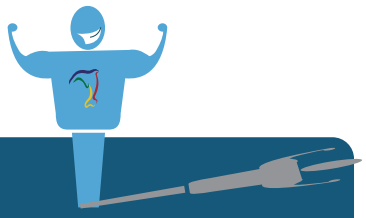
autentia

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...

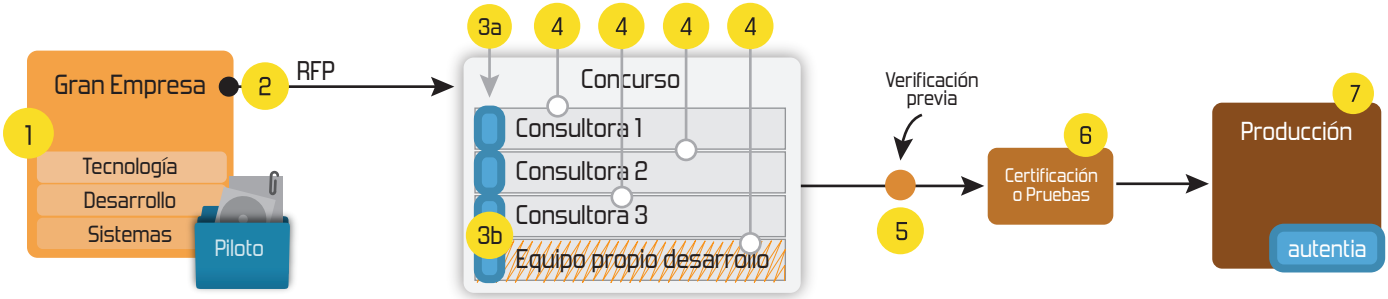
1. Desarrollo de componentes y proyectos a medida



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

DESARROLLADO POR:
Carlos García Pérez

Técnico especialista en informática de empresa (CEU).

Ingeniero Técnico en Informática de Sistemas (UPM)

Creador de MobileTest, Haaala!, Girillo, toi18n.

Charla sobre desarrollo de aplicaciones en Android.

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

Fecha de publicación del tutorial: 2011-10-11

69

Validación de formularios con el plugin JQuery Validator

Índice de contenidos

1. Introducción
2. Requisitos previos
3. Características principales de JQuery Validator
4. Ejemplo de instalación y uso de JQuery Validator
5. Otros métodos de especificar las reglas de validación
6. Conclusiones

Introducción

En el mundo de desarrollo de aplicaciones Web, independientemente de las tecnologías de servidor utilizadas *J2EE (Java), PHP, .Net, etc.*, la libería JQuery así como sus múltiples Plugins JQuery están siendo cada día más utilizados en la parte cliente (el navegador de Internet de tu pc o dispositivo móvil).

Si mi aplicación o portal web tiene como requisito tener activado **JavaScript** para funcionar, aprovechémoslo y usemos los recursos (CPU, memoria) de las máquinas cliente y descargemos a los servidores, disminuyamos el tráfico de red entre cliente y servidor en la medida de lo posible y hagamos tener una experiencia más positiva de navegación y usabilidad al usuario.

Desde su nacimiento, JavaScript fué principalmente pensado para hacer **validaciones de datos en la parte cliente** (que eso no excluye que también haya que hacerlo en la parte servidor), así como ejecutar funciones y tratamientos gráfica en la parte cliente.

En este tutorial, explicaremos como usar la librería JQuery y el plugin de validación JQuery Plugin mediante un ejemplo que veremos a continuación.

Requisitos previos

Catálogo de servicios Autentia

Últimas Noticias

Mis impresiones sobre la Apache Barcamp Spain 2011 - Alejandro Pérez García

Próximo lanzamiento de la serie iTerrakas!

Material de la charla de TDD con Objective-C en la XPWeek 2011

Actualizamos el tutorial de tramites administrativos tras el nacimiento de un hijo

Autentia patrocina la CAS2011

Histórico de NOTICIAS

Últimos Tutoriales

Cómo incluir un botón personalizado para nuestro CMS en la barra de menú de TinyMCE

El patrón de diseño Template Method

Clean Code: reglas y principios

Token con

Se requieren conocimientos previos de JQuery, DOM y JavaScript.

Características principales de JQuery Validator

- Configuración flexible y fácil.
- Aproximadamente 20 reglas de validación preconstruidas (requerido, email, números, fechas, etc.) y posibilidad de crear nuevas reglas reutilizables.
- Mensajes de error de validación muy personalizables.
- Especificación de las reglas de validación de forma no intrusiva (no hace falta modificar el html) o modificando mediante clases css (ejemplo class="required date")

Ejemplo de instalación y uso de JQuery Validator

A continuación vamos a configurar JQuery Validator para validar el siguiente formulario de una forma **no intrusiva** aplicando las siguientes reglas:

- *login:* Requerido, además no debe existir un usuario con ese mismo login. (Ej: **cgarcia** está ocupado)
- *Contraseña:* Requerido, además deben tener al menos 4 caracteres y coincidir la conraseña con su verificación.
- *Nombre completo:* Requerido.
- *E-Mail:* Requerido, y debe tener un formato de email válido.
- *Página web* Opcional pero si existe el formato debe ser correcto.
- *Fecha de nacimiento:* Opcional pero si existe se validará que es una fecha correcta.
- *Nº años de antigüedad:* Requerido y número positivo entre 1 y 50.
- *Nº personas a su cargo:* Requerido, número entre 0 y 1000.
- *Secreto:* Requerido, es una regla personalizada que valida que el usuario introduzca la respuesta correcta.
- *CampoX:* Es opcional a menos que se haga click en el enlace que establece una regla dinámica sobre el, convirtiéndolo en un campo requerido.

Haz clic aquí **para ver e interactuar con el ejemplo.**

caducidad en Spring Security

Creación de un componente en JSF2: separando la renderización del propio componente

Últimos Tutoriales del Autor

Instalación y uso del plugin de comentarios de Facebook en nuestra Web

Construcción personalizada de objetos JSON en cliente (JavaScript)

MyBatis Generator (MGB): Generador de código para MyBatis e iBATIS

Android: Leer correos de Gmail

Construcción de un control personalizado en Android

Síguenos a través de:

Últimas ofertas de empleo

2011-07-06
Otras Sin catalogar - LUGO.

2011-06-20
Comercial - Ventas - SEVILLA.

2011-05-24
Contabilidad - Expecialista Contable - BARCELONA.

2011-05-14
Comercial - Ventas - TARRAGONA.

2011-04-13
Comercial - Ventas - VALENCIA.

```
view plain  print  ?
01. <form id="entityDataForm" action="#" method="post">
02.     <fieldset>
03.         <legend>Datos del Usuario</legend>
04.         <div class="field">
05.             <label for="login">Login:</label>
06.             <input type="text" id="login" name="login" size="20" maxlength=
07.         </div>
08.         <div class="field">
09.             <label for="pass">Contraseña: (dos veces)</label>
10.             <input type="password" id="pass"  name="pass"  size="5"/>
11.             <input type="password" id="pass2" name="pass2" size="5"/>
12.         </div>
13.         <div class="field">
14.             <label for="name">Nombre completo:</label>
15.             <input name="name" id="name" size="35" maxlength="100"/>
16.         </div>
17.         <div class="field">
18.             <label for="email">E-Mail:</label>
19.             <input type="text" id="email" name="email" size="20" maxlength=
20.         </div>
21.         <div class="field">
22.             <label for="website">Página web:</label>
23.             <input name="website" id="website" size="35" maxlength="100"/>
24.         </div>
25.         <div class="field">
26.             <label for="fnac">Fecha de nacimiento (dd/mm/aaaa):</label>
27.             <input type="text" name="fnac" id="fnac" size="20"/>
28.         </div>
29.         <div class="field">
30.             <label for="antiguedad">N° años de antigüedad:</label>
31.             <input type="text" name="antiguedad" id="antiguedad" />
32.         </div>
33.         <div class="field">
34.             <label for="numPersonas">N° personas a su cargo:</label>
35.             <input type="text" name="numPersonas"  id="numPersonas" size="1
36.         </div>
37.         <div class="field">
38.             <label for="secreto">Secreto: ¿5+5?:</label>
39.             <input type="text" name="secreto"  id="secreto"/>
40.         </div>
41.         <div class="field">
42.             <label for="campoX">CampoX:</label>
43.             <input type="text" name="campoX"  id="campoX"/>
44.         </div>
45.     </fieldset>
46.     <div id="entity_commands">
47.         <input type="submit" value="Aceptar"/>
48.         <input type="submit" value="Cancelar" class="cancel"/>
49.
50.         <a href="#" id="manual">Validacion manual</a>
51.         <a href="#" id="addRule">Agrega una regla a CampoX</a>
52.     </div>
53. </form>
```

Debes descargarte las librerías JQuery y JQuery Validator e incluirlas en las páginas Web las librerías

```
view plain  print  ?
01. <!-
    - Incluimos las librerías en la página que contiene el formulario a valida
    ->
02. <script type="text/javascript" src="TU_PATH/jquery-
    1.6.4.min.js"></script>
03. <script type="text/javascript" src="TU_PATH/jquery.validate.min.js"></scri
04.
05. <!-
    - Mensajes de validación por defecto que serán mostrados si el programador
    ->
06. <!-
    - cada regla/campo concreto mediante la propiedad messages que veremos post
    ->
07. <script type="text/javascript" src="TU_PATH/messages_es.js"></script>
08.
```

Cuando ya esté cargado el DOM activamos haciendo uso de JavaScript la función de validación sobre el formulario.

```
view plain  print  ?
01. $(document).ready(function() {
02.     <!--      Activamos la validación sobre el formulario -->
03.     $("#entityDataForm").validate({
04.         errorContainer: "#errores",
05.         errorLabelContainer: "#errores ul",
06.         wrapper: "li",
07.             errorElement: "em",
08.             rules: {
09.                 login:   {required: true, remote: {url: "check-
    login.php", type: "get"}},
10.                 pass:    {required: true, minlength: 4},
11.                 pass2:   {required: true, minlength: 4, equalTo: "#pass"},
12.                 name:    {required: true},
13.                 email:   {required: true,  email: true},
14.                 website: {required: false, url: true},
15.                 fnac:    {required: false, date: true},
16.                 antiguedad:  {required: true, number: true, min: 1, max: 5(
17.                 numPersonas: {required: true, range: [0, 1000]},
18.                 secreto:     {basicoCaptcha: 10}
19.             },
20.         messages: {
21.             login:   {
22.                 required: "Campo requerido: Login",
23.                 remote:   "Ya existe un usuario con ese login"
24.             },
25.             email:       {
26.                 required: "Campo requerido: E-Mail",
27.                 email:    "Formato no valido: E-Mail"
28.             },
29.             secreto: {
30.                 basicoCaptcha: "Introduzca el secreto"
31.             }
32.             }
33.     });
34.     $.validator.methods.basicoCaptcha = function(value, element, param) {re
35.
36.     <!-
    -  Cuándo hagamos clic en el enlace manual mostraremos el número de errores
    ->
37.     $("#manual").click(function() {
38.           alert("¿Formulario válido?: " + $("#entityDataForm").validate().i
39.           alert("Existen " + $("#entityDataForm").validate().numberOfInval
40.     });
41.
42.     <!-
    -  Cuándo hagamos clic en el enlace addRule agregaremos una regla de valida
    ->
43.     $("#addRule").click(function() {
44.             $("#campoX").rules("add", {
45.                 required: true, minlength: 3,
46.                 messages: {
47.                     required: "Ahora el campo es requerido",
48.                     minlength: jQuery.format("Son necesarios al menos {0} c
49.                 }
50.             });
51.     });
52. });
```

En el código fuente Javascript anterior podemos observar:

- Mediante la propiedad ***rules*** especificamos las reglas de validación que tienen que cumplir cada campo referenciándolo por su "id".
- Mediante la propiedad ***messages*** especificamos los mensajes que deseamos mostrar al usuario cuando el campo no cumpla la regla de validación.
  Si no se especifican se mostrarán los mensajes de error de por defecto.
- Mediante las propiedades ***errorContainer, errorLabelContainer, wrapper, errorElement*** especificamos dónde y cómo queremos formatear los mensales de error.
  Son opcionales y en caso de no especificarse, los errores se muestran en rojo al lado del campo con errores de validación.
- El campo cuya etiqueta es "Secreto: ¿5+5?" tiene una regla de validación personalizada que valida que el contenido sea igual a un valor concreto. Para crear la regla se ha usado el código fuente:
  `$.validator.methods.basicoCaptcha = function(value, element, param) {return value == param;};`

Código fuente de la implementación de la funcionalidad en la parte servidora que valida la regla remote y valida que el login no esté ya siendo usado por otro usuario.

En PHP:

```
<?php
```

```
        $isLoginAvailable = "true";

        if (strcasecmp($_GET['login'], "cgarcia") == 0) {
                $isLoginAvailable = "false";
        }

        header('Content-type: application/json');
        echo $isLoginAvailable;
?>
```

En Java (un servlet mapeado sobre la url check-login.php)

```
view plain  print  ?
01. package es.carlosgarcia;
02.
03. import java.io.IOException;
04.
05. import javax.servlet.ServletException;
06. import javax.servlet.annotation.WebServlet;
07. import javax.servlet.http.HttpServlet;
08. import javax.servlet.http.HttpServletRequest;
09. import javax.servlet.http.HttpServletResponse;
10.
11. @WebServlet("/check-login.php")
12. public class CheckLogin extends HttpServlet {
13.     private static final long serialVersionUID = -9220125965466401589L;
14.
15.     protected void doGet(HttpServletRequest request, HttpServletResponse re
16.         String  login      = request.getParameter("login");
17.         boolean available  = ! "cgarcia".equalsIgnoreCase(login);
18.
19.         response.setStatus(HttpServletResponse.SC_OK);
20.         response.setContentType("application/json");
21.         response.getWriter().print(available);
22.     }
23. }
24.
```

Otros métodos de especificar las reglas de validación

JQuery Validator plugin permite incluir las reglas de validación directamente en el HTML, mediante propiedades personalizadas y clases css, por ejemplo:

```
view plain  print  ?
01. <input type="text" id="ca" name="ca" class="required email"/>
02. <input type="text" id="cb" name="cb" class="required" minlength="2" />
```

Conclusiones

Gracias a JQuery Validator, podemos validar nuestros formularios web de una forma cómoda, fexible, rápida y no intrusiva.

Un saludo.
Carlos García.

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

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

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

COMENTARIOS