El editor de DataTables (VIII). Los campos de tipo checkbox.

En el artículo anterior vimos un ejemplo de un dataset tal que, al editarlo, empleábamos campos de tipo radio button y también un campo de casilla checkbox. El uso de checkbox en el editor de DataTables no quedó demasiado claro entonces, y me comprometí a preparar un ejemplo que explicase realmente cómo funciona. En este artículo hemos vuelto a contar con una base de datos de desarrolladores y especialidades (más una tabla intermedia para relacionarlas). El objetivo de este artículo es comprender cómo gestiona el editor los campos checkbox.
Además, a partir de este artículo, introduciremos una novedad: no publicaremos en el texto todos los códigos completos, ya que, cada vez, son más largos, y recargan el artículo. Sólo incluiremos aquellos fragmentos de código que impliquen novedades, o en los que debamos centrarnos por cuestiones didácticas. Por supuesto, los códigos completos de cada artículo estarán, como siempre, disponibles para que te los descargues.

EL ESCENARIO
El escenario ya nos es familiar de otro artículo anterior. Contamos, como he mencionado ma´s arriba, con una tabla de desarrolladores, una tabla de las posibles especialidades que puede tene cada desarrollador, y una tabla intermedia para relacionar ambas. La diferencia está en que, en esta ocasión, cuando abramos el formulario para editar la ficha de un desarrollador (o para crear uno nuevo), la lista de especialidades irá en forma de campos checkbox, a razón de una casilla por especialidad. Las casillas las marcaremos o desmarcaremos en base a las especialidades que domina el desarrollador con el que estamos trabajando. Si se trata de una edición, al abrir el formulario veremos ya marcadas las casillas de las especialidades que el desarrollador ya tiene asignadas.
EL SCRIPT PRIMARIO
En el script primario hay algunos cambios que debemos tener en cuenta. En primer lugar, y puesto que vamos a reconocer las especialidades por el id que tengan estas en la correspondiente tabla MySQL, debemos crear, en la tabla HTML, una columna para alojar estos ID's. Como los ID's no le dicen nada a la persona que visualiza la tabla, esta columna será luego declarada como no visible, pero debemos contar con ella para tener los datos en nuestro objeto datatables. Así pues, la cabecera de la tabla HTML queda declarada del siguiente modo:

```
<thead>
 <tr>
  <th> </th>
  <th>NOMBRE</th>
  <th>APELLIDOS</th>
  <th>F. INC.</th>
  <th id="cabeceraDeSalario">SALARIO</th>
  <th> </th> <!-- Para los ids de especialidades. No será visible. -->
 </tr>
</thead>
```

Cómo ves, la última definición de la cabecera está sin contenido. Nos da igual, porque, cómo hemos dicho, no será visible. Cuando creamos el objeto DataTables, y definimos la columna que contendrá estos ID's lo hacemos del siguiente modo:

```
{"data": 'id_especialidades[]', visible: false}
```

Fíjate que, aparte del hecho de ser invisible, hay algo más que nos llama la atención. Tratamos los ID's de las especialidades de cada desarrollador como una matriz. Observa los dos corchetes que hay a continuación del nombre del dato id_especialidades. Esto es así porque, cuando nos metamos con este campo en el editor, el correspondiente objeto editor debe de verlo como una matriz, ya que los campos de tipo checkbox deben ser, por la propia naturaleza del editor, procesados de este modo.

Y, ya que a eso vamos, observa cómo creamos el campo checkbox en el objeto editor:

```
{
 label: 'Especialidades:',
 name: 'id_especialidades[]',
 type: 'checkbox',
 fieldInfo: 'Selecciona las especialidades',
 options: [
  {label: "HTML 5", value: "1"},
  {label: "JavaScript 6", value: "2"},
  {label: "CSS 3", value: "3"},
  {label: "SCSS", value: "4"},
  {label: "PHP", value: "5"},
  {label: "MySQL", value: "6"},
  {label: "jQuery", value: "7"},
  {label: "jQueryUI", value: "8"},
  {label: "Bootstrap", value: "9"},
  {label: "jQuery Mobile", value: "10"},
  {label: "Android", value: "11"}
 ]
},
```

Observa la forma en que le damos el nombre al campo, indicando, cómo hemos mencionado, que este dato se procesa como una matriz, mediante el uso de los corchetes. En el atributo options ponemos las especialidades que pueda haber. Cada una de ellas tiene una etiqueta (label) que aparecerá en el formulario junto a la casilla correspondiente. También tiene un valor (value), que será el que tenga la casilla correspondiente, y que el editor asignará al correspondiente atrributo al crear el HTML del formulario de edición. Este atributo debe coincidir con el id que la especialidad tiene en la tabla MySQL.

Observa que en el listado de las opciones, estas no aparecen en orden alfabético. Sin embargo, en el formulario de edición es conveniente que así sea, para facilitarle la lectura al usuario. No hay problema. Una vez más, la detección del evento preOpen nos ayuda releyendo la tabla de especialidades, esta vez en orden alfabético. El evento preOpen tiene asignado el siguiente código:

```
objetoEditor.on('preOpen', function(e, mode, action){
 if (action == "remove") return;
 $.ajax({
  url:"leer_especialidades_editor_08.php",
  async:false,
  dataType: "JSON",
  complete:function(datosRecibidos) {
   listaDeEspecialidades = datosRecibidos.responseText;
  }
 });
 var matrizDeEspecialidades = JSON.parse(listaDeEspecialidades);
 objetoEditor.field('id_especialidades[]').update(matrizDeEspecialidades);
});
```

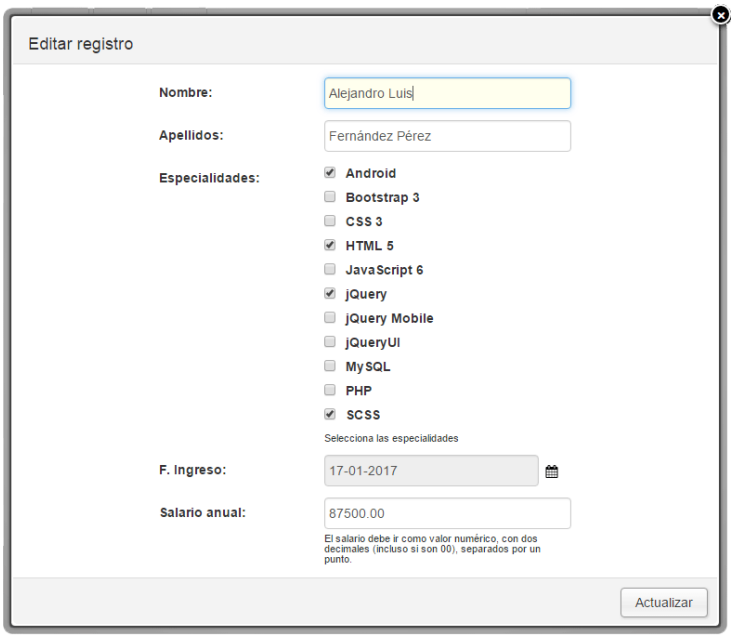
El script que lee las especialidades (leer_especialidades_editor_08.php) no puede ser más simple:

```
<?php
 // Establecemos la codificacion para las llamadas y respuestas HTTP
 mb_internal_encoding ('UTF-8');

 /* CREAMOS LA CONEXION A LA BASE DE DATOS, O BIEN LA IMPORTAMOS
 DESDE UN ARCHIVO EXTERNO DE CONFIGURACION. */
 include ('conexion_bd_editor.php');
```

```
/* Se obtiene la lista de las especialidades a partir de la correspondiente tabla. */
$consulta = "SELECT ";
$consulta .= "especialidad AS label, ";
$consulta .= "id AS value ";
$consulta .= "FROM especialidades ";
$consulta .= "ORDER BY especialidad;";
$hacerConsulta = $conexion->query($consulta);
$matrizDeEspecialidades = $hacerConsulta->fetchAll(PDO::FETCH_ASSOC);
$hacerConsulta->closeCursor();
$listaDeEspecialidades = json_encode ($matrizDeEspecialidades, JSON_HEX_QUOT);
echo $listaDeEspecialidades;
?>
```

Cuando tenemos cargada la tabla de datos en una ventana del navegador, seleccionamos una fila y pulsamos el botón de edición. El formulario que se abre tiene el siguiente aspecto:



Cómo puedes ver, se ha colocado la lista de checkboxes con las etiquetas que le hemos asignado, en el orden alfabético que se han recuperado gracias a la lectura que se hace cómo respuesta al evento preOpen. Aparecen marcadas las casillas que corresponden a las especialidades del desarrollador que estamos editando. El editor reconoce cuales son las que debe seleccionar gracias al contenido de la columna que hicimos invisible, lo que nos muestra la importancia que tiene recuperar este dato para poder editarlo.

ENVIANDO EL FORMULARIO

Veamos qué es lo que enviamos al script de edición (crud_editor_08.php, en este ejemplo), cuando pulsamos el botón de actualizar. Suponte que, en el ejemplo que estamos viendo, seleccionamos, además de las especialidades que ya están seleccionadas, MySQL (por ejemplo). Los ID's de las especialidades atribuidas al desarrollador de nuestra prueba son: 1 (HTML 5), 4 (SCSS), 6 (MySQL), 7 (jQuery) y 11 (Android). El script de edición recibirá una matriz como la siguiente:

```
array(2) {
 ["action"]=>string(4) "edit"
 ["data"]=>array(1) {
  [13]=>array(6) {
   ["nombre"]=>string(14) "Alejandro Luis"
   ["apellidos"]=>string(17) "Fernández Pérez"
   ["id_especialidades"]=>array(5) {
    [0]=>string(2) "11"
```

```
    [1]=>string(1) "1"
    [2]=>string(1) "7"
    [3]=>string(1) "6"
    [4]=>string(1) "4"
   }
   ["id_especialidades-many-count"]=>string(1) "5"
   ["fecha_incorporacion"]=>string(10) "17-01-2017"
   ["salario_bruto_anual"]=>string(8) "87500.00"
  }
 }
}
```

Fíjate en que, dentro de data, las especialidades constituyen, a su vez, una matriz que contiene los elementos que corresponden a las casillas marcadas. Esta matriz se almacena bajo una clave que corresponde con el nombre del campo (id_especialidades). Además, aparece un nuevo valor, con el nombre del campo seguido de -many-count (en nuestro caso, id_especialidades-many-count), que contiene el número total de casillas checkbox seleccionadas en ese campo.
Ahora supongamos que, a este desarrollador, le desmarcamos todas las especialidades. La matriz de datos del formulario será enviada del siguiente modo:

```
array(2) {
  ["action"]=>string(4) "edit"
  ["data"]=>array(1) {
    [13]=>array(5) {
      ["nombre"]=>string(14) "Alejandro Luis"
      ["apellidos"]=>string(17) "Fernández Pérez"
      ["id_especialidades-many-count"]=>string(1) "0"
      ["fecha_incorporacion"]=>string(10) "17-01-2017"
      ["salario_bruto_anual"]=>string(8) "87500.00"
    }
  }
}
```

Fíjate que aquí no aparece la matriz id_especialidades. No es que aparezca vacía. Es que, simplemente, no aparece. Además, el campo id_especialidades-many-count tiene el valor 0.

EDICIONES MÚLTIPLES

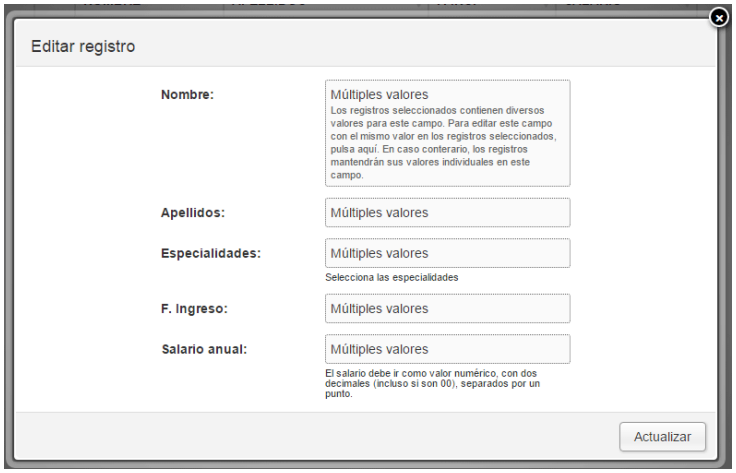
Vamos a suoner que en nuestro datatables permitimos la edición de múltiples registros simultáneamente. Para ello vamos a editar el script primario (articulo_editor_08.php). Buscamos la siguiente línea:

```
style: 'single',
```

En este script está en la línea 327. Sustituyela por:

```
style: 'multiple',
```

Ahora recarga la página. Ya puedes seleccionar varios registros para edición múltiple. En mi ejemplo he seleccionado los tres primeros que aparecen. Al pulsar el botón de edición, el formulario tiene el siguiente aspecto:



Cuando pulsamos el botón de enviar, la matriz de datos que se envía se parece a la siguiente:

```
array(2) {
 ["action"]=>string(4) "edit"
 ["data"]=>array(3) {
  [13]=>array(6) {
   ["nombre"]=>string(14) "Alejandro Luis"
   ["apellidos"]=>string(17) "Fernández Pérez"
   ["id_especialidades"]=>array(4) {
    [0]=>string(1) "1"
    [1]=>string(1) "4"
    [2]=>string(1) "7"
    [3]=>string(2) "11"
   }
   ["id_especialidades-many-count"]=>string(1) "4"
   ["fecha_incorporacion"]=>string(10) "17-01-2017"
   ["salario_bruto_anual"]=>string(8) "87500.00"
  }
  [16]=>array(6) {
   ["nombre"]=>string(7) "Alfonso"
   ["apellidos"]=>string(12) "López Banto"
   ["id_especialidades"]=>array(6) {
    [0]=>string(1) "1"
    [1]=>string(1) "2"
    [2]=>string(1) "3"
    [3]=>string(1) "7"
    [4]=>string(1) "8"
    [5]=>string(1) "9"
   }
   ["id_especialidades-many-count"]=>string(1) "6"
   ["fecha_incorporacion"]=>string(10) "18-05-2015"
   ["salario_bruto_anual"]=>string(9) "165200.00"
  }
  [18]=>array(6) {
   ["nombre"]=>string(6) "Andrea"
   ["apellidos"]=>string(13) "Adrián Mares"
   ["id_especialidades"]=>array(3) {
    [0]=>string(1) "1"
    [1]=>string(1) "4"
    [2]=>string(2) "10"
   }
   ["id_especialidades-many-count"]=>string(1) "3"
   ["fecha_incorporacion"]=>string(10) "13-10-2014"
   ["salario_bruto_anual"]=>string(9) "165000.00"
  }
 }
}
```

Observa que dentro de data tenemos tres matrices, una por cada desarrollador que estábamos editando. Cada una de ellas incluye una matriz id_especialidades, con las especialidades atribuidas a ese desarrollador. Si alguno de ellos no tuviera especialidades, la correspondiente matriz id_especialidades no existiría, cómo ya sabemos. Además, cada desarrollador cuenta con el campo

id_especialidades-many-count, que lleva la cuenta de las especialidades que tiene ese desarrollador.
Todo esto debemos tenerlo en cuenta al diseñar el script que efectúa la edición, el borrado o la creación de un registro. En el script crud_editor_08.php se han incluido los comentarios necesarios para que entiendas cómo se procesa pero, ten en cuenta que, una vez que entendemos el formato en que se envían los datos, realmente la lógica del procesado de estos es PHP elemental, si te pones a mirarlo.
Todos los scripts de este ejercicio, así como la base de datos, te los puedes descargar en este enlace.