Dynamic Tables

Manual de usuario v1.2

Contenido

[Historial de cambios: 3](#_Toc388001701)

[Descripción 4](#_Toc388001702)

[Modo de uso 5](#_Toc388001703)

[Ejemplo de uso 7](#_Toc388001704)

# Historial de cambios:

|  |  |
| --- | --- |
| 1 | No se debe importar el js, el css y las imágenes para poder usar la biblioteca, todo está ubicado en un directorio dentro del servidor. |
| 2 | La función loadTable ya no tiene parámetros. |
| 3 | No es necesario importar el archivo css de las tablas si se utilizarán los estilos por defecto. |
| 4 | Las imágenes de las acciones de edición y borrado no deben ser importadas, se encuentran en el de la biblioteca en el servidor. |
| 5 | Se agregaron los atributos *imgWidth* e *imgHeight* para el manejo del ancho y largo de las imágenes para las acciones de edición y borrado |
| 6 | Se agregó el parámetro *imgsPath* para especificar la ruta de las imágenes para las acciones de edición y borrado. |

# Descripción

Esta clase se encarga de modelar tablas dinámicas con JavaScript y css. Para crear una nueva tabla se debe contar con n campos domino donde se almacenaran los datos ingresados separados por ";", n campos HTML para el modo lectura donde se cargaran los datos de los campos domino antes mencionados, n campos HTML para el ingreso de los datos de la tabla y un div contenedor cuyo identificador será requerido para la creación de la tabla.

Los atributos obligatorios son los siguientes:

* ***cantColumns*** : cantidad de columnas que tendrá la tabla (entero > 0)
* ***header*** : vector de largo *cantColumns* con el nombre de cada columna
* ***dominoFields*** : vector de largo *cantColumns* con el id del campo de cada columna (campos multivaluados, separador ";")
* ***htmlFields*** : vector de largo *cantColumns* con el id de los campos HTML de ingreso de datos para la tabla (tipos admitidos: *text*, *textArea*, *checkbox*, *combobox*, *radio*)
* ***tableId***  : identificador HTML de la tabla a construir
* ***columnKey*** : indica que columna es clave de la tabla para la validación de datos repetidos (si *columnKey* <0 no hay validación)
* ***validator*** : función de validación (*null* en caso de no contar con validador, en caso contrario función definida por el usuario: retorna true si cumple las validaciones, false en caso contrario; )
* ***tableVarName*** : nombre de la variable *DynamicTable* empleada para la tabla
* ***editEnabled*** : bandera booleana para habilitar la edición de una fila
* ***deleteEnabled*** : bandera booleana para habilitar el borrado de una fila
* ***dynTableDiv*** : id del div contenedor de la tabla dinámica a construir
* ***tableWidth*** : ancho de la tabla (% o pixeles)
* ***editable*** : establece si el modo edición está habilitado o no (booleano)

Existen también atributos opcionales

* ***buttonsNames*** : vector de largo 2 con el nombre del botón de creación y el de actualización de una fila de la tabla (por defecto "Agregar", "Modificar")
* ***colWidths*** : vector de largo cantColumns con el ancho de cada columna de la tabla (% o pixeles, por defecto todas las columnas tienen el mismo ancho)
* ***buttonsDiv*** : identificador del div donde estarán contenidos los botones de la tabla. (Por defecto es el mismo div donde estará la tabla)
* ***doubleClick*** : acción definida para el evento doble click sobre las filas de cada tabla. (Por defecto la acción simplemente copia los valores de la fila seleccionada en los campos de ingreso).
* ***imgSize*** : elemento compuesto por el alto y ancho de las imágenes de edición y borrado. (Por defecto *width*=20, *height*=20) **(5)**
* ***imgsPath*** : especifica la ubicación de las imágenes de las acciones de edición y borrado para cada fila.**(6)**

# Modo de uso

Para hacer uso de esta biblioteca se debe incluir el subformulario llamado “TablasDinamicas”. Este subformulario importa el css, la lógica y la ubicación de la biblitec. Se accede a la misma mediante el parámetro de configuración “sPathDynamicTable” extraido de la configuración de iGDoc. Este parámetro contiene el path de la carpeta donde están los archivos de la biblioteca, en el caso de desarrollo2, el parámetro es “/TablasDinamicas/1.1”. **(1)**

Para crear una nueva tabla se debe instanciar la función *'DynamicTable'* con todos los parámetros obligatorios mencionados anteriormente.

En caso de querer modificar alguno de los parámetros opciones se debe invocar la función correspondiente para el atributo a modificar (*updateButtonsNames*, *updateColWidths, updateDoubleClickAction* *o setButtonsDiv*).

* *updateButtonsNames :* actualiza el nombre de los botones de una tabla (vector de largo 2)
* *updateColWidths :* actualiza el ancho de cada columna (vector de largo *cantColumns*)
* *setButtonsDiv :* define el lugar donde se ubicaran los botones de la tabla
* *updateDoubleClickAction :* actualiza la función definida para el evento *doble click* sobre una fila determinada
* *updateImgSize :* actualiza el ancho y largo de las imágenes para las acciones de edición y borrado. **(5)**
* *updateImgsPath :* actualiza la ubicación de las imágenes para las acciones de edición y borrado. **(6)**

Una vez cargados todos los parámetros se debe invocar a la función *loadTable* de la tabla creada para que cargue los datos existentes y habilite las acciones configuradas. **(2)**

Las tablas cuentan con estilos css. Estas clases están definidas en el archivo *dynTableCss.css* el cual ya está importado en el subformulario de la biblioteca. **(3)**

En caso de querer personalizar el estilo de los botones y tablas se debe realizar una copia de dicho archivo, modificar las clases existentes siempre respetando los nombres de las mismas e importarlo en el HTML Head Content del formulario.

En caso de estar en modo lectura, o trabajando con formularios y sea necesario discriminar por secciones, por ejemplo, se debe especificar la visibilidad de la tabla de los datos de entrada (style="display: ..."). Es recomendado usar un *<Computed text>* donde la fórmula del mismo especifique en qué casos debe ser *'none'* el atributo y en cuáles no. Además de este último paso se debe setear en la tabla que no está habilitado el modo de edición de la misma al momento de inicializarla (parámetro *‘editable’).* De esta forma si el parámetro es *true* se estará indicando que el modo de edición está habilitado.

Los campos HTML que se corresponden con los campos domino, deben ser identificados de la misma forma que los campos domino y además deben estar ocultos en modo edición por medio de una formular de ocultamiento sobre el texto de los mismos. Esto se debe a que, en modo lectura, los campos domino dejaran de ser campos HTML y únicamente se encontraran los campos HTML existentes y en modo edición los únicos campos necesarios serán los domino. El atributo *value* de cada campo HTML debe ser un ‘*Computed text’* con la siguiente formula*: @Implode(idCampoDomino;";" para respetar el mismo formato de los datos en los campos domino.*

Las acciones integradas en cada tabla son 3.

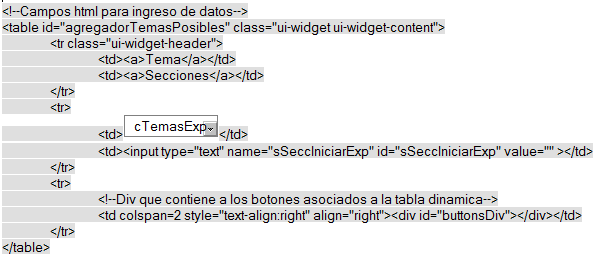
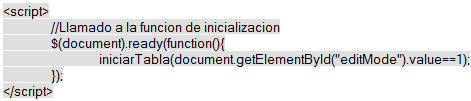
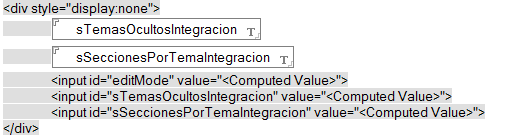
* **Editar** : mediante el botón de edición de cada fila se cargan los datos en los campos de entrada y se cambia la acción del botón de agregado. Al modificar los campos se vuelve a realizar las validaciones correspondientes y en caso de verificarlas se modifica la fila en la tabla y los datos en los campos domino correspondientes.
* **Remover** : mediante el botón de borrado de cada fila se remueve la misma de la tabla y también todos de los campos domino correspondiente.
* **Copiar** : mediante un “doble click” en una fila, se cargan los datos de la misma para ser usados en una nueva fila (recordar que la acción de doble click puede llegar a ser definida por el desarrollador creando una función para dicho evento).

## Ejemplo de uso

A continuación se presenta un ejemplo de uso de esta clase.

En este ejemplo se utilizan una tabla dinámica contenida en el div *dynTableDiv1*. Cabe destacar que en caso de tener que utilizar radio buttons como campos de entrada HTML, es necesario que se incluya el atributo *value* con el valor de la etiqueta del botón.

Los campos HTML son *sTemasOcultosIntegracion y sSeccionesPorTemaIntegracion* los cuales coinciden en su identificador con los campos domino que se encuentran definidos encima de ellos. La fórmula en el Computed value de cada uno de estos campos es el siguiente: @Implode(sTemasOcultosIntegracion;";") y @Implode(*sSeccionesPorTemaIntegracion*;";") respectivamente.

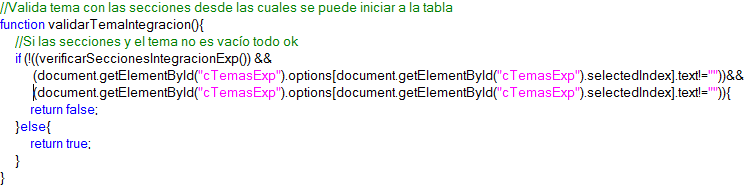


La fórmula del *Computed Text* es la siguiente: *@If(@IsDocBeingEdited;"";”none)*

Luego en le js header del form se deben crear tantas variables globales como tablas se quieran modelar. Se crea una función para ser invocada en el evento *onLoad* del formulario. En esta función se inicializan los parámetros para cada tabla y por medio de la sentencia *tableDyn1=new DynamicTable(...)* se inicializa una de las tablas donde, en este caso, el parámetro *tableVarName* es "tableDyn1" el nombre de la variable global para referenciar a la tabla. Luego se actualizan los parámetros opcionales y por medio de la función loadTable(…) se cargan los datos existentes en los campos domino y los botones y estructuras a utilizar.

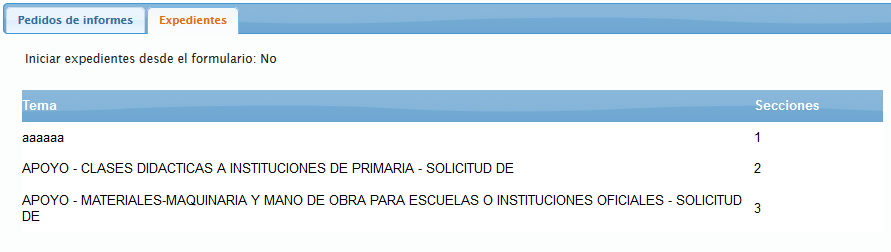


También se define una función *validarTemaIntegracion ()* que es la encargada de validar los datos previo al ingreso en la tabla. En caso de cumplir con los criterios establecidos por el usuario retorna true y en caso contrario false. Si el usuario no cuenta con una función de validación se debe pasar *null* como parámetro.



Este ejemplo se encuentra en la siguiente url: en el subForm *IntegraciónExpedientes* y es usado por el formulario *Definición Formularios*  en la base *Desarrollo\Administ.nsf*. A continuación se pueden ver algunas imágenes demostrando como se ve este ejemplo.

Modo edición deshabilidato:



Modo edición habilitado:

