager

» SOA vs. SOAP y REST

» REST, el principio HATEOAS y Spring HATEOAS

Categorías del Tutorial

Javascript / JQuery

Dentro de la directiva google-map destacamos las siguientes propiedades:

- **center:** expresión que, al ser evaluada, debe tomar el valor de un objeto con las propiedades: "latitude" y "longitude", que representarán las coordenadas del centro de nuestro mapa.
- **zoom:** valor o expresión que indicará el nivel de zoom de nuestro mapa. Debe ser un número entre 1 y 20.
- **dragable:** booleano que indicará si podemos arrastrar el mapa.
- **options:** expresión que, al ser evaluada, tomará el valor de un objeto con propiedades adicionales del mapa. Se corresponden con las que ofrece el API de Google.
- **control:** propiedad que vincularemos a un objeto en nuestro controlador (vía $scope) y que nos proporcionará métodos de utilidades como obtener una instancia del mapa o refrescarlo.

Y en la directiva **marker** observamos estas propiedades:

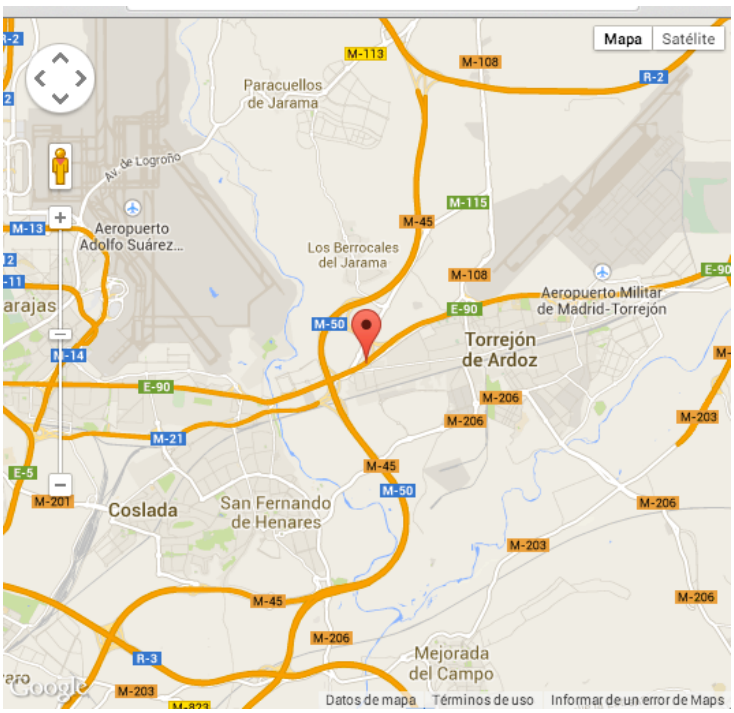
- **coords:** expresión que deberá resolverse del mismo modo que la propiedad center (latitud y longitud) que vimos anteriormente y que indicará las coordenadas donde se posicione nuestro marcador.
- **idkey:** identificador único que tendrá nuestro marcador.
- **options:** expresión que, al ser evaluada, devolverá opciones adicionales que proporcionaremos a nuestro marcador.

Ya tenemos todo lo que necesitamos en nuestra vista. Ahora **vincularemos el mapa a nuestro controlador**. Para ello crearemos un **fichero map-module.js** con un módulo llamado "AngularGoogleMap" y un controlador "MapCtrl" a los que referenciábamos anteriormente con las directivas **ng-app** y **ng-controller**.

```
var app = angular.module('AngularGoogleMap', ['google-maps']);

app.controller('MapCtrl', ['$scope', function ($scope) {

    $scope.map = {
        center: {
            latitude: 40.454018,
            longitude: -3.509205
        },
        zoom: 12,
        options : {
            scrollwheel: false
        },
        control: {}
    };
    $scope.marker = {
        id: 0,
        coords: {
            latitude: 40.454018,
            longitude: -3.509205
        },
        options: {
            draggable: true
        }
    };
}]);
```

Como vemos lo único que hacemos es **vincular**, a través el objeto $scope, las propiedades de nuestras directivas google-map y marker a nuestro controlador mediante un objeto "map" y otro objeto "marker".

Y con esto ya tendríamos nuestro mapa.



6. Añadiendo markers.

Vamos a complicar ligeramente la cosa creando un mapa que nos permita **añadir marcadores dinámicamente** a nuestro mapa.

Lo primero que haremos será sustituir la directiva **marker** por **markers**. Nuestro mapa en la vista quedaría de la siguiente forma.

```
<google-map center="map.center"
            zoom="map.zoom"
            draggable="true"
            options="map.options"
            control="map.control">
    <markers models="map.markers" coords="'self'" options="'options'" isLabel="true">
    </markers>
```

```
</google-map>
```

Observamos que la directiva markers viene con algunas propiedades:

- **models:** expresión que, al ser evaluada, debe referenciar un array de markers que serán añadidos al mapa.
- **coords:** cadena de caracteres que indica el nombre de la propiedad de cada objeto marker del array que contiene las coordenadas del marcador. La palabra reservada "self" indica que, tanto la latitud (latitude) como la longitud (longitude) del marcador están directamente como propiedades el objeto marker (no contenidas en otro objeto).
- **options:** expresión que, al ser evaluada, devolverá opciones adicionales que proporcionaremos a cada marcador.
- **isLabel:** expresión que, al ser evaluada, nos devolverá un booleano indicando si nuestros markers pueden llevar etiqueda con texto o no.

6.1 Por coordenadas.

Para añadir un marcador por coordenadas hacemos algo como lo siguiente:

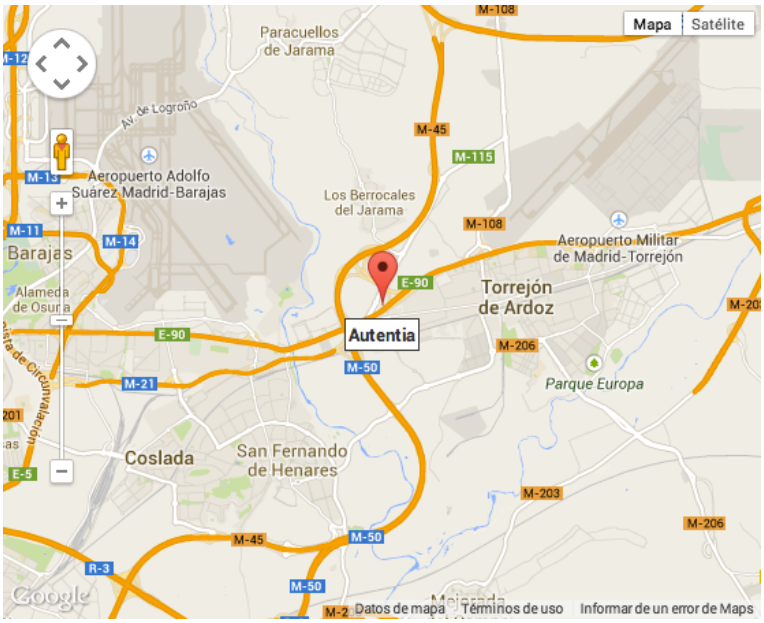
```
app.controller('MapCtrl', ['$scope', function ($scope) {

    var markerId = 0;

    function refresh(marker) {
        $scope.map.control.refresh({latitude: marker.latitude,
        longitude: marker.longitude});
    }

    function create(latitude, longitude) {
        var marker = {
            options: {
                animation: 1,
                labelAnchor: "28 -5",
                labelClass: 'markerlabel'
            },
            latitude: latitude,
            longitude: longitude,
            id: ++markerId
        };
        return marker;
    }

    function invokeSuccessCallback(successCallback, marker) {
        if (typeof successCallback === 'function') {
            successCallback(marker);
        }
    }

    function createByCoords(latitude, longitude, successCallback) {
        var marker = create(latitude, longitude);
        invokeSuccessCallback(successCallback, marker);
    }

    createByCoords(40.454018, -3.509205, function (marker) {
        marker.options.labelContent = 'Autentia';
        $scope.autentiaMarker = marker;
        refresh(marker);
    });
    $scope.map = {
        center: {
            latitude: $scope.autentiaMarker.latitude,
            longitude: $scope.autentiaMarker.longitude
        },
        zoom: 12,
        markers: [],
        control: {},
        options: {
            scrollwheel: false
        }
    };

    $scope.map.markers.push($scope.autentiaMarker);

}]);
```

Podemos destacar varias cosas en el anterior código.

Observamos que en el objeto map de nuestro $scope hemos definido un **array de markers**, al que tendremos que añadir los diferentes marcadores que queremos que aparezcan en nuestro mapa.

Observamos que tenemos una función **createByCoords** que recibirá la latitud y la longitud de nuestro marcador además de una función de callback que recibirá nuestro marker y en la que decidiremos qué hacer con él. Lógicamente lo que haremos será presentarlo en el mapa añadiéndolo al array de markers.

Por último, podríamos destacar la función create que recibe una latitud y una longitud a partir de las cuales, construirá nuestro marcador. Observamos que tiene **diferentes propiedades**, entre las que destacan: **labelAnchor** y **labelClass**, que nos permitirán que podamos añadir una etiqueta a nuestro marcador. El contenido de la etiqueta se lo indicaremos con la propiedad **labelContent**. Observamos que la función está preparada para **devolver un identificador de marcador distinto** en cada uno que creemos. **Si creamos varios marcadores con el mismo identificador sólo se añadirá uno al mapa**.



6.2 Por posición actual.

Siguiendo un poco la línea de lo que hemos hecho en el punto anterior y haciendo uso del API de Geolocalización de HTML5 podríamos obtener nuestra posición actual y marcarla en el mapa.

```
function createByCurrentLocation(successCallback) {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(function (position) {
            var marker = create(position.coords.latitude, position.coords.longitude);
            invokeSuccessCallback(successCallback, marker);
        });
    } else {
        alert('Unable to locate current position');
    }
}

createByCurrentLocation(function (marker) {
    marker.options.labelContent = 'You´re here';
    $scope.map.markers.push(marker);
    refresh(marker);
});
```

6.3 Por dirección.

Y por último, gracias al servicio de geocodificación que nos ofrece el API Javascript de Google Maps, podríamos obtener un marcador por una dirección dada. Ej: Concha Espina 1, Madrid.

```
function createByAddress(address, successCallback) {
    var geocoder = new google.maps.Geocoder();
    geocoder.geocode({'address' : address}, function (results, status) {
        if (status === google.maps.GeocoderStatus.OK) {
            var firstAddress = results[0];
            var latitude = firstAddress.geometry.location.k;
            var longitude = firstAddress.geometry.location.B;
            var marker = create(latitude, longitude);
            invokeSuccessCallback(successCallback, marker);
        } else {
            alert("Unknown address: " + address);
```

```
            }
        });
    }
```

Podéis VER EL EJEMPLO EN FUNCIONAMIENTO AQUÍ.

7. Referencias.

- Ver ejemplo en funcionamiento
- Código fuente del ejemplo.
- Documentación angular-google-maps.
- AngularJS: primeros pasos.

8. Conclusiones.

En este tutorial hemos visto lo sencillo que es trabajar con Google Maps y AngularJS gracias a la librería de directivas angular-google-maps. Si no estás muy acostumbrado a trabajar con AngularJS debes saber que Angular es bastante "celoso" con el uso de librerías externas que manipulen o interactuen con elementos del DOM. En vez de utilizar directamente estas librerías, la recomendación general suele ser hacerlo mediante el uso de directivas de AngularJS.

Cualquier duda en la sección de comentarios.

Espero que este tutorial os haya sido de ayuda. Un saludo.

Miguel Arlandy

marlandy@autentia.com

Twitter: @m_arlandy

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

Share | 0

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

» Registrate y accede a esta y otras ventajas **«**

SOME RIGHTS RESERVED Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

PUSH THIS Page Pushers Community Help?

---- no clicks

0 people brought clicks to this page

powered by karmacracy

Copyright 2003-2014 © All Rights Reserved | Texto legal y condiciones de uso | Banners | Powered by Autentia | Contacto

XHTML 1.0 CSS XML RSS XML ATOM