

# ¿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

Tareas programadas (Quartz)  
Gestor documental (Alfresco)  
Inversión de control (Spring)

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

BPM (jBPM o Bonita)  
Generación de informes (JasperReport)  
ESB (Open ESB)

AdictosAlTrabajo

Terrakas 1x05  
¡¡Ya está en la web!! :-)  
terrakas.com

**autentia**  
 Soporte a desarrollo informático  
 Hosting patrocinado por  
**enredados**

 Entra en Adictos a través de    
 E-mail:   
 Contraseña:   
 Entrar Deseo registrarme  
Olvidé mi contraseña
[Inicio](#) [Quiénes somos](#) [Formación](#) [Comparador de salarios](#) [Nuestros libros](#) [Más](#)  

» Estás en: Inicio » Tutoriales » AngularJS: primeros pasos.

**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](#)**Catálogo de servicios Autentia**

www.autentia.com

**Fecha de publicación del tutorial: 2013-02-21**Tutorial visitado 1 veces [Descargar en PDF](#)**AngularJS: primeros pasos.****0. Índice de contenidos.**

- 1. Introducción.
- 2. Entorno.
- 3. Características de AngularJS.
- 4. Ejemplo: creando un cuestionario con AngularJS.
  - 4.1 Definiendo las preguntas.
  - 4.2 Definiendo nuestro modelo de datos.
  - 4.3 Definiendo la vista y su vinculación con el modelo.
- 5. Referencias.
- 6. Conclusiones.

**1. Introducción**

En plena "era HTML5" y Web 2.0 pocos son los que discuten que Javascript y CSS3 son los reyes absolutos en lo que respecta al desarrollo "front-end". Los sitios web son mucho **más dinámicos** (algunas web's ya parecen incluso verdaderas aplicaciones de escritorio) de forma que la experiencia del usuario mejora notablemente.

HTML es un lenguaje para definir documentos estáticos y el camino para dinamizarlos pasa irremediablemente por Javascript. Sin embargo, este proceso de dinamización de documentos HTML puede ser una tarea bastante tediosa que, a menudo, se suele hacer bastante difícil de mantener si no se diseña correctamente.

En este tutorial intentaremos explicar qué es AngularJS y cómo intenta dinamizar los documentos HTML mediante la vinculación (data binding) del modelo de datos con componentes de la vista mediante un ejemplo.

**2. Entorno.**

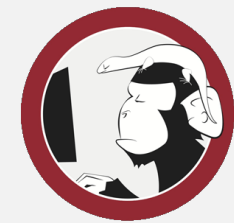
El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 15' (2.2 Ghz Intel Core i7, 8GB DDR3).
- Sistema Operativo: Mac OS Mountain Lion 10.8
- Entorno de desarrollo: [IntelliJ Idea 11.1 Ultimate](#).
- AngularJS 1.0.4

**3. Características de AngularJS.**

AngularJS es un **framework Javascript** que corre en el lado del cliente (navegador) y que se centra en intentar **dinamizar documentos HTML**, lo que comúnmente se conoce como DHTML (Dynamic HTML). Normalmente esto se consigue haciendo uso de CSS y Javascript de forma que, en función de los eventos que se produzcan en nuestra página (acciones del usuario, respuestas del servidor vía AJAX o Websockets), se actualizan, crean o eliminan determinados componentes de nuestro DOM (una imagen, un párrafo, o lo que sea...).

El objeto "document" de Javascript nos proporciona una serie de métodos como "getElementById" o "createElement" que nos

**Síguenos a través de:****Últimas Noticias**

» Ya está a la venta mi segundo libro: **Planifica tu éxito, de aprendiz a empresario**

» Ya está disponible en eBook mi primer libro: **Informática Profesional**

» Comentando el libro: **La inteligencia reformada, las inteligencias múltiples en el siglo XXI** de Howard Gardner

» Hangout "El precio de un servicio"

» Comentando el Libro HACER:¡NADA! DE J. Keith Murnighan

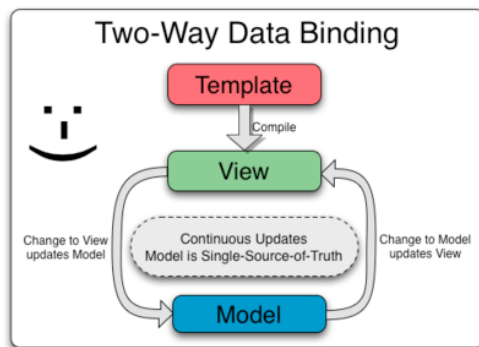
[Histórico de noticias](#)**Últimos Tutoriales**

» Gestión de expedientes en el ámbito de las Administraciones Públicas (II): requisitos.

» Introducción a OpenLayers: un visor de mapas javascript.

permiten manipular, crear o eliminar elementos de nuestro documento HTML. No obstante, esto suele ser una tarea relativamente tediosa y muchas veces compleja, sobre todo para la gente que no está acostumbrada a trabajar con Javascript (es cierto que JQuery facilita mucho el trabajo).

AngularJS cambia un poco el enfoque de "dinamización" de documentos HTML estáticos mediante la **vinculación de elementos de nuestro documento HTML con nuestro modelo de datos** (data binding). De este modo, definimos un modelo de datos (Javascript) que se corresponderá con determinadas partes de nuestro HTML y, siempre que haya cambios en una parte, automáticamente se verán reflejados en la otra.



Veamos un ejemplo para intentar asimilar mejor estos conceptos.

#### 4. Ejemplo: creando un cuestionario con AngularJS.

En este ejemplo vamos a crear un cuestionario con preguntas que deben ser respondidas por el usuario. Para ello partiremos de un set de preguntas que deben ser cargadas en el documento HTML. Además, según vaya respondiendo el usuario a cada pregunta, ésta se deberá actualizar de forma que sepa si ha respondido correctamente o no. Por último tendremos un indicador que muestre el rango del usuario en función de la cantidad de preguntas que va acertando.

Por tanto, nuestros **requisitos** son:

- Queremos que las **preguntas se carguen dinámicamente** en el documento.
- Por cada pregunta que responda el usuario **se debe indicar si la respuesta es correcta o no**.
- Se debe **actualizar el rango del usuario** en función de las preguntas que vaya acertando.

##### 4.1 Definiendo las preguntas.

En nuestro modelo de datos una pregunta se representaría de la siguiente forma:

```

1  {
2    id : 1,
3    text : 'Esto es una pregunta',
4    validAnswer : 1,
5    userAnswer : null,
6    status : '',
7    answers: [
8      {id : 1, text : 'Respuesta 1'},
9      {id : 2, text : 'Respuesta 2'},
10     {id : 3, text : 'Respuesta 3'}
11   ]
12 }
```

Las propiedades de nuestra pregunta son:

- **id**: identificador de la pregunta
- **text**: enunciado de la pregunta
- **validAnswer**: identificador de la respuesta correcta
- **userAnswer**: identificador de la respuesta seleccionada por el usuario (null si no ha respondido)
- **status**: indica si la pregunta se ha respondido de forma correcta o no
- **answers**: listado de posibles respuestas a la pregunta. Cada pregunta está compuesta por un id y un texto

##### 4.2 Definiendo nuestro modelo de datos.

Como explicamos anteriormente, AngularJS vincula nuestro modelo de datos con el documento HTML. Para conseguir esto, hace uso de un objeto **scope** donde se definen las propiedades del modelo.

```

1  function TestController($scope) {
2    $scope.questions = [
3      {
4        id : 1,
5        text:'Esto es una pregunta',
6        validAnswer : 1,
7        userAnswer : null,
8        status : '',
9        answers: [
10         {id : 1, text : 'Respuesta 1.1'},
11         {id : 2, text : 'Respuesta 1.2'},
12         {id : 3, text : 'Respuesta 1.3'}
13       ]
14     },
15     {
16       id : 2,
```

» Trabajar con tablas en JasperReport

» Gestión de expedientes en el ámbito de las Administraciones Públicas (I): ámbito funcional.

» CDI: Inyección de dependencias en JEE y ejecución de test de integración con el soporte de Arquillian.

#### Últimos Tutoriales del Autor

» Sonar y Javascript: obteniendo la cobertura de nuestro código

» Sonar y Total Quality: midiendo la calidad total de nuestros proyectos

» Servicios REST documentados y probados con Swagger

» Creación de plantillas DSL con Drools

» Introducción a Drools.

#### Últimas ofertas de empleo

2011-09-08

Comercial - Ventas - MADRID.

2011-09-03

Comercial - Ventas - VALENCIA.

2011-08-19

Comercial - Compras - ALICANTE.

2011-07-12

Otras Sin catalogar - MADRID.

2011-07-06

Otras Sin catalogar - LUGO.

```

17         text: 'Otra pregunta',
18         validAnswer : 2,
19         userAnswer : null,
20         status : '',
21         answers: [
22             {id : 1, text : 'Respuesta 2.1'},
23             {id : 2, text : 'Respuesta 2.2'}
24         ]
25     }
26 ];
27
28 $scope.userStatus = '';
29
30 $scope.validAnswers = 0;
31
32 $scope.validate = function (question) {
33     if (question.validAnswer == question.userAnswer) {
34         $scope.validAnswers ++;
35         question.status = 'ok';
36     } else {
37         if (question.status == 'ok' && $scope.validAnswers > 0) {
38             $scope.validAnswers --;
39         }
40         question.status = 'error';
41     }
42
43     updateUserStatus();
44 };
45
46 function updateUserStatus() {
47     var avgValidAnswers = ($scope.validAnswers / $scope.questions.length) * 100;
48     if (avgValidAnswers == 0) {
49         $scope.userStatus = 'looser';
50     } else if (avgValidAnswers == 100) {
51         $scope.userStatus = 'guru';
52     } else {
53         $scope.userStatus = 'poor';
54     }
55 }
56
57 }

```

Como podemos ver, hemos definido tres propiedades además de un método en nuestro modelo de datos:

- **questions**: Listado de preguntas con sus respectivas respuestas que serán cargadas dinámicamente en nuestro documento HTML.
- **userStatus**: Rango del usuario en función de las preguntas que se van respondiendo.
- **validAnswers**: Representa el número de respuestas acertadas por el usuario.
- **validate**: Valida una pregunta en función de la respuesta del usuario. Actualiza userStatus y validAnswers además del status del objeto question.

Si nos fijamos, el **método validate**, que será invocado cuando el usuario responda a una pregunta (lo veremos en el próximo punto), **actualiza las propiedades validAnswers, userStatus y question.status**. Los valores asignados a question.status (ok o error) están **representados en una hoja de estilos** del mismo modo que los valores "looser", "poor" y "guru" del estado del usuario. Esto hará que, cuando cambie el valor de estas propiedades, se cargue en el documento una regla distinta de la hoja de estilos con lo que el usuario apreciará un cambio en el documento.

✖ Estilo de una pregunta respondida erróneamente

✔ Estilo de una pregunta respondida correctamente



Usuario con rango "looser"



Usuario con rango "poor"



Usuario con rango "guru"

#### 4.3 Definiendo la vista y su vinculación con el modelo.

Pues ya lo único que queda es definir nuestro documento HTML como si fuese una plantilla, vincularla con nuestro modelo y hacer que AngularJS despliegue su "magia".

```

1 <!DOCTYPE HTML>
2 <html ng-app>
3 <head>

```

```

4      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
5      <title>Cuestionario con AngularJS</title>
6      <script src="js/angular-1.0.4.min.js"></script>
7      <script src="js/script.js?09022013"></script>
8      <link type="text/css" rel="stylesheet" href="css/test.css?09022013"/>
9    </head>
10   <body>
11     <div id="test" ng-controller="TestController">
12       <h1>Ejemplo de cuestionario utilizando AngularJS</h1>
13       
14       <div class="questions">
15         <div class="question" ng-repeat="question in questions">
16           <p class="status-{{question.status}}">{{question.text}}</p>
17           <div class="answers" ng-repeat="answer in question.answers">
18             <input type="radio" ng-model="question.userAnswer" value="{{answer.id}}">
19               name="question-{{question.id}}" id="answer-{{question.id}}-{{ansv
20             ng-change="validate(question)"/>
21             <label for="answer-{{question.id}}-{{answer.id}}">{{answer.text}}</label>
22           </div>
23         </div>
24       </div>
25       <div class="userStatus {{userStatus}}">
26         <span>Has acertado {{validAnswers}} de {{questions.length}}</span>
27       </div>
28     </div>
29   </body>
30 </html>

```

Pues lo que estamos viendo es todo lo que necesitamos para construir nuestra vista y que, además tenga un comportamiento dinámico. Expliquemos algunos puntos:

- **Línea 2:** vemos que el tag html está marcado con la directiva **ng-app**. Lo que se está haciendo es definir el ámbito sobre el que AngularJS actuará. En nuestro ejemplo sería en todo el documento HTML.
- **Línea 11:** vemos que el elemento div tiene una directiva **ng-controller** con valor igual a **TestController**. Con esto estamos definiendo el controlador que actuará sobre el elemento y sus nodos hijo. Si nos fijamos el valor "TestController" coincide con la función donde definimos las propiedades de nuestro modelo y su comportamiento. Para el que conozca JSF sería algo como configurar el Back Bean que hay detrás de la vista.
- **Línea 15:** vemos que hay un div con una directiva **ng-repeat**. Lo que hacemos es repetir ese elemento div (y sus nodos hijo) por cada "question" que tenemos definidas en el modelo (propiedad questions). Además define una variable question (la pregunta concreta) que podrá ser utilizada en ese ámbito. Si nos fijamos en la siguiente línea, observamos que podemos acceder directamente a las propiedades de question (recordemos que es un elemento del modelo) mediante la sintaxis **{{}}**. En concreto obtenemos los valores de las propiedades status y text.
- **Línea 18:** Observamos que se están pintando los elementos input (radio-buttons) que representan las posibles respuestas de un usuario a una pregunta. Con la directiva **ng-model** igual a **question.userAnswer** lo que hacemos es vincular el valor de este elemento, que si nos fijamos es el id de la respuesta, con la propiedad **userAnswer** que teníamos definida en las preguntas de nuestro modelo. Con esto, cuando el usuario seleccione una pregunta, el id de la pregunta seleccionada se vinculará automáticamente con dicha propiedad.
- **Línea 20:** en ese mismo elemento input definimos una directiva **ng-change** con valor igual a **validate(question)**. Lo que haremos será invocar al método validate que definimos en el modelo cuando se seleccione una respuesta. Con esto actualizaremos el número total de respuestas acertadas (validAnswers) y el rango del usuario (userStatus). Este cambio en el modelo se verá automáticamente reflejado en la vista.

Pues con esto ya tendríamos nuestro cuestionario dinámico implementado con AngularJS :-).

## Ejemplo de cuestionario utilizando AngularJS

✗ ¿Cómo declaramos en Java un atributo de una clase que solo pueda ser accedido desde la propia clase?

- ☐ Con "private"  
☐ Con "public"  
☒ Con mucho cariño  
☐ ¡Eso no se puede hacer!

Has acertado 0 de 3



♥ ¿Qué significan las siglas del estado MVC?

[VER EJEMPLO EN FUNCIONAMIENTO.](#)

### 5. Referencias.

- AngularJS: Guía para desarrolladores

### 6. Conclusiones.

En este tutorial hemos visto cómo dinamizar nuestras páginas HTML en el lado del cliente con ayuda del framework Javascript: AngularJS. Este tipo de frameworks (algunos los llaman MVC, otros MVVM) nos permiten separar la lógica de presentación de la vista mediante la vinculación de datos del modelo con componentes HTML.

Bajo mi punto de vista, AngularJS proporciona dos grandes ventajas:

- **Separación de la lógica de presentación de la vista**, de forma que el código Javascript es independiente al código HTML por lo que podríamos rehacer la vista sin necesidad de tener que tocar ni una sola línea de nuestra lógica de negocio. Además, nuestro código es extremadamente fácil testear ya que no dependemos de ningún contexto externo (es lógica pura).
- **Facilidad a la hora de manipular los elementos del DOM** y sus propiedades. Esta es una tarea que, por norma

general, es bastante tediosa y a medida que la lógica de presentación crece suele ser complicado escribir código que pueda mantenerse con facilidad. AngularJS simplifica mucho la labor en este sentido.

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

Miguel Arlandy

[marlandy@autentia.com](mailto:marlandy@autentia.com)

Twitter: [@m\\_arlandy](https://twitter.com/m_arlandy)

### A continuación puedes evaluarlo:

[Regístrate para evaluarlo](#)



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



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

» [Regístrate](#) y accede a esta y otras ventajas «



Esta obra está licenciada bajo licencia [Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

Copyright 2003-2013 © All Rights Reserved | [Texto legal y condiciones de uso](#) | [Banners](#) | [Powered by Autentia](#) | [Contacto](#)

