

# Guide du développeur

*Utilisation de DataTable*



Edouard COMTET  
Février 2014

# Guide du développeur

## *Utilisation de DataTable*

Introduction	3
Installation	4
Création d'un objet Table	5
Création de la page HTML	5
Mise en place du PHP	5
Personnalisation des options	6
Ajout des colonnes et des filtres	7
Ajout d'un filtre	7
Ajout d'une colonne	7
Affinement des autres options	9
Fonctions CRUD	9
Pagination & tri	9
Langue	10
Rendu de l'interface	11
Modifier	11
Ajouter	11
Supprimer	12
Introspection sur le type des champs	12
Affichage des erreurs	12
Outils de détection des erreurs	13

# 1. Introduction

Lors de la création d'une partie d'administration, la plupart des développeurs doivent régulièrement mettre en place des tables pour la gestion (table des utilisateurs, des news, ...).

Le but de ce framework est de vous permettre, en quelques lignes de code, de générer une table HTML/CSS/Javascript relié à votre base de données. La modification de la table sera, de manière automatique, répercuté dans la base.

Ce document est destiné aux développeurs qui devront l'utiliser. Il vous expliquera quelle fonction utilisé et comment le paramétrer.

Dans la suite de ce document, on considèrera que le framework est installé dans un dossier du nom *DataTable* et que le fichier utilisant ce dernier est à la racine et se nomme *index.php*.

Ce document est rattaché à un deuxième qui permet de comprendre l'architecture du framework et comment le modifier.

Ce framework se base sur DataTable, disponible à l'adresse <http://datatables.net/>. Le code source de cette amélioration est disponible à <https://github.com/doudou34/DataTable>.

## 2. Installation

L'installation du framework se décompose en 3 étapes simplistes:

- Copier l'intégralité des fichiers dans le répertoire d'installation
- Configurer le dossier *DataTable/temp/* pour que le serveur web soit accessible en lecture et en écriture (celui qui exécutera les pages PHP).
- Initialiser les variables de connexion disponible au début du fichier *DataTable/php/Table.php*.

```
// Default parameters
protected static $BDD_URL = "127.0.0.1";
protected static $BDD_LOGIN = "root";
protected static $BDD_PASSWORD = "";
protected static $BDD_DATABASE = "pfe";
protected static $BDD_DRIVER = "mysql";
protected static $LANGUAGE = "en";
protected static $ENCODING = "utf8";
```

Pour le paramètre de langues, se référer à la partie “Langue”, pour ajouter un driver, se référer au deuxième document.

Ces paramètres seront utilisés par défaut pour la connexion à la base de données et le rendu. Il est important de ne pas mettre de constante dans ces variables. En effet, pour pouvoir assurer la transition entre javascript et php, l'objet *Table* est sérialisé. L'utilisation de constante, avec l'instruction “define” par exemple, n'assure pas la désérialisation.

Ce type d'erreurs peut-être facilement détectées, référé vous à la partie “Outils de détection des erreurs”.

### 3. Création d'un objet Table

#### 1. Création de la page HTML

Pour créer une nouvelle page avec un Table dedans, il faut d'abords créer une page HTML. Voici un exemple:

```
<html>
  <head>
    <title>Test DataTable</title>

    <link href="./DataTable/css/bootstrap.css" rel="stylesheet">
    <link href="./DataTable/css/bootstrap-theme.css" rel="stylesheet">
    <link href="./DataTable/css/dataTables.css" rel="stylesheet">
  </head>
  <body>
    <div class="container">
      <div class="row">
        <h1>Test du PFE avec affichage en bootstrap</h1>
      </div>
      <div class="row">
        <?php
          // CODE PHP ICI
        ?>
      </div>
    </div>

    <script type="text/javascript" src="./DataTable/js/jquery.js"></script>
    <script type="text/javascript" src="./DataTable/js/bootstrap.js"></script>
    <script type="text/javascript" src="./DataTable/js/dataTables.js"></script>
    <script type="text/javascript" src="./DataTable/js/table.js"></script>
  </body>
</html>
```

Il est nécessaire d'inclure la librairie Bootstrap (autant au niveau Javascript que CSS), la librairie DataTable de base ainsi que le fichier CSS du framework et son équivalent Javascript. L'utilisateur devra également avoir activé Javascript.

#### 2. Mise en place du PHP

La structure de base étant en place, la création d'un objet Table sera un jeu d'enfant. Dans la partie PHP, vous pouvez y inscrire ceci:

```
require_once('./DataTable/php/include.php');

$table = new DataTable\php\Table();
$table->init("user");

$table->show("./DataTable/");
```

Cette partie de code permet de créer un objet Table, de le connecter à la table *user* et de l'afficher. La fonction *show* prends un seul paramètre: le chemin où est installé le framework depuis le fichier d'utilisation. Il doit se terminer par un */*.

### 3. Personnalisation des options

Il est possible de modifier les paramètres de connexion et de langue grâce à la fonction *init*.

S'il n'y a pas de paramètre passé ou qu'ils sont vide, les valeurs par défaut vu plus haut seront utilisées.

Nous étudierons plus tard comment modifier le style de pagination, la gestion du tri, de la modification, de la suppression, de l'ajout et des langues.

```
public function init(  
    $name,  
    $url = "",  
    $login = "",  
    $password = "",  
    $database = "",  
    $driver = "",  
    $language = "",  
    $encoding = ""  
);
```

Dans le deuxième document, il est également expliqué comment ajouter un driver et la personnalisation du CSS.

La partie suivante permet d'ajouter des colonnes et des filtres pour afficher une table selon vos besoins.

## 4. Ajout des colonnes et des filtres

Pour ajouter des colonnes et des filtres, il faut utiliser des fonctions pour chaque type de colonnes ou de filtres.

### 1. Ajout d'un filtre

Un filtre permet de n'afficher qu'une certaine partie des données d'une table, par exemple si on ne souhaite afficher que les utilisateurs qui sont connecté on fera:

```
$table->addFilter("connected", "= TRUE");
```

Cette fonction prends en paramètre la colonne sur lequel on fera le tri et la condition à prendre en compte.

### 2. Ajout d'une colonne

Il est possible d'ajouter différents types de colonnes à un objet *Table* qui seront, suivant les paramètres, affichés ou non, modifiables ou non, ... Voici la liste intégrale des colonnes disponibles:

- *Col*: Une colonne classique. La fonction prends en paramètres un libellé affiché à l'utilisateur et le nom de la colonne dans la table.

```
public function addCol($label, $name)
```

- *ColIndex*: Cette colonne sera utilisée comme index lors de la modification mais il est impossible de la modifier. Il faudra la supprimer et la rajouter. La fonction prends en paramètre un libellé pour la colonne, le nom de la colonne dans la table, un premier booléen pour gérer l'affichage et un deuxième pour préciser s'il faut proposer la colonne lors de l'ajout.

```
public function addColIndex($label, $name, $visible, $creatable)
```

- *ColHidden*: Cette colonne n'est jamais affiché à l'utilisateur mais lors de l'ajout, la valeur par défaut de la colonne sera ajouté. La fonction prends en paramètre un libellé, non utilisé, le nom de la colonne dans la table et la valeur par défaut (non vide).

```
public function addColHidden($label, $name, $default)
```

- *ColLinked*: Cette colonne est liée à une autre table (notion de clef étrangère). La liaison sera faite automatiquement avec la seconde table et affichera la valeur de la seconde table. La fonction prends en paramètre un libellé affiché, le nom de la colonne dans la table, le nom de la table où faire liaison, l'index de la seconde table (une seule colonne) et la colonne qui sera affiché à l'utilisateur (unique). Sur ce type de colonne il est également possible d'ajouter un filtre avec la fonction `getCol` et `addFilter`.

```
public function addColLinked($label, $name, $otherTable, $otherIndex, $otherName)
$table->getCol("user_group")->addFilter("label", "= 'Admin'");
```

- *ColIndexLinked*: Une colonne qui servira d'index et qui est lié à une autre table. Elle prends en paramètre les mêmes paramètres que la fonction `ColLinked`. Il n'est pas nécessaire de gérer l'affichage et l'ajout car l'utilisation de `ColIndex` remplit ces cas d'utilisations.

```
public function addColIndexLinked($label, $name, $otherTable, $otherIndex,
$otherName)
```



## 3. Affinement des autres options

### 1. Fonctions CRUD

Lors de la création d'une fonction, il existe principalement 4 sous fonctions: la création (Create), la lecture (Read), la modification (Update) et la suppression (Delete). Ce framework permet de distinguer ces 4 possibilités.

```
// Operation available
public static $CREATE = 0x01;
public static $READ = 0x02;
public static $UPDATE = 0x04;
public static $DELETE = 0x08;

public function canCreate($create);
public function canRead($read);
public function canUpdate($update);
public function canDelete($delete);

public function can($flags)
```

La dernière fonction utilise les 4 premiers drapeaux. On pourra par exemple donner le droit de création et de lecture avec (ce qui enlèvera le droit de modification et de suppression):

```
$table->can(DataTable\php\Table::$CREATE | DataTable\php\Table::$READ);
```

### 2. Pagination & tri

De la même manière, on peut activer la pagination ou la désactiver avec les drapeaux et la fonction suivante:

```
// Style of pagination
public static $PAGINATION_FULL_NUMBERS = "full_numbers";
public static $PAGINATION_NO = "";

public function setPaginateStyle($paginate)
```

Et la fonction pour activer ou désactiver le tri qui prends en paramètre un booléen:

```
public function canSort($sort);
```

### 3. Langue

Il est également possible de personnaliser le framework en modifiant la langue avec celles disponibles et la fonction `init` vu précédemment.

```
// Language available
public static $FRANCAIS = "fr";
public static $ENGLISH = "en";
```

Il est également possible de rajouter une langue en créant un nouveau fichier dans `DataTable/translations/`. Le nom du fichier se compose `nomDeLaLangue.json` et le fichier se structure de cette manière:

```
{
    "sProcessing": "Traitement...",
    "sSearch": "Rechercher",
    "sLengthMenu": "Afficher _MENU_ éléments",
    "sInfoThousands": " ",
    "sInfo": "Affichage des éléments _START_ à _END_ sur _TOTAL_ éléments",
    "sInfoEmpty": "Aucun éléments",
    "sInfoFiltered": "(filtré de _MAX_ éléments au total)",
    "sInfoPostFix": "",
    "sLoadingRecords": "Chargement...",
    "sZeroRecords": "Aucun élément à afficher",
    "sEmptyTable": "Aucune donnée disponible dans le tableau",
    "sDragDrop": "Glissez-Déposez une ligne pour la supprimer",
    "oPaginate": {
        "sFirst": "Premier",
        "sPrevious": "Précédent",
        "sNext": "Suivant",
        "sLast": "Dernier"
    },
    "oAria": {
        "sSortAscending": ": Activer pour trier la colonne par ordre croissant",
        "sSortDescending": ": Activer pour trier la colonne par ordre décroissant"
    },
    "eFunctionUnkown" : "Fonction inconnue",
    "eMissingPatameters": "Paramètres non trouvés",
    "eUnserialize": "Impossible de désérialiser",
    "eUnkonwn": "Erreur inconnu",
    "eAction": "Impossible d'effectuer cette action",
    "eQuery": "Erreur dans la requête",
    "eFindCol": "Impossible de trouver la colonne",
    "eFile": "Impossible de trouver le fichier",
    "eFail": "Erreur",
    "eNone": "Succès"
}
```

## 4. Rendu de l'interface

### Test de DataTable

Show 10 entries

+

Search:

N	Login	Mail	Date de naissance	est Grand	Peut se connecter	Groupe
1	eco	ecomtet@test.com	1991-06-13	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Admin
2	farcham	farcham@test.com	1991-06-24	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Redacteur
3	mdarthos	mdarthos@test.com	1991-06-24	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Admin
4	svangaev	svangaev@test.com	1987-11-28	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Redacteur
16	tata	tata@test.com Ceci est un test avec accent &é"(\$\$è!!	2014-01-17	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Utilisateur
17	tonton	titi@test.com	2014-02-20	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Utilisateur
18	titi	titi@test.com	2014-02-20	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Admin
19	tete	tete@test.com	2014-02-21	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Admin
20	tyty	tyty@test.com	2014-02-22	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Admin
21	zaza	zaza@test.com Ceci est un test avec accent &é"(\$\$è	2014-02-23	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Admin

Showing 1 to 10 of 13 entries

First

Previous

1

2

Next

Last

### 1. Modifier

Pour modifier une case, il suffit de double cliquer dessus. Un clique autre part sauvegarde la modification, une pression sur la touche échappe l'annule. Si il y a une erreur, elle sera affichée sur la case en cours de modification.

### 2. Ajouter

Une icône avec un “+” permet d'ajouter une nouvelle ligne. Une nouvelle fenêtre s'ouvre et vous permet de remplir les cases. En cliquant sur enregistrer, la nouvelle ligne s'ajoute ou une erreur est affichée sur l'icône “+”.

Ajouter un enregistrement

Login

Mail

Date de naissance

dd/mm/yyyy

☒ est Grand

☒ Peut se connecter

Groupe

Admin

Annuler

Enregistrer

### 3. Supprimer

Il existe deux manières de supprimer:

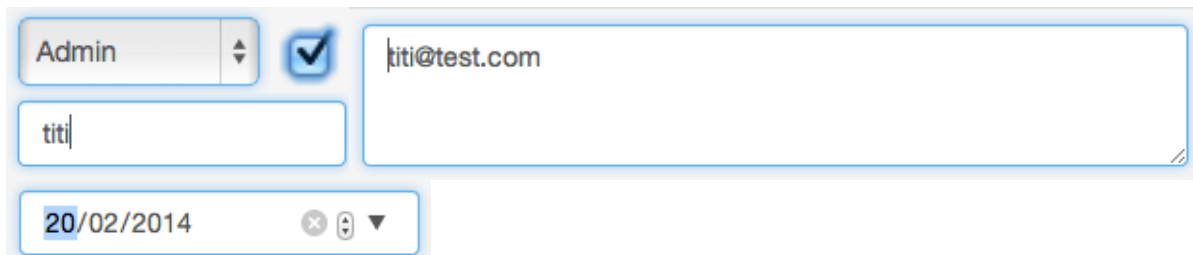
- En glissant déposant une ligne sur l'icône de poubelle.
- En sélectionnant des lignes et en cliquant sur la poubelle. Il est possible de sélectionner plusieurs lignes.

17	nom	email	date	statut	statut	utilisateur
18	titi	titi@test.com	2014-02-20			Admin
19	tete	tete@test.com	2014-02-21			Admin
20	tyty	tyty@test.com	2014-02-22			Admin
21	zaza	zaza@test.com Ceci est un test avec accent &é"(\$\$è	2014-02-23			Admin

En cas d'erreurs, elle sera affiché sur l'icône de la poubelle.

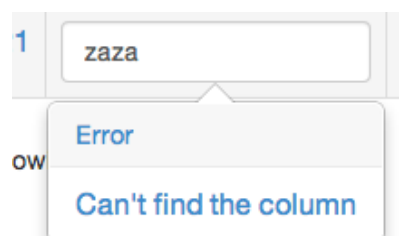
### 4. Introspection sur le type des champs

Ce framework détecte également automatiquement le type des champs de la base. Pour cela, une requête est exécutée et récupère le type de chaque colonne. L'affichage se transforme suivant le type (linked, booléen, varchar, text, date, ...).



### 5. Affichage des erreurs

Les erreurs s'affichent sous forme de bulle comme ceci:



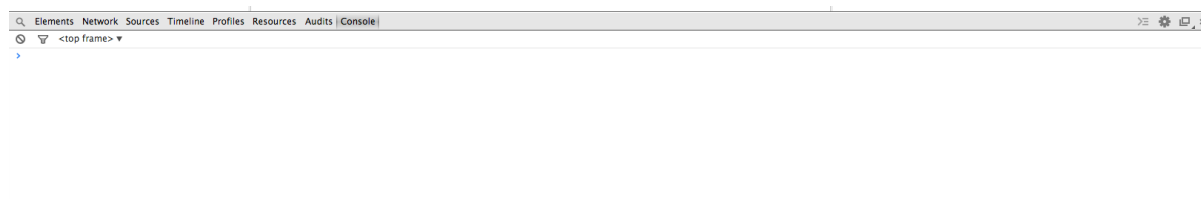
## 6. Outils de détection des erreurs

Ce projet, encore en phase de test, peut présenter des erreurs. Il est possible de les signaler ici: <https://github.com/doudou34/DataTable/issues>. Une correction sera apportée le plus rapidement possible.

Il peut aussi s'agir d'une erreur de votre part suite à un manque de documentation. Pour vous aider, voici deux méthodes.

Les erreurs classiques, en PHP. Pour que PHP affiche les erreurs il est nécessaire de mettre la variable `display_errors` à "On". Deux manières, directement en PHP, ici <http://php.net/manual/fr/errorfunc.configuration.php>, ou dans le fichier `php.ini`. Pour trouver ce fichier, la fonction `phpinfo()` vous aidera.

Les erreurs en javascript peuvent être détectées sous Chrome avec la console. Pour cela faites un clique droit, inspecter l'éléments. Vous devriez voir apparaitre une fenêtre en bas:



L'onglet console vous permet de détecter les erreurs javascript. Un deuxième onglet, Réseau, vous permettra de détecter les erreurs AJAX en récupérant les fichiers requis par la page, notamment le fichier `ajax.php` qui sert de passerelle entre le PHP et le javascript dans le framework.

