

UniversalForm

V3.16.0

Table des matières

1. Classe UniversalForm	3
2. Prérequis système	3
3. Prérequis	4
4. Notre premier formulaire UniversalForm	5
Création	5
Initialisation des données du buffer	7
Construction des champs du formulaire	8
Afficher le formulaire	9
Mise en forme des champs	12
5. Gestion du formulaire.....	15
6. Types de champs.....	17
Type de champ « text »	19
Type de champ « radio »	26
Type de champ « checkbox »	32
Type de champ « switch »	37
Type de champ « select »	41
Type de champ « area »	45
Type de champ « bouton »	49
Type de champ « image »	52
Type de champ « search »	55
Type de champ « filtretext »	60
Type de champ « filtreselect »	66
Type de champ « séparateur »	71
Type de champ « comment »	73
Type de champ « div »	76
Type de champ « divfin »	76
Type de champ « hidden »	76
7. Test du formulaire.....	77
Tests unitaires (champ par champ)	77
Tests supplémentaires.....	81
Tests supplémentaires postérieurs	82
8. Construisons un beau formulaire !.....	83
Boutons radio	83
Checkbox	84
Select	86
9. Affiner la construction du formulaire	88
10. Bibliothèque de fonctions JavaScript	90
11. Modifier les données avant réception	92
12. Envoyer un message à l'application.....	93
13. Code complet	93
14. Getters et Setters	98
Getters	98
Setters	101
15. Récapitulatif des méthodes disponibles	104
16. Gestion complète d'un formulaire	107
17. Annexes	110

1. Classe UniversalForm

Le présent document est un tutoriel qui, par l'exemple, va expliquer pas à pas comment utiliser la classe **UniversalForm** pour créer des formulaires fiables et riches dans du code PHP.

Un formulaire est un ensemble de champs qui vont être soumis à l'utilisateur et par lequel il pourra saisir des données. Le formulaire peut être composé de plusieurs types de champs de saisie : texte, bouton radio, boîtes à cocher, listes déroulantes, boutons de validation, texte caché, etc.

Lorsqu'un utilisateur valide la saisie d'un formulaire, PHP génère un `POST`, c'est-à-dire qu'il envoie les données au système ; données que le code doit récupérer.

NB : Cette version 3.x.x est basée sur les styles CSS Bootstrap (créé par Twitter). Contrairement à la version 2.x.x, il n'est plus nécessaire d'inclure la feuille de styles particulière `universalForm.css`. La présente version est basée sur la version Bootstrap v4.3.1. **Important** : il n'est pas conseillé d'utiliser cette version d'UniversalForm conjointement avec une version inférieure de Bootstrap.

2. Prérequis système

Pour que fonctionne correctement la classe UniversalForm, l'extension `fileinfo` de PHP doit être activée. Pour ce faire, dans le fichier `PHP.INI` de votre serveur web, supprimez le commentaire devant la ligne suivante : `extension=php_fileinfo.dll`

3. Prérequis

Nous allons écrire une petite vidéothèque pas à pas afin de montrer comment fonctionne UniversalForm. Cette classe s'inscrivant totalement dans le squelette **UniversalWeb**, nous allons au préalable installer l'arborescence UniversalWeb comme indiqué dans le document idoine en ayant pris soin de paramétrer notre nouvelle application.

Pour rappel voici la structure du *backend* obtenue que nous allons utiliser pour écrire notre code :

```
backend                                     //emplacement de votre code
  bootstrap-4.3.1-dist                     //emplacement des fichiers du framework Bootstrap
    css                                   //emplacement des fichiers CSS de Bootstrap
    js                                    //emplacement des fichiers js (javascript) de Bootstrap
  css                                       //emplacement des fichiers css du site
  fontawesome-free-5.7.2-web              //emplacement de la librairie font-awesome
    css                                   //sous dossiers de la librairie font-awesome
    webfonts                              //sous dossiers de la librairie font-awesome
  images                                  //emplacement des images du site
    common                               //emplacement des images communes
    drapeaux                             //emplacement des drapeaux
    en                                    //emplacement des images anglaises
    fr                                    //emplacement des images françaises
  js                                       //emplacement du code javascript du site
  libs                                    //code secondaire (librairies)
    armoire                              //emplacement d'échange
    briques                              //emplacement des briques de construction UniversalWeb
    classes                              //emplacement des classes du site
    DB_backup                            //emplacement des sauvegardes de base de données
    langues                               //emplacement des fichiers de langues
    sql                                  //emplacement des requêtes SQL
```

NB : Il est aussi possible d'utiliser la classe en *standalone* sans la structure UniversalWeb.

NB : Pour ceux qui sont un peu plus pressés, cette petite application est déjà disponible et installée avec le squelette UniversalWeb. Pour y accéder tapez l'url **exemple_videotheque.php**.

4. Notre premier formulaire UniversalForm

Un formulaire **UniversalForm** est une instance (objet) d'une classe qui hérite de la classe **UniversalForm**.

Nous allons construire 2 fichiers PHP :

- **Form_videothèque.class.php** contiendra la classe dérivée qui s'occupera de la construction du formulaire et sera déposé dans le dossier CLASSES.
- **saisie_videothèque.php** s'occupera de la gestion du formulaire. C'est notre page web.

Notons qu'il est nécessaire de construire 2 fichiers pour chaque formulaire : un fichier de construction et un fichier de gestion. Si votre application possède 4 formulaires, il faudra écrire 8 fichiers de code.

Création

1 – Création de la nouvelle classe. Créer un fichier PHP intitulé **Form_videothèque.class.php**, y insérer le code suivant puis l'enregistrer sous le dossier **libs/classes** :

```
<?php

class Form_videothèque extends UniversalForm {
}
```

Notre classe dérivée est créée.

2 – Création de la page Web. Créer le fichier PHP **saisie_videothèque.php**, y insérer le code suivant puis l'enregistrer directement à la racine du backend ou se situent vos pages :

```
<?php

require_once('libs/classes/UniversalForm.class.php');
require_once('libs/classes/Form_videothèque.class.php');

$monFormulaire = new Form_videothèque(UniversalForm::AJOUTER, 1);
echo $monFormulaire->getOperation();
```

Important : Enregistrez vos fichier sous le jeu de caractère **UTF-8** afin que le jeu de caractères utilisé soit totalement compatible avec **UniversalWeb** et **UniversalForm**.

NB : Dans ce code est implémenté le tag d'ouverture de php **<?php**. Notez que celui-ci est tacite et que par soucis de simplification nous ne le ferons plus dans la suite de notre développement.

Noter le formalisme d'écriture correct : le nom d'une classe commence toujours par une majuscule. Noter également que le nom de la classe **Form_videothèque** et celui du fichier **Form_videothèque.class.php** doivent être identiques. Le nom du fichier devant être suffixé par « .class.php »

Si vous lancez **saisie_videothèque.php**, la page va juste afficher le texte « ajouter ».

Classe UniversalForm

Explications : La nouvelle classe **Form_videotheque** a été instancié en l'objet **\$monFormulaire**. Un identifiant (numéro) unique de formulaire lui a été attribué via le second paramètre (ici 1; le prochain formulaire présent dans le même script devra avoir un numéro différent). Il a ensuite été fait appel à la méthode **getOperation** de l'objet qui a pour rôle de renvoyer le type d'opération demandée ; ici l'ajout d'informations depuis un formulaire qui reste encore à construire. Cette méthode est bien entendu héritée de la classe mère **UniversalForm**. Noter le passage de la constante **UniversalForm::AJOUTER** qui est elle aussi déclarée dans la classe **UniversalForm** (voir les autres constantes possibles).

Vous avez remarqué que notre page web commence par l'inclusion des classes qui entrent en jeu dans notre script : la classe **UniversalForm** et notre classe **Form_videotheque**.

```
require_once('libs/classes/UniversalForm.class.php');
require_once('libs/classes/Form_videotheque.class.php');
```

Sans ces inclusions, l'exemple ne fonctionnerait pas et afficherait l'erreur suivante :

```
Fatal error: Class 'Form_videotheque' not found
```

Il est important de savoir que les classes doivent être chargées en mémoire avant leur appel. En effet, vu du fichier **saisie_videotheque.php** rien ne dit au code où trouver la classe **Form_videotheque**. Pour remédier au problème, 2 solutions : la moins élégante (celle mise en place ici) consiste à ajouter en début du fichier un **include** de chaque classe qui va être utilisée, c'est-à-dire la classe que l'on vient de créer **Form_videotheque.class.php** mais aussi tous les fichiers de classes mères tels que **UniversalForm.class.php**, **UniversalField.class.php**, etc.

Fort heureusement ceci peut être fait de manière automatique avec le petit bout de code suivant :

```
function chargerClasses($classe) {
    require_once($classe.'.class.php');
}
spl_autoload_register('chargerClasses');
```

Avec ce code, à chaque fois qu'une nouvelle classe sera requise, PHP ira charger le fichier correspondant, d'où l'intérêt de bien nommer vos fichiers contenant les classes avec le suffixe **.class.php**. Le code du fichier **saisie_videotheque.php** peut alors être le suivant :

```
function chargerClasses($classe) {
    require_once('libs/classes/'.$classe.'.class.php');
}
spl_autoload_register('chargerClasses');

$monFormulaire = new Form_videotheque(UniversalForm::AJOUTER, 1);
echo $monFormulaire->getOperation();
```

Il pourra aussi être judicieux d'inscrire le code de chargement des classes dans un fichier commun à toute l'application. Pour votre information, ceci est déjà fait pour vous dans la librairie **common.inc.php** d'UniversalWeb. Il vous suffit de l'inclure au début de votre code.

```
require_once('libs/common.inc.php');

$monFormulaire = new Form_videotheque(UniversalForm::AJOUTER, 1);
echo $monFormulaire->getOperation();
```

A noter : Il existe une méthode `setOperation($valeur)` qui permet de forcer l'opération en cours, ce qui peut être parfois utile...

Initialisation des données du buffer

Poursuivons la création de notre formulaire qui jusqu'ici est totalement absent. Notre formulaire va travailler avec un buffer de données qui lui est propre et qu'il faut créer. En informatique il est de bon ton d'initialiser les variables avant de s'en servir. C'est ce que nous allons maintenant faire. Dans notre fichier de classe `Form_videotheque.class.php` ajoutons le code ci-dessous, une méthode `init()` qui va faire appel à deux méthodes héritées : `initDonnees()` et `construitChamps()`.

```
class Form_videotheque extends UniversalForm {

    private $_tab_donnees = array();          //tableau des données (buffer) associées au formulaire

    //=====
    // Méthodes publiques
    //=====

    //initialisation des données (buffer) et construction des champs initialisés
    public function init() {
        $this->initDonnees();                //initialisation des données (buffer)
        $this->construitChamps();             //construction à vide... (cad avec données d'initiation)
    }
}
```

Un simple appel de cette nouvelle méthode depuis le gestionnaire du formulaire (fichier `saisie_videotheque.php`) ne changera pas grand-chose au résultat, du moins en apparence.

Ces deux classes sont des classes abstraites qu'il va nous falloir surcharger. Nous les avons réunies dans une seule méthode que nous venons de construire : `init()`.

`initDonnees()` : est l'endroit où l'on va initialiser les données de notre formulaire
`construitChamps()` : est l'endroit où l'on va construire notre formulaire

Implémentons maintenant la surcharge de la méthode `initDonnees()`.

```
class Form_videotheque extends UniversalForm {

    private $_tab_donnees = array();          //tableau des données associées au formulaire

    //=====
    // Méthodes protégées
    //=====
    // Initialisation des données de travail
    //=====

    protected function initDonnees() {
        //initialisations supplémentaires
        $this->_tab_donnees['titre'] = '';
        $this->_tab_donnees['annee'] = '';
    }

    //=====
    // Méthodes publiques
    //=====

    //initialisation des données et construction des champs initialisés
    public function init() {
        $this->initDonnees();                //initialisation des données
        $this->construitChamps();             //construction à vide... (cad avec données d'initiation)
    }
}
```

Classe UniversalForm

Notre classe va utiliser un tableau de données (le buffer du formulaire) qui sera utilisé pour remplir les (futurs) champs de notre formulaire. Le tableau en question est une propriété privée que l'on choisit d'appeler (par exemple) `$_tab_donnees`. A noter que toutes les déclarations de propriétés privées doivent commencer par le signe underscore. Il suffit maintenant d'initialiser ce tableau dans la méthode surchargée `initDonnees()`. Pour commencer nous allons prévoir deux champs texte :

titre : va recevoir le titre de notre film

annee : va recevoir l'année de production du film

Appeler ensuite la méthode `init()` créée depuis le gestionnaire du formulaire.

```
$monFormulaire = new Form_videotheque(UniversalForm::AJOUTER, 1);
echo $monFormulaire->getOperation();
$monFormulaire->init();
```

Vous remarquerez qu'il ne se passe toujours rien ! Normal, nous n'avons pas encore précisé à notre classe comment le formulaire était composé. Cela va se faire dans la méthode `construitChamps()` qu'il va falloir surcharger.

Construction des champs du formulaire

Nous y voici. Pour ce premier exemple, nous allons créer un formulaire simple contenant seulement deux champs texte. Comme on vient de le voir, l'un pour le titre du film, l'autre pour l'année de production. Entrez ce code pour implémenter la méthode `construitChamps()`.

```
//-----
// construction des champs du formulaire
//-----

protected function construitChamps() {
    //construction des champs du formulaire
    parent::construitChamps();
    $this->createField('text', 'titre', array(
        'newLine' => true,
        'dbfield' => 'titre',
        'label' => 'Titre du film',
        'testMatches' => array('REQUIRED'),
        'value' => $this->_tab_donnees['titre'],
    ));
    $this->createField('text', 'annee', array(
        'newLine' => true,
        'dbfield' => 'annee',
        'label' => 'Année de production',
        'testMatches' => array('REQUIRED', 'CHECK_INTEGER_4'),
        'value' => $this->_tab_donnees['annee'],
    ));
    //construction bouton Submit
    $this->createField('bouton', 'submit', array(
        'newLine' => true,
        'inputType' => 'submit',
        'label' => 'Ok',
        'lclass' => 'btn btn-primary',
        'value' => 'Ok'
    ));
}
```

Pour commencer, il est obligatoire de faire appel à la méthode parente `construitChamps()` par `parent::construitChamps()`.

Ensuite la méthode construit chaque champ grâce à l'appel de la méthode `createField()`. Cette méthode reçoit 3 paramètres :

Classe UniversalForm

- Le type de champ à choisir parmi `area`, `bouton`, `checkbox`, `comment`, `div`, `divfin`, `filtreselect`, `filtretext`, `hidden`, `image`, `radio`, `search`, `select`, `separateur` et `text`. Voir en annexe les spécifications de chaque type de champ. Noter que si le nom du type de champ n'existe pas, un champ de type « `text` » est créé par défaut.
- Le nom du champ : c'est une chaîne de caractère qui va permettre d'identifier le champ (id).
- Un tableau de propriétés du champ. Nous reviendrons ultérieurement sur ces propriétés pour les expliquer.

Nous venons de construire notre formulaire, il comporte 2 champs texte et 1 bouton de validation. Mais on ne voit encore rien sur l'écran ! C'est normal, nulle part nous n'avons encore dit à notre classe d'afficher notre formulaire tout frais.

Afficher le formulaire

L'affichage du formulaire va se faire via une nouvelle méthode à créer : la méthode `afficher()`. Voici son code :

```
//-----  
// affichage du formulaire  
//-----  
public function afficher() {  
    parent::afficher(); //permet d'ajouter des tests de construction du formulaire  
    //ATTENTION : les champs disabled ne renvoient aucun POST !!!  
    //Donc impossible de récupérer les données depuis une suppression  
    $enable = (!($this->getOperation() == self::CONSULTER) ||  
        ($this->getOperation() == self::SUPPRIMER));  
  
    $chaine = '';  
    $chaine.= '<form class="uf" action="'.$_SERVER['REQUEST_URI'].'" method="post" enctype="multipart/form-  
data">';  
    $chaine.= '<fieldset class="border p-3">';  
    $chaine.= '<h1>Formulaire</h1>';  
    $chaine.= $this->draw($enable);  
    $chaine.= '<p class="small">(*) Champ requis (1) Lecture seule</p>';  
    $chaine.= '</fieldset>';  
    $chaine.= '</form>';  
    return $chaine;  
}
```

Ce code est le code de base de tout formulaire créé avec les classe UniversalForm. `<form>` redirige l'action du formulaire vers sa propre page (on boucle donc) via le code `$_SERVER['REQUEST_URI']`.

- On remarquera que la première ligne de commande consiste à appeler la méthode parente `afficher()`. Ceci est **obligatoire**.
- A noter également la présence **obligatoire** dans les paramètres de `<form>` de la ligne `enctype="multipart/form-data"`. Sans cette information, nous ne pourrions pas utiliser de champs de type `file` qui permet le téléchargement de fichiers.
- Il est **obligatoire** de donner à notre formulaire la classe css « uf » (pour **U**niversal**F**orm).
- Enfin **il ne faut pas utiliser** le contenu du buffer `_tab_donnees` dans cette méthode car son contenu n'est pas mis à jour en cas d'erreur dans la saisie du formulaire. A la place utiliser les méthodes de l'objet (ex : `$this->field('nom_champ')->value()`)

Il faut maintenant appeler cette méthode dans notre page web gestionnaire du formulaire :

```
echo $monFormulaire->afficher();
```

Classe UniversalForm

Cela marche mais ce n'est pas beau ! Pourquoi ? Parce que jusqu'à présent nous n'avons pas écrit la moindre balise HTML qui va mettre en forme notre code. Voici ci-dessous le code complet du fichier **saisie-formulaire.php** qui servira de base à ce didacticiel pour poursuivre notre tutoriel sur l'utilisation des classes UniversalForm. Vous remarquerez que quelques balises bootstrap nous aident à mettre en page notre exercice. Il est donc préférable d'avoir quelques notions de base du Framework développé par Twitter. Notons aussi, au passage, encore, que notre page utilisera le codage UTF-8 qui est obligatoire pour l'utilisation de ces classes.

Fichier : Saisie_videotheque.php

```
<?php
require_once('libs/common.inc.php');

//création du formulaire
$monFormulaire = new Form_videotheque(UniversalForm::AJOUTER, 1);
//echo $monFormulaire->getOperation();

//création de la page HTML
echo '<!doctype html>';
echo '<html lang="fr">';
echo '<head>';
    echo '<meta charset="utf-8" />';
    //prise en compte pour l'affichage responsive
    echo '<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">';
    echo '<meta http-equiv="x-ua-compatible" content="ie=edge">';
    echo '<link rel="stylesheet" href="bootstrap-4.3.1-dist/css/bootstrap.min.css">'; //CSS Bootstrap
    echo '<title>Vidéotheque</title>';
echo '</head>';

echo '<body>';
    echo '<div class="container-fluid">';
        echo '<div class="row mt-3">';
            echo '<div class="col-6 ml-auto mr-auto">';

                //panel de l'application
                echo '<div class="row p-3">';
                    echo '<div class="col-12 bg-light ">';
                        echo '<p class="h1">Saisie vidéotheque</p>';
                        echo '<p class="lead">Version UniversalForm : '.UniversalForm::VERSION.'</p>';
                    echo '</div>';
                echo '</div>';

                //code propre à la page
                echo '<div class="row mt-3">';
                    echo '<div class="col-12">';

                        $monFormulaire->init();
                        echo $monFormulaire->afficher();

                    echo '</div>';
                echo '</div>';

            echo '</div>'; //fin colonne centrale
        echo '</div>';
    echo '</div>';

    //Chargement du code Javascript nécessaire au fonctionnement de bootstrap
    echo '<script src="js/jquery-3.3.1.min.js"></script>';
    echo '<script src="bootstrap-4.3.1-dist/js/bootstrap.bundle.min.js"></script>';

echo '</body>';
echo '</html>';
```

Voici le résultat : Nous avons un formulaire avec deux champs et un bouton OK

Saisie vidéothèque

Version UniversalForm : v3.15.0 (2019-08-02)

Formulaire

Titre du film*

Année de production*

(*) Champ requis (1) Lecture seule

Nous remarquerons que les libellés « Titre du film » et « Année de production » sont suivis d'une astérisque. Ceci pour préciser que la saisie de ces champs est obligatoire comme le spécifie la propriété `testMatches` que nous avons donné à nos champs et sur laquelle nous reviendrons plus tard.

Notons également que l'appel à la méthode `UniversalForm::VERSION` renvoie la version de la classe UniversalForm actuellement employée.

Mise en forme des champs

Comme on peut le voir, les champs ne sont pas très bien alignés. Grâce à quelques propriétés supplémentaires prédéfinies, nous allons pouvoir paramétrer chaque champ pour qu'il soit positionné différemment. Il y a 2 positionnements prédéfinis : **inline** et **online**.

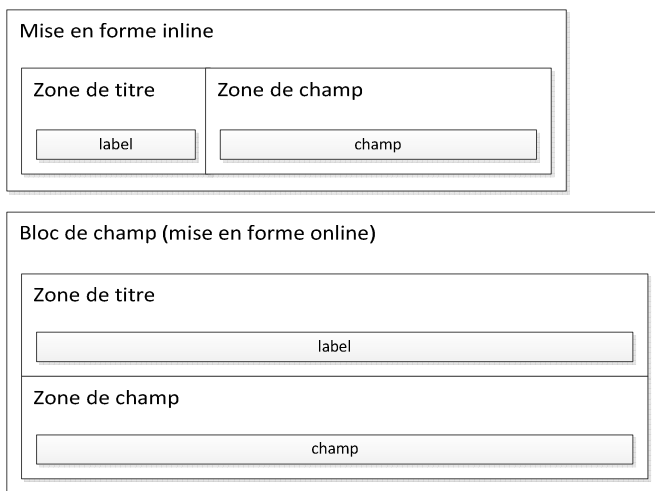
Un petit peu de langage est nécessaire pour savoir de quoi nous allons parler ci-dessous.

Le « label » est le libellé du champ. Il est contenu dans une « zone de titre »

Le « champ » de saisie proprement dit est contenu dans une « zone de champ »

NOTE : Dans les versions précédente, la « zone de titre » et la « zone de champ » étaient elles-mêmes encapsulés dans un « bloc de champ ». Maintenant il n'existe plus de « bloc de champ » pour le positionnement inline. Pas de changement pour le positionnement online.

Le schéma ci-dessous montre la position des champs et labels selon le positionnement prédéfini choisi :



Le positionnement par défaut est **inline**, ce qui signifie que chaque « zone de titre » et « zone de champ » sont contigus. Mais comme les largeurs de chaque zone n'ont pas été définies, chaque élément utilise la place par défaut.

Pour essayer un autre positionnement, il suffit d'ajouter à la description de chaque champ (dans le fichier **Form_videothèque.classe.php**) la propriété **design** et de lui fournir le type de positionnement souhaité. Par exemple pour faire apparaître le label « titre du film » au-dessus du champ, c'est-à-dire la « zone de titre » au-dessus de la « zone de champ », il faudra écrire :

```
$this->createField('text', 'titre', array(
    'newLine' => true,
    'design' => 'online',
    'dbfield' => 'titre',
    'label' => 'Titre du film',
    'testMatches' => array('REQUIRED'),
    'value' => $this->_tab_donnees['titre'],
));
```

Mais revenons à notre première présentation utilisant un positionnement **inline** (qui est le positionnement par défaut). Nous voulons aligner les deux champs. Pour ce faire, il suffit que les 2 « zones de titres » aient la même longueur. La propriété **long** ('l' comme label) permet de donner à

Classe UniversalForm

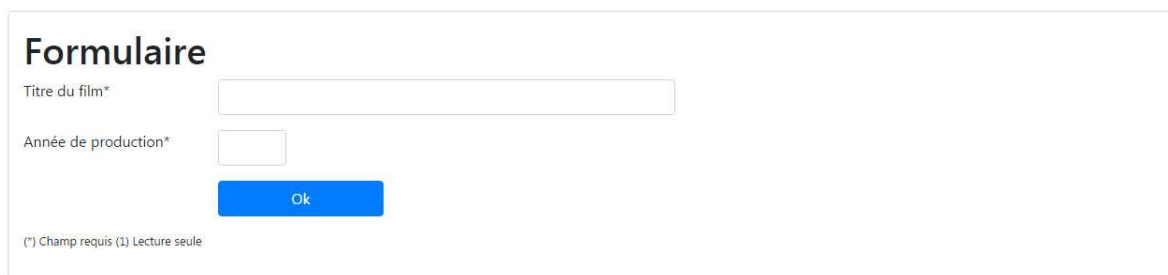
une zone de titre **inline** une longueur particulière, tandis que la propriété **clong** ('c' comme champ) permet de fixer la longueur de la zone de champ. On écrira donc :

```
$this->createField('text', 'titre', array(
    'newLine' => true,
    'dbfield' => 'titre',
    'label' => 'Titre du film',
    'llong' => 'col-2',
    'clong' => 'col-5',
    'testMatches' => array('REQUIRED'),
    'value' => $this->_tab_donnees['titre'],
));
$this->createField('text', 'annee', array(
    'newLine' => true,
    'dbfield' => 'annee',
    'label' => 'Année de production',
    'llong' => 'col-2',
    'clong' => 'col-1',
    'value' => $this->_tab_donnees['annee'],
));
//construction bouton Submit
$this->createField('bouton', 'submit', array(
    'newLine' => true,
    'inputType' => 'submit',
    'decalage' => 'col-2',
    'label' => 'Ok',
    'llong' => 'col-12',
    'lclass' => 'btn btn-primary',
    'clong' => 'col-2',
    'value' => 'Ok'
));
```

Les propriétés **clong** donnent la longueur des champs de saisie et la longueur maximum du bouton de validation du formulaire.

Ici, nous avons demandé à chaque « zone de titre » une longueur identique de **col-2** colonnes Bootstrap. Ceci permet d'aligner les libellés sur la même verticale. De même, vous remarquerez que la propriété **decalage** sur le bouton est à la valeur **col-2**. Cette propriété a pour effet d'aligner le bouton avec les 2 champs de saisie.

Le résultat obtenu est :



Cas particulier : d'une manière générale « llong » donne la longueur de la « zone de titre » et « clong » donne la longueur de la « zone de champ » dans le cas d'un design « inline ». Or, un bouton ne possède pas de « zone de titre ». Dans ce cas, **clong** doit recevoir la longueur du bouton (la zone de champ). On fournira à **llong** une valeur (en colonnes Bootstrap) qui dira au bouton de se dessiner sur la totalité de sa longueur (**col-12**), ou sur une partie (**col-6** par exemple pour un affichage de moitié). Si **llong** n'est pas renseigné, alors la longueur du bouton s'adapte à la longueur du libellé fourni par la propriété **label**.

A comprendre aussi : dans le cas d'un design **online**, la propriété **llong** n'est pas prise en compte car dans ce positionnement, la longueur de la « zone de titre » est forcément la même que la longueur de la « zone de champ » (**clong**).

Classe UniversalForm

Nous reviendrons sur toutes les propriétés qui nous permettront de personnaliser notre formulaire plus tard. Nous verrons aussi quels autres types de champs sont possibles et comment les mettre en œuvre. Mais en attendant nous allons maintenant nous intéresser à la gestion de ce formulaire car il faut en récupérer les valeurs saisies par l'utilisateur.

5. Gestion du formulaire

Comme nous l'avons vu dans la méthode d'affichage de notre formulaire, l'action de validation recharge le même code. Autrement dit, chaque fois que l'on clique sur OK, le code `saisie_formulaire.php` est rechargé... et donc le formulaire est réaffiché ! On boucle !

Voyons comment récupérer les valeurs saisies. La classe UniversalForm fournit la méthode publique `getAction()`. Appelée aussitôt à la suite de l'instanciation du formulaire, la méthode nous renseigne sur l'action à mener. Entrons et examinons le code suivant :

```
//création du formulaire
$monFormulaire = new Form_videotheque(UniversalForm::AJOUTER, 1);
$action = $monFormulaire->getAction();
echo 'Action générée = '.$action;
```

Lorsque la page est chargée, s'affiche alors Action générée = `ajouter`

Si l'on clique sur le bouton OK du formulaire alors s'affiche Action générée = `valid_ajouter`

Le formulaire nous a bien renseigné sur son état. La première fois, la méthode `getAction` n'a fait que nous retourner l'action demandée à gérer par le formulaire : « ajouter » un film. Le fait d'avoir cliqué sur le bouton OK, le formulaire signale que des données saisies sont disponibles en renvoyant l'action `valid_ajouter`. Il va alors suffire de tester cette action fournie par le formulaire pour le traiter.

Notons que ce `ajouter` ne vient pas de nulle part. Lorsque nous avons créé le formulaire (par l'instanciation de la classe), nous avons passé le paramètre `UniversalForm::AJOUTER` qui est une constante qui ne contient rien d'autre que le texte « ajouter ». Lorsque le formulaire a été validé, `getAction` a juste ajouté le préfixe « valid_ » à l'action de base.

Les constantes prédéfinies et utilisables sont les suivantes : `CONSULTER` ('consulter'), `AJOUTER` ('ajouter'), `MODIFIER` ('modifier'), `SUPPRIMER` ('supprimer'), `DUPLIQUER` ('dupliquer'), `RETIRER` ('retirer'), `ENVOYER` ('envoyer'). Vous pouvez aussi créer vos propres constantes, ou, tout simplement, passer le texte que vous voulez. Ainsi avec le code suivant, la validation du formulaire enverra comme action `valid_toto`. Essayez-le !

```
//création du formulaire
$monFormulaire = new Form_videotheque('toto', 1);
$action = $monFormulaire->getAction();
echo 'Action générée = '.$action;
```

Ces informations prises en compte, le code de la gestion de notre formulaire pourrait ainsi ressembler à ça :

```
//création du formulaire
$monFormulaire = new Form_videotheque(UniversalForm::AJOUTER, 1);
$action = $monFormulaire->getAction();

if ($action == 'ajouter') {
    $monFormulaire->init();
    echo $monFormulaire->afficher();
}
elseif ($action == 'valid_ajouter') {
    $lesDonnees = $monFormulaire->getData();
    echo '<pre>';
    print_r($lesDonnees);
    echo '</pre>';
}
```

Classe UniversalForm

```
}
```

La saisie suivante ...



... renvoie

```
Array
(
    [titre] => La guerre des étoiles
    [annee] => 1977
)
```

C'est un tableau qui contient la valeur saisie de chaque champ.

Chaque champ ? Pas vraiment. En réalité seuls les champs dont la propriété `dbfield` est renseignée sont renvoyés. Cette propriété associe le nom du champ à récupérer à la saisie (voir la description des champs `titre` et `annee` dans le fichier `Form_videotheque.class.php`).

Remarque : récupérer la valeur du bouton (qui est « Ok ») n'est pas ici indispensable. C'est pour cela qu'aucune propriété `dbfield` n'a été spécifiée pour le bouton.

Remarque : les valeurs des champs retournés sont déjà échappées... et donc déjà utilisables pour une éventuelle injection SQL. S'il faut juste afficher les données saisie, penser à supprimer les éventuels "slashes" contenus dans la saisie (ex : "L'enfer de Troie") via la fonction php `stripslashes()`.

Il existe deux autres constantes

VERSION : Celle-ci contient le numéro de version de la bibliothèque de classes UniversalForm. Pour afficher la version des scripts il suffit d'écrire : `echo UniversalForm::VERSION.`

COPYRIGHT : Celle-ci contient le copyright des classes UniversalForm. Pour afficher cette information il suffit d'écrire : `echo UniversalForm::COPYRIGHT.`

6. Types de champs

Plusieurs types de champs sont définis et couvrent la majorité des besoins pour construire des formulaires esthétiques et fonctionnels. Rappelons que l'on crée un champ par appel à la méthode `createField()` du formulaire. Chaque champ se bâtit à l'aide de propriétés qu'il faut passer en paramètre à la fonction sous forme d'un tableau de propriétés.

`createField(typeDeChamp, nomDeChamp, tableau de propriétés).`

Les propriétés disponibles selon le type de champ souhaité sont énumérées pour chaque type dans pages suivante. Bien qu'une majorité des propriétés soient communes, elles ne sont pas forcément toutes applicables d'un type de champ à un autre. Il faut bien vérifier les propriétés utilisées selon le type de champ utilisé. Voir en annexe un rappel des propriétés qui sont disponibles pour chaque type de champ.

NB : Les propriétés `titreHelp` et `labelHelp` permettent respectivement d'afficher un texte au survol du titre d'un champ ou de son label. Ces propriétés mettent naturellement en œuvre le dispositif « tooltip » de Bootstrap pour obtenir le résultat graphique suivant en lieu et place du simple « title » HTML :

Aide sur titre nb calories

Nombre de calories

100 200 500

Cependant, pour que cela fonctionne, il est nécessaire que la librairie javascript « popper » soit chargée (celle-ci est par défaut intégrée à la librairie Bootstrap `bootstrap.bundle.min.js`. Si cette librairie est chargée il n'est pas nécessaire de saisir la ligne ci-dessous).

```
echo '<script src="js/popper.min.js"></script>';
```

... et que la fonction jQuery suivante soit chargée :

```
$("[data-toggle='tooltip']").tooltip();
```

Exemple de chargement à positionner dans le footer de l'application après le chargement de la librairie jQuery :

```
$chaine.= '<script>';
$chaine.= '$(document).ready(function () {';
$chaine.= '$("[data-toggle=\'tooltip\']").tooltip();';
//autres fonctions jQuery à charger ...
$chaine.= '});';
$chaine.= '</script>';
```

NB : Le squelette UniversalWeb permet d'éviter cette écriture répétitive pour chaque page puisque ce code est écrit une fois pour toute dans la brique « footer » (pied de page) de chaque script.

Notez qu'une fois la fonction chargée, elle peut être également appliquée à n'importe quel élément HTML de votre application qui supporte et met en œuvre le tag `title`. Il suffit pour cela d'ajouter le code suivant `data-toggle="tooltip"` à chaque tag HTML qui inclue la propriété `title`. Exemple :

```
echo '<a href="#">';
```

Classe UniversalForm

```
echo '<span data-toggle="tooltip" data-placement="right" title="texte au survol du lien"></span>';  
echo '</a>';
```

De plus les propriétés `titreHelpPos` et `labelHelpPos` permettent de définir le positionnement de ce texte d'aide par rapport à son champ. A choisir parmi : `auto` (défaut), `left`, `center`, `right`, `bottom`). Si `auto` est sélectionné, alors c'est l'application qui décide de la meilleure position en fonction du contexte de la page. C'est donc la valeur conseillée.

NB : Au niveau HTML, chaque type de champ génère une structure constituée – comme nous l'avons vu plus tôt - d'une « zone de titre » et d'une « zone de champ ». Le modèle HTML de chaque type de champ est décrit à titre d'information dans les pages suivantes, information qui peut être utilisée afin de comprendre à quel tag HTML est associé chaque propriété de l'objet.

Faisons maintenant un petit tour d'horizon des types de champs possibles et des propriétés associables...

Type de champ « text »

C'est le plus commun, une simple zone de texte. Cela peut aussi être la plus complexe. C'est aussi le type de champ qui est créé par défaut si le paramètre `typeDeChamp` de la méthode `createField()` n'est pas reconnu (si vous avez entré n'importe quoi donc !)

Modèle html « inline » :

```
<zone titre : lalign llong lclass invisible>
  <label>label</label>
</zone titre>
<zone champ : clong invisible>
  <input : inputType enable maxlength spellcheck cclass value></input>
</zone champ>
```

Modèle html « online » :

```
<bloc champ : clong invisible>
  <zone titre : lalign lclass>
    <label>label</label>
  </zone titre>
  <zone champ>
    <input : inputType enable maxlength spellcheck cclass value>
  </input>
  </zone champ>
</bloc champ>
```

Propriété	Informations	Oblig.	Défaut
newLine	Booléen. Si <code>true</code> , l'élément doit être dessiné sur une nouvelle ligne du formulaire et deviendra de facto le premier champ de cette nouvelle ligne.	non	<code>false</code>
flexLine	Permet d'utiliser les possibilités <code>flex</code> de Bootstrap sur la totalité de la ligne du formulaire. Selon la valeur passée, les champs qui composent la ligne pourront être positionnés ou cadrés de façon personnalisée. Pour plus d'explications, lire ci-après.	non	" (vide)
dbfield	Nom de la donnée affectée au champ. La donnée (c'est à dire le contenu du champ <code>value</code>) pourra ainsi être récupérée lors de l'appel à la méthode <code>getData</code> . Si cette propriété n'est pas renseignée, la valeur donnée au champ (donc saisie par l'utilisateur) ne pourra pas être restituée.	non	


Classe UniversalForm

inputType	<p>Choix parmi : <code>button, checkbox, color, date, datetime, datetime-local, email, file, hidden, image, month, number, password, radio, range, reset, search, submit, tel, text, time, url, week.</code></p> <p>Voir la signification de chaque valeur au niveau HTML. D'une manière générale on utilisera :</p> <p><code>text, email, password, date</code> ou <code>datetime.</code></p> <p>NOTE : Dans le cas d'un type <code>file</code> qui sert à charger des fichiers, la classe renvoie non pas une chaîne de caractère, mais un tableau (de type <code>\$_FILES</code>) avec les champs suivants :</p> <pre>[name] => fiche.txt [type] => text/plain [tmp_name] => H:\ServeursWeb\xampp_5_6_11\tmp\phpAB83.tmp [error] => 0 [size] => 16851</pre> <p>voir sur Google la signification et l'usage de ces informations</p> <p>IMPORTANT : Pour qu'un champ de type <code>file</code> fonctionne et renvoie les données du fichier choisi il est impératif que l'instruction <code>enctype="multipart/form-data"</code> soit présente dans l'énoncée du formulaire (tag <code><form></code>)</p>	non	text
min	<p>Positionne l'attribut <code>min</code> de <code>HTML5</code> qui permet de fixer une valeur minimale à la saisie. Cette propriété est uniquement valable pour les champs input de type : <code>number, range, date, datetime, datetime-local, month, time</code> et <code>week.</code></p>	non	" (vide)
max	<p>Positionne l'attribut <code>max</code> de <code>HTML5</code> qui permet de fixer une valeur maximale à la saisie. Cette propriété est uniquement valable pour les champs input de type : <code>number, range, date, datetime, datetime-local, month, time</code> et <code>week.</code></p>	non	" (vide)
step	<p>Positionne l'attribut <code>step</code> de <code>HTML5</code> qui permet de fixer le pas de la saisie. Cette propriété est uniquement valable pour les champs input de type : <code>number, range, date, datetime, datetime-local, month, time</code> et <code>week.</code> Pour autoriser toutes les valeurs d'un flottant, passez <code>step</code> à la valeur <code>any.</code></p>	non	" (vide)

Classe UniversalForm

autocomplete	<p>Positionne l'attribut <code>autocomplete</code> qui spécifie si oui ou non le champ peut bénéficier du dispositif d'auto complétion. Celui-ci permet au navigateur de prédire la valeur saisie. Lorsque l'utilisateur commence à saisir le champ, le navigateur propose des valeurs en se basant sur les celles précédemment saisies. Les valeurs possibles sont :</p> <p>on : auto complétion active (valeur par défaut) off : auto complétion désactivée</p> <p>Le dispositif d'auto complétion est seulement pris en compte pour les champs <code><input></code> de type : text, search, url, tel, email, password, datepickers, range, et color.</p>	non	'on'
pattern	<p>Spécifie une expression régulière à laquelle la valeur de l'élément <code><input></code> doit être comparée pour être valide. Exemple : <code>.{6,}</code> => 6 caractères minimum</p>	non	" (vide)
design	<p>Design du champ.</p> <p>inline : la zone de titre est positionnée avant ou après la zone de champ.</p> <p>online : la zone de titre est positionnée au-dessus ou au-dessous de la zone de champ.</p>	non	inline
decalage	<p>Décalage préliminaire vers la droite du dessin du champ sur la ligne par ajout d'un certain nombre de colonnes Bootstrap vides (ex : <code>col-3</code> pour un décalage de 3 colonnes sur 12). Le décalage ne fait pas partie de la zone de champ.</p>	non	" (vide)
label	Libellé de la zone de titre.	non	'champ'
llong	<p>Longueur de la zone de titre dans le système de grille Bootstrap (ex : <code>col-5</code> pour une largeur de 5 colonnes sur 12).</p> <p>Si cette propriété est omise, alors la longueur de la zone de titre est limitée à la longueur du texte passé dans la propriété label.</p> <p>Cette propriété est seulement prise en compte dans le cas d'un design inline.</p>	non	Longueur du texte présent dans 'label'
lclass	<p>Classe CSS personnalisée du libellé fourni dans label. A savoir : Le code CSS s'applique à la « zone de titre ».</p>	non	" (vide)
lalign	(LibelleAlignement). Alignement du libellé (label) dans la zone de titre. Choisir parmi left , center , right , justify .	non	left
labelHelp	Chaine de caractère qui apparait au survol du label par la souris.	non	" (vide)
labelHelpPos	Position du labelHelp par rapport au label. Choisir parmi : auto , left , top , right , bottom . La valeur auto par laisse l'application décider de la meilleure position en fonction des autres éléments alentour.	non	auto

Classe UniversalForm

lpos	<p>Position de la zone de titre par rapport à la zone de champ.</p> <p>before : dans le cas d'un design inline, la zone de titre est affichée devant la zone de champ. Dans le cas d'un design online, la zone de titre est affichée au-dessus de la zone de champ.</p> <p>after : dans le cas d'un design inline, la zone de titre est affichée après la zone de champ. Dans le cas d'un design online, la zone de titre est affichée au-dessous de la zone de champ.</p>	non	before
labelPlus	<p>Si cette propriété est spécifiée (n'est pas vide) elle permet de scinder le libellé de la zone de titre en 2 libellés distincts, chacun positionné de part et d'autre de la zone de champ. Elle peut être par exemple utilisée pour ajouter une icône en plus du libellé et à l'opposé du libellé ... et pourquoi pas cliquable !</p> <p>Attention : la propriété est seulement valable pour un design online.</p> 	non	" (vide)
labelPlusHelp	Chaine de caractère qui apparait au survol du libellé de droite.	non	" (vide)
labelPlusHelpPos	<p>Position du labelPlusHelp par rapport au label. Choisir parmi : auto, left, top, right, bottom.</p> <p>La valeur auto par laisse l'application décider de la meilleure position en fonction des autres éléments alentour.</p>	non	auto
multiple	<p>Booléen. Concerne uniquement les champs de type file.</p> <p>true : il est possible de sélectionner plusieurs fichiers.</p> <p>false : un seul fichier sélectionnable.</p>	non	false
accept	<p>Chaine de caractères : Concerne uniquement les champs de type file.</p> <p>Doit recevoir la liste des extensions que le sélecteur de fichiers va seulement proposer séparées par une virgule.</p> <p>Exemple : '.txt, .csv'</p> <p>Par défaut cette propriété est vide, c'est-à-dire que le sélecteur de fichier va proposer tous les fichiers disponibles.</p>	non	" (vide)
clong	<p>Longueur de la zone de champ dans le système de grille Bootstrap. (ex : col-5 pour une longueur de champ de 5 colonnes sur 12).</p> <p>Si cette propriété est omise, alors la longueur de la zone de champ s'étend sur la totalité de la ligne du formulaire (100%).</p>	non	Donnée par le navigateur

Classe UniversalForm

cclass	Classe CSS personnalisée du champ de saisie. A savoir : le code CSS n'est pas appliqué à la zone de champ mais au tag html input .	non	" (vide)
maxlength	Nombre maximum de caractères que l'on peut saisir dans le champ. La valeur par défaut 0 signifie « aucune limite ».	non	0
spellcheck	Booléen. Active ou désactive le correcteur orthographique du navigateur pour le champ (les éventuelles fautes d'orthographe sont soulignées en rouge). Par défaut à true, ce qui signifie que cette option dépend du paramétrage du navigateur .	non	true
placeholder	Texte qui apparaît par défaut dans le champ. Cette propriété n'est pas prise en compte pour un champ de type file .	non	" (vide)
testMatches	Conditions de test (voir paragraphe détaillé sur les tests réalisables sur les champs).	non	null
value	Contenu (valeur) du champ texte. Important : ne pas donner de booléen true/false - ou alors sous forme d'une chaîne de caractères : ' true ' / ' false '.	non	" (vide)
complement	Information complémentaire. Cette propriété est utilisée dans le cadre d'un type de champ file . Dans ce cas-là, complement doit recevoir un tableau de type mime à comparer avec les données reçues d'un champ de type file (upload de fichier). Voir ci-dessous pour plus d'explications.	oui si inputType = file	" (vide)
autofocus	Booléen. Spécifie si l'élément doit recevoir le focus lorsque le formulaire est chargé. true : l'élément reçoit le focus lors du chargement du formulaire false : l'élément ne reçoit pas le focus au chargement du formulaire	non	false
erreur	Booléen. Indique si une erreur est à produire sur le champ.	non	false
liberreur	Libellé de l'erreur à lever.	non	" (vide)
liberreurHelp	Libellé à afficher sur le texte d'erreur au survol de la souris. Sert par exemple à donner des explications supplémentaires sur une erreur.	non	" (vide)
javascript	Code javascript à associer au champ	non	" (vide)
enable	Booléen. Active ou désactive le champ. Dans ce dernier cas, le champ reste visible, grisé et ne peut plus recevoir de saisie. NB : il n'est effectué aucun test sur les champs désactivés. NB : un champ désactivé renvoie NULL à l'appel de la méthode getData .	non	true

Classe UniversalForm

readonly	Booléen. Positionne le champ est en lecture seule. En lecture seule on ne peut pas modifier le contenu du champ. Il est alors grisé. Un petit '1' est ajouté au label correspondant pour confirmer cet état. NB : Contrairement à la propriété enable , même en lecture seule, la valeur du champ (son contenu) est renvoyée à l'appel de la méthode getData .	non	false
Invisible	Booléen. Si true, zone de titre et zone de champ sont rendues invisibles. NB : Le contenu du champ (même invisible) est toutefois renvoyé par la méthode getData .	non	false

Remarque : les valeurs retournées des champs de type 'text' sont déjà échappées... et donc déjà utilisables pour une injection SQL

A propos de la propriété « flexLine »

Cette propriété permet d'appliquer les possibilités « **flex** » de Bootstrap sur la totalité de la ligne de champs. Selon la valeur passée, les champs qui composent la ligne pourront être positionnés ou cadrés de façon personnalisée. Par exemple :

flex-row-reverse : cadre à droite et inverse l'ordre d'affichage des champs (par exemple pour cadrer les boutons à droite) dans la ligne.

justify-content-center : centre les champs dans la ligne.

justify-content-between : répartit les champs sur les côtés dans la ligne.

justify-content-around : répartit les champs sur le centre dans la ligne.

On peut aussi utiliser des commandes responsives :

flex-lg-row-reverse : cadre à droite pour les écran large (lg), sinon pas de changement

justify-content-sm-center : centre les champs à partir des petits écrans (sm).

justify-content-lg-between : répartit les champs sur les côtés à partir des écrans large (lg).

justify-content-xl-around : répartit les champs sur le centre à partir des écrans très larges (xl)

Noter que cette propriété s'applique à la définition d'une nouvelle ligne et n'a pas de lien direct avec un type de champ. Ainsi, **elle est sans effet si appliquée sur un champ pour lequel `newLine` est à `false`.**

Pour découvrir toutes les possibilité de cette propriété, reportez-vous à la documentation **flex** de Bootstrap.

A propos de la propriété « complement »

Cette propriété a un usage différent selon le type de champ. Par exemple :

1. Dans le cas d'un champ de type **search**, la propriété doit recevoir un tableau de couples **valeur => libellé** destiné à remplir l'extension (éventuelle, également appelée add-on) du champ de recherche (l'extension d'un champ **search** est une liste déroulante adjacente à l'objet **search**).
2. Dans le cas de champs de type **select** et **filtersselect**, la propriété doit recevoir la fonction callback chargée de remplir le sélecteur.

3. Dans le cas d'un champ du type `filtreText` la propriété doit recevoir un tableau du type `clé => valeur` qui viendra nourrir la liste déroulante des `option` du `select`. Exemple :

```
$_menu = array(
    '1' => 'Tous',
    '2' => 'Egal à',
    '3' => 'Différent de',
    '4' => 'Commence par',
    '5' => 'Contient',
    '6' => 'Ne contient pas',
    '7' => 'Se termine par');
```

4. Dans le cas d'un champ `text` de type `file` (chargement d'un fichier externe pour par exemple choisir un fichier à uploader), la propriété `complement` est utilisée pour recevoir un tableau des types mime autorisés (c'est-à-dire les types de fichiers autorisés à la sélection). Ainsi on pourra par exemple définir un tel champ de la sorte :

```
$this->createField('text', 'saisie'.$i, array(
    'dbfield' => 'saisie'.$i,
    'inputType' => 'file',
    'complement' => $this->_tabAutorise ));
```

avec (par exemple)

```
private $_tabAutorise = array(
    'image/gif' => array('.gif', 250000),
    'image/jpeg' => array('.jpg', 250000),
    'image/pjpeg' => array('.jpg', 250000),
    'image/png' => array('.png', 250000));
```

Le tableau de types mime doit impérativement avoir la structure ci-dessus, c'est-à-dire des tableaux de couples : `array('type mime', 'taille maxi de fichier autorisée')`.

A noter que la classe `UniversalForm` utilise ces informations pour valider le champ `text` de type `file`. Elle teste donc si le(s) fichier(s) est(sont) autorisé(s) (test sur le type mime) et si la taille du(des) fichier(s) ne dépasse pas la taille maxi spécifiée.

Exemple de construction

```
$this->createField('text', 'data0', array(
    'newLine' => true, //nouvelle ligne ? false par défaut
    'dbfield' => 'data0', //retour de la saisie
    'inputType' => 'file', //type d'input
    'design' => 'online', //inline (default) / online
    'decalage' => 'col-2', //décalage en colonnes bootstrap
    'label' => 'Data 0', //label
    'llong' => 'col-2', //longueur de la zone de titre
    'lclass' => 'texte_bleu', //classe du label
    'lalign' => 'left', //left (default) / right / center / justify
    'labelHelp' => 'Aide sur label', //aide sur le label
    'lpos' => 'before', //before (default) / after
    'clong' => 'col-8', //longueur de la zone de champ
    'cclass' => '', //classe de la zone de champ
    'maxlength' => 10, //nb caractères max en saisie
    'spellcheck' => true, //correction orthographique
    'placeholder' => 'Saisie', //texte pré-affiché
    'testMatches' => array('REQUIRED'), //test de la saisie
    'value' => $this->_tab_donnees['data0'], //valeur de la saisie
    'complement' => $this->_tabAutorise, //tableau des saisies autorisées
    'javascript' => '', //code javascript associé
    'enable' => true, //champ actif (dbfield renvoie NULL si false)
    'readonly' => false, //lecture seule (dbfield renvoie value si readonly)
    'invisible' => false //rend invisible le champ
));
```

Type de champ « radio »

Champ radio. Un seul champ radio n'a aucun sens. Il est nécessaire d'en créer au moins 2 pour que le dispositif soit efficient. Les boutons radio ayant le même fonctionnement font partie d'un même et unique groupe défini par la propriété **groupName**. C'est sur le premier radio bouton du groupe qu'il faudra définir certaines propriétés comme **border** ou encore positionner les erreurs.

Modèle html :

```
<legend : tlong tclass>titre</legend>          (premier bouton du groupe seulement)
<div : clong border>                            (premier bouton du groupe seulement)
<bloc champ : invisible>
  <zone titre>
    <label : lclass>label</label>
  </zone titre>
  <zone champ>
    <input radio : enable cclass value checked></input>
  </zone champ>
</bloc champ>
</div>                                          (dernier bouton du groupe seulement)
```

Propriété	Informations	Oblig.	Défaut
newLine	Booléen. Si true , l'élément doit être dessiné sur une nouvelle ligne du formulaire et deviendra de facto le premier champ de cette nouvelle ligne.	non	false
flexLine	Permet d'utiliser les possibilités flex de Bootstrap sur la totalité de la ligne du formulaire. Pour en savoir plus, lire la définition de cette propriété au chapitre 0.	non	" (vide)
dbfield	Nom de la donnée affectée au groupe de boutons radio. La donnée pourra ainsi être récupérée lors de l'appel à la méthode getData . Si cette propriété n'est pas renseignée, la saisie du champ ne pourra pas être récupérée. Important : Tous les boutons d'un même groupe de boutons radio doivent obligatoirement avoir la même valeur dbfield .	non	
groupName	Pour spécifier qu'un bouton radio appartienne à un certain groupe de boutons radio, tous doivent posséder le même nom. Cette propriété est obligatoire pour le fonctionnement naturel du bouton radio. Donner un nom identique à chaque bouton radio d'un même groupe. NB : Si la propriété est manquant, un message d'erreur rappellera ce manque.	oui	

Classe UniversalForm

design	<p>Design du champ. Particularité pour les champs radio, le bouton radio (zone de champ) et son libellé (zone de titre) sont toujours collés l'un à la suite de l'autre, en ligne.</p> <p>inline : le bloc champ (libellé + bouton) est affiché en ligne avec le prochain.</p> <p>online : le bloc champ (libellé + bouton) est affiché au-dessus ou au-dessous du prochain bloc de champ.</p>	non	inline
dpos	<p>(DesignPosition). Position du bouton radio dans le groupe. Choisir parmi :</p> <p>first : premier bouton radio du groupe. inter : bouton radio intermédiaire dans le groupe. last : dernier bouton radio du groupe.</p> <p>En cas d'oubli de cette propriété, ou d'une valeur différente, un message d'erreur est levé.</p>	oui	alone
titre	<p>Titre optionnel (libellé encore appelé légende) donné au groupe de boutons. Ce libellé est alors affiché avant le groupe de boutons. La propriété, ainsi que les propriétés associées tlong, tclass et titreHelp doivent être renseignés lors de la création du premier bouton du groupe. Ces propriétés sont ignorées sur les boutons suivants.</p>	non	" (vide)
tlong	<p>Longueur du titre dans le système de grille Bootstrap (ex : col-5 pour une largeur de 5 colonnes sur 12). Si cette propriété est omise, alors la longueur de la zone de titre s'étend sur la totalité de la ligne du formulaire (100%).</p> <p>Cette propriété est à fournir au premier bouton radio du groupe. Ignoré sur les suivants.</p>	non	100% de la largeur du formulaire
tclass	<p>Classe CSS personnalisée du titre de groupe de boutons. Cette propriété est à fournir au premier bouton radio du groupe. Ignorée sur les suivants.</p>	non	" (vide)
talign	<p>(TitreAlignement). Alignement du titre dans la zone de titre. Choisir parmi left, center, right, justify.</p>	non	left
titreHelp	<p>Chaine de caractère qui apparait au survol du titre de groupe par la souris.</p> <p>Cette propriété est à fournir au premier bouton radio du groupe. Ignorée sur les suivants.</p>	non	" (vide)
titreHelpPos	<p>Position du titreHelp par rapport au titre. Choisir parmi : auto, left, top, right, bottom.</p> <p>Cette propriété est à fournir au premier bouton radio du groupe. Ignorée sur les suivants.</p>	non	auto

Classe UniversalForm

decalage	<p>Décalage vers la droite du bouton radio (de son libellé et de l'éventuel titre) sur la ligne par ajout d'un certain nombre de colonnes Bootstrap vides (ex : <code>col-3</code> pour un décalage de 3 colonnes sur 12).</p> <p>Le même effet pourrait être obtenu en ajoutant un offset à la propriété <code>tlong</code> qui détermine la longueur du titre du groupe de radio boutons (ex : <code>offset-3</code>).</p> <p>Le décalage ne fait pas partie de la zone de champ.</p>	non	" (vide)
label	<p>Libellé de la zone de titre, c'est-à-dire du bouton radio.</p> <p>NB : la propriété <code>llong</code> n'est pas utilisée pour les champs radio. Le bouton, et son libellé étant indissociables, la longueur du champ est donnée par la propriété <code>clong</code>.</p> <p>Astuce : Par défaut le libellé est <code>'champ'</code>. Si on ne souhaite pas de libellé alors il faut entrer un <code>label</code> vide (").</p>	non	'champ'
lclass	Classe CSS personnalisée du label (et non de la zone de titre).	non	" (vide)
labelHelp	Chaine de caractère qui apparaitre au survol du label par la souris.	non	" (vide)
labelHelpPos	Position du <code>labelHelp</code> par rapport au label. Choisir parmi : <code>auto</code> , <code>left</code> , <code>top</code> , <code>right</code> , <code>bottom</code> . La valeur <code>auto</code> laisse l'application décider de la meilleure position en fonction des autres éléments alentour.	non	<code>auto</code>
lpos	<p>(LabelPosition). Ne pas confondre avec les propriétés <code>design</code> et <code>dpos</code>. Position du bouton radio par rapport à son libellé.</p> <p>after : le libellé est positionné après le bouton radio. before : le libellé est positionné devant le bouton radio.</p>	non	<code>before</code>
clong	<p>Longueur du groupe de boutons radio. Cette longueur doit être définie lors de la création du premier bouton radio. Elle est exprimée en colonnes de grille Bootstrap (ex : <code>col-5</code> pour une longueur de champ de 5 colonnes sur 12).</p> <p>Cette propriété n'est pas prise en compte par les boutons suivants. Si cette propriété est omise, un message d'erreur est levé.</p>	oui	" (vide)
cclass	Classe CSS personnalisée du bouton radio (et non de la zone de titre).	non	" (vide)
border	<p>Booléen ou code CSS.</p> <p>Bordure affichée autour du groupe de boutons radio. Le décalage éventuel n'est pas concerné. Plusieurs entrées possible :</p> <p>false : aucune bordure n'est dessinée autour du groupe de boutons. true : le groupe de boutons est entouré d'une bordure standard. CSS : code CSS personnalisé pour la bordure.</p> <p>Exemple : <code>'border' => 'border-bottom:2px dotted silver;'</code></p> <p>NB : Cette propriété doit impérativement être définie sur le premier bouton radio du groupe. Elle est sans effet sur les boutons suivants.</p>	non	<code>false</code>

Classe UniversalForm

value	Valeur qui sera renvoyée dans <code>dbfield</code> si le bouton radio est coché. Cela peut être n'importe quelle valeur : une chaîne, un entier. Important : ne pas donner de booléen <code>true</code> / <code>false</code> - ou alors sous forme d'une chaîne de caractères : <code>'true'</code> / <code>'false'</code> .	oui	0
checked	Booléen. Coche (<code>true</code>) ou décoche (<code>false</code>) le bouton radio. NB : Comme il ne peut y avoir qu'un seul bouton coché à la fois (principe des boutons radio), si plusieurs boutons ont la valeur <code>true</code> , le dernier à <code>true</code> est coché. Attention : entrer le booléen <code>true</code> / <code>false</code> et non <code>'true'</code> / <code>'false'</code> . Important : ne pas confondre <code>checked</code> et <code>value</code> .	non	<code>false</code>
erreur	Booléen. Indique si une erreur est à produire sur le groupe de bouton radio. Si une erreur est levée, le groupe de bouton radio est affiché en erreur. NB : l'erreur est toujours attribuée au groupe de boutons et non au bouton individuellement. NB : l'erreur du groupe doit impérativement être communiquée sur le premier bouton du groupe pour qu'elle soit prise en compte.	non	<code>false</code>
liberreur	Libellé de l'erreur à lever sur le groupe de boutons radio. NB : tout comme <code>erreur</code> cette propriété doit impérativement être renseignée sur le premier bouton radio du groupe.	non	" (vide)
liberreurHelp	Libellé à afficher sur le texte d'erreur au survol de la souris. Sert par exemple à donner des explications supplémentaire sur une erreur. NB : tout comme <code>erreur</code> et <code>liberreur</code> cette propriété doit impérativement être renseignée sur le premier bouton radio du groupe.	non	" (vide)
javascript	Code javascript à associer au champ	non	" (vide)
enable	Booléen. Active ou désactive le bouton radio. Dans ce dernier cas, le bouton (et son libellé) reste visible, grisé et ne peut plus être cliqué. NB : il n'est effectué aucun test sur les champs désactivés. NB : un champ désactivé renvoie NULL à l'appel de la méthode <code>getData</code> .	non	<code>true</code>

Classe UniversalForm

readonly	<p>Booléen. Positionne le bouton radio est en lecture seule. En lecture seule on ne peut pas cliquer le bouton radio. Il est alors grisé. Un petit '1' est ajouté au label correspondant pour confirmer cet état.</p> <p>NB : Même en lecture seule, la valeur du bouton radio est renvoyée à l'appel de la méthode getData.</p> <p>NB : En lecture seule on ne peut pas cocher ou décocher le champ. On peut par contre changer son état en cochant ou décochant un autre bouton radio.</p>	non	false
Invisible	<p>Booléen. Si true, le bloc champ (zone de titre et zone de champ) est rendu invisible.</p> <p>NB : Le contenu du champ (même invisible) est toutefois renvoyé par la méthode getData. Donc si un bouton radio invisible est coché par défaut, sa valeur sera quand même renvoyée à dbfield comme valeur choisie pour l'ensemble du groupe.</p>	non	false

Exemple de construction

```
//premier radio
$this->createField('radio', 'genre_homme_0', array(
    'newLine' => true,                //nouvelle ligne ? False par défaut
    'groupName' => 'optGenre0',       //le groupName est obligatoire
    'dbfield' => 'genre0',            //retour de la saisie
    'design' => 'inline',              //inline (défaut) / online
    'dpos' => 'first',                //first / last / inter
    'titre' => 'Titre Inline',         //titre sur premier élément seulement, sans effet sur les autres
    'tlong' => 'col-2 offset-2',      // titre sur 2 colonnes bootstrap décalé de 2 colonnes à droite
    'tclass' => 'text-right',         //classe du titre
    'titreHelp' => 'Genre de l\'abonné', //aide sur le titre
    'decalage' => 'col-2',            //décalage en colonnes bootstrap
    'label' => 'Homme',               //label
    'lclass' => 'bleu',               //classe du label
    'labelHelp' => 'Aide sur label Homme', //aide sur le label
    'lpos' => 'after',                //position label par rapport au bouton : before (défaut) / after
    'clong' => 'col-3',              //longueur champ en col bootstrap à définir sur premier du groupe
                                        //Sans effet sur les autres)
    'cclass' => '',                  //classe du bouton
    'border' => false,               //défaut : false. A définir une seule fois sur le premier élément
                                        //(false/true(encadrement par défaut) ou bordure personnalisée)
    'border' => 'border-bottom:2px dotted silver;', //bordure personnalisée
    'value' => 'Homme',               //valeur renvoyée dans dbfield si radio cliqué
    'checked' => true,                //coché (true) / décoché (false)
    'erreur' => true,                 //A définir une seule fois sur 1er élément. Ignoré sur autres.
    'liberreur' => 'TEST ERREUR',     //A définir une seule fois sur 1er élément. Ignoré sur autres.
    'liberreurHelp' => 'libelle erreur', //A définir une seule fois sur 1er élément. Ignoré sur autres.
    'javascript' => 'onclick="alert(\'Homme cliqué\');"', //code javascript associé
    'enable' => true,                 //disponible (dbfield renvoie NULL sinon)
    'readonly' => true,               //lecture seule (dbfield renvoie value si readonly)
    'invisible' => false              //rend invisible (true) le champ
));

//deuxième radio
$this->createField('radio', 'genre_femme_0', array(
    'newLine' => false,               //nouvelle ligne ? false par défaut
    'groupName' => 'optGenre0',       //le groupName est obligatoire
    'dbfield' => 'genre0',            //retour de la saisie
    'design' => 'inline',              //inline (défaut) / online
    'dpos' => 'inter',                //first / last / inter
    'decalage' => 'col-2',            //décalage en colonnes bootstrap
    'label' => 'Femme',               //label
    'lclass' => 'vert',               //classe du label
    'labelHelp' => 'Aide sur label Femme', //aide sur le label
    'lpos' => 'after',                //position label par rapport au bouton : before (default) / after
    'cclass' => '',                  //classe du bouton
    'value' => 'Femme',               //valeur renvoyée dans dbfield si radio cliquée
    'checked' => false,               //coché (true) / décoché (false)
    'javascript' => 'onclick="alert(\'Femme cliquée\');"', //code javascript associé
    'enable' => true,                 //disponible (dbfield renvoie NULL sinon)
    'readonly' => false,              //lecture seule (defield renvoi value si readonly)
    'invisible' => false              //rend invisible (true) le champ
));
```

```
));  
  
//troisième radio  
$this->createField('radio', 'genre_inconnu_0', array(  
    'newLine' => false,                //nouvelle ligne ? false par défaut  
    'groupName' => 'optGenre0',        //le groupName est obligatoire pour grouper les radios  
    'dbfield' => 'genre0',              //retour de la saisie  
    'design' => 'inline',                //inline (default) / online  
    'dpos' => 'last',                   //first / last / inter  
    'decalage' => 'col-2',               //décallage en colonnes bootstrap  
    'label' => 'Inconnu',                //label  
    'lclass' => 'rouge',                 //classe du label  
    'labelHelp' => 'Aide sur Inconnu',   //aide sur le label  
    'lpos' => 'after',                  //position label par rapport au bouton : before (default) / after  
    'cclass' => '',                      //classe du bouton  
    'value' => 'Inconnu',                //valeur renvoyée dans dbfield si radio cliquée  
    'checked' => false,                  //coché (true) / décoché (false)  
    'javascript' => 'onclick="alert(\'Inconnu cliquée\');"', //code javascript associé  
    'enable' => false,                   //disponible (dbfield renvoie NULL sinon)  
    'readonly' => false,                 //lecture seule (defield renvoi value si readonly)  
    'invisible' => false                 //rend invisible (true) le champ  
));
```

Type de champ « checkbox »

Case à cocher. Contrairement à un bouton radio, une case à cocher se suffit à elle-même pour donner un résultat. Bien que la notion de groupe ne soit pas nécessaire, toute case à cocher - même unique – **doit être définie comme appartenant à un groupe**. Ceci permet de pouvoir grouper virtuellement les cases dans un simple intérêt graphique (affichage plus serré, bordure groupée sur plusieurs cases, gestion des erreurs sur un groupe).

Modèle html :

```
<legend : tlong tclass>titre</legend>          (premier bouton du groupe seulement)
<div : clong border>                             (premier bouton du groupe seulement)
<bloc champ : invisible>
  <zone titre>
    <label : lclass>label</label>
  </zone titre>
  <zone champ>
    <input checkbox : enable cclass value checked></input>
  </zone champ>
</bloc champ>
</div>                                           (dernier bouton du groupe seulement)
```

Propriété	Informations	Oblig.	Défaut
newLine	Booléen. Si true , l'élément doit être dessiné sur une nouvelle ligne du formulaire et deviendra de facto le premier champ de cette nouvelle ligne.	non	false
flexLine	Permet d'utiliser les possibilités flex de Bootstrap sur la totalité de la ligne du formulaire. Pour en savoir plus, lire la définition de cette propriété au chapitre 0.	non	" (vide)
dbfield	Nom de la donnée affectée à la case à cocher. La donnée pourra ainsi être récupérée lors de l'appel à la méthode getData . Si cette propriété n'est pas renseignée, la saisie du champ ne pourra pas être récupérée.	non	
groupName	Nom (chaîne de caractère) qui permet de rassembler plusieurs cases à cocher sous le même groupe. Bien qu'obligatoire, cette propriété peut toutefois être omise dans le cas d'une checkbox seule. Par défaut, UniversalForm lui donnera le même nom que le champ défini dans createField .	oui non si seule	nom du champ
design	Design du champ. Particularité pour les champs checkbox, comme pour les radio, la case à cocher (zone de champ) et son libellé (zone de titre) sont toujours collés l'un à la suite de l'autre, en ligne. inline : le bloc champ (libellé + case à cocher) est affiché en ligne avec le prochain. online : le bloc champ (libellé + case à cocher) est affiché au-dessus ou au-dessous du prochain bloc de champ.	non	inline

Classe UniversalForm

dpos	<p>(DesignPosition). Position de la case à cocher radio dans le groupe. Choisir parmi</p> <p>alone : la case à cocher est autonome (seule). first : première case à cocher du groupe du groupe. inter : case à cocher intermédiaire dans le groupe. last : dernière case à cocher du groupe.</p> <p>En cas d'oubli de cette propriété, ou d'une valeur erronée, un message d'erreur est levé.</p>	oui	alone
titre	<p>Titre optionnel (libellé encore appelé légende) donné au groupe de cases à cocher. Ce libellé est alors affiché avant le groupe.</p> <p>La propriété, ainsi que les propriétés associées tlong, tclass et titreHelp doivent être renseignées lors de la création de la première case à cocher du groupe. Ces propriétés sont ignorées sur les cases suivantes.</p>	non	" (vide)
tlong	<p>Longueur du titre dans le système de grille Bootstrap (ex : col-5 pour une largeur de 5 colonnes sur 12). Si cette propriété est omise, alors la longueur de la zone de titre s'étend sur la totalité de la ligne du formulaire (100%).</p> <p>On peut utiliser cette propriété pour passer un décalage. Ex 'col-2 offset-3' définit un Titre sur 2 colonnes décalé de 3 colonnes sur la droite.</p> <p>Cette propriété est à fournir à la première case à cocher du groupe. Ignoré sur les suivantes.</p>	non	100% de la ligne du formulaire
tclass	<p>Classe CSS personnalisée du titre de groupe de cases à cocher. Cette propriété est à fournir à la première case à cocher du groupe. Ignorée sur les suivantes.</p>	non	" (vide)
talign	<p>(TitreAlignement). Alignement du titre dans la zone de titre. Choisir parmi left, center, right, justify.</p>	non	left
titreHelp	<p>Chaine de caractère qui apparait au survol du titre de groupe par la souris. Cette propriété est à fournir à la première case à cocher du groupe. Ignorée sur les suivantes.</p>	non	" (vide)
titreHelpPos	<p>Position du titreHelp par rapport au titre. Choisir parmi : auto, left, top, right, bottom. Cette propriété est à fournir au premier bouton radio du groupe. Ignorée sur les suivants.</p>	non	auto
decalage	<p>Décalage vers la droite de la case à cocher (et de son libellé) sur la ligne par ajout d'un certain nombre de colonnes Bootstrap vides (ex : col-3 pour un décalage de 3 colonnes sur 12).</p> <p>Le décalage ne fait pas partie de la zone de champ.</p>	non	" (vide)

Classe UniversalForm

label	<p>Libellé de la zone de titre, c'est-à-dire de la checkbox.</p> <p>NB : la propriété llong n'est pas utilisée pour les champs checkbox. La case à cocher, et son libellé étant indissociables, la longueur du champ est donnée par la propriété clong.</p> <p>Astuce : Par défaut le libellé est 'champ'. Si on ne souhaite pas de libellé alors il faut entrer un label vide (").</p>	non	'champ'
lclass	Classe CSS personnalisée du label (et non de la zone de titre).	non	" (vide)
labelHelp	Chaine de caractère qui apparaitre au survol du label par la souris.	non	" (vide)
labelHelpPos	Position du labelHelp par rapport au label. Choisir parmi : auto , left , top , right , bottom . La valeur auto laisse l'application décider de la meilleure position en fonction des autres éléments alentour.	non	auto
lpos	<p>Ne pas confondre avec les propriétés design et dpos.</p> <p>Position de la case à cocher par rapport à son libellé.</p> <p>after : le libellé est positionné après la case à cocher.</p> <p>before : le libellé est positionné devant la case à cocher.</p>	non	before
clong	<p>Longueur du groupe de cases à cocher. Cette longueur doit être définie lors de la création de la première case à cocher du groupe. Elle est exprimée en colonnes de grille Bootstrap (ex : col-5 pour une longueur de champ de 5 colonnes sur 12).</p> <p>Cette propriété n'est pas prise en compte par les checkbox suivantes. Si cette propriété est omise, un message d'erreur est levé.</p>	oui	" (vide)
cclass	Classe CSS personnalisée de la case à cocher (et non de la zone de titre).	non	" (vide)
border	<p>Booléen ou code CSS.</p> <p>Bordure affichée autour du groupe de cases à cocher. Le décalage éventuel n'est pas concerné. Plusieurs entrées possible :</p> <p>false : aucune bordure n'est dessinée autour du groupe.</p> <p>true : le groupe est entouré d'une bordure standard.</p> <p>CSS : code CSS personnalisé pour la bordure.</p> <p>Exemple : 'border' => 'border-bottom:2px dotted silver;'</p> <p>NB : Cette propriété doit impérativement être définie sur la première case à cocher du groupe. Elle est sans effet sur les cases suivantes.</p>	non	false

Classe UniversalForm

value	<p>Valeur qui sera renvoyée dans <code>dbfield</code> si la checkbox est cochée. Cela peut être n'importe quelle valeur : une chaîne, un entier.</p> <p>Important : ne pas donner de booléen <code>true</code> / <code>false</code> - ou alors sous forme d'une chaîne de caractères : <code>'true'</code> / <code>'false'</code>.</p> <p>Important : Ne pas confondre avec la propriété <code>checked</code> qui sert à donner l'état de la case à cocher !</p>	non	1
valueInverse	<p>Valeur qui sera renvoyée dans <code>dbfield</code> si la checkbox n'est pas cochée. Cela peut être n'importe quelle valeur : une chaîne, un entier.</p> <p>Important : ne pas donner de booléen <code>true</code> / <code>false</code> - ou alors sous forme d'une chaîne de caractères : <code>'true'</code> / <code>'false'</code>.</p>	non	0
checked	<p>Booléen. Coche (<code>true</code>) ou décoche (<code>false</code>) la checkbox.</p> <p>Attention : Entrer le booléen <code>true</code> / <code>false</code> et non <code>'true'</code> / <code>'false'</code>.</p> <p>Important : Ne pas confondre <code>checked</code> et <code>value</code>.</p>	non	<code>false</code>
erreur	<p>Booléen. Indique si une erreur est à produire sur le groupe. Si une erreur est levée, le groupe de cases à cocher est affiché en erreur.</p> <p>NB : L'erreur est toujours attribuée au groupe et non à la case à cocher individuellement.</p> <p>NB : l'erreur du groupe doit impérativement être communiquée sur la première case à cocher du groupe pour qu'elle soit prise en compte.</p>	non	<code>false</code>
liberreur	<p>Libellé de l'erreur à lever sur le groupe de checkbox.</p> <p>NB : tout comme <code>erreur</code> cette propriété doit impérativement être renseignée sur la première case à cocher du groupe.</p>	non	" (vide)
liberreurHelp	<p>Libellé à afficher sur le texte d'erreur au survol de la souris. Sert par exemple à donner des explications supplémentaire sur une erreur.</p> <p>NB : tout comme <code>erreur</code> et <code>liberreur</code> cette propriété doit impérativement être renseignée sur la première case à cocher du groupe.</p>	non	" (vide)
javascript	Code javascript à associer au champ	non	" (vide)
enable	<p>Booléen. Active ou désactive la case à cocher. Dans ce dernier cas, la checkbox (et son libellé) reste visible, grisé et ne peut plus être cliqué.</p> <p>NB : il n'est effectué aucun test sur les champs désactivés.</p> <p>NB : un champ désactivé renvoie <code>NULL</code> à l'appel de la méthode <code>getData</code>.</p>	non	<code>true</code>

Classe UniversalForm

readonly	<p>Booléen. Positionne la case à cocher en lecture seule. En lecture seule on ne peut pas cliquer la case. Elle est alors grisée. Un petit '1' est ajouté au label correspondant pour confirmer cet état.</p> <p>NB : même en lecture seule, la valeur de la checkbox est renvoyée à l'appel de la méthode getData.</p>	non	false
Invisible	<p>Booléen. Si true, le bloc champ (zone de titre et zone de champ) est rendu invisible.</p> <p>NB : Le contenu du champ (même invisible) est toutefois renvoyé par la méthode getData. Donc si une checkbox invisible est cochée, sa valeur sera quand même renvoyée à dbfield.</p>	non	false

Exemple de construction

```
$this->createField('checkbox', 'enfant', array(
    'newLine' => true,                //nouvelle ligne ? false par défaut
    'groupName' => 'enfant',         //le groupName est facultatif si dpos alone
    'dbfield' => 'enfant',            //retour de la saisie
    'design' => 'inline',             //inline (défaut) / online (forcé à inline pour dpos alone)
    'dpos' => 'alone',               //first / last / inter / alone
    'titre' => 'Tranches d\'ages',    //Titre (premier élément seulement, sans effet sur les autres)
    'tlong' => 'col-2',              //on veut un titre sur 2 colonnes bootstrap
    'tclass' => 'vert',              //style du titre
    'titreHelp' => 'Aide sur titre', //aide du titre
    'decalage' => 'col-2',           //décalage en colonnes bootstrap
    'label' => 'Enfant',             //label
    'lclass' => 'bleu',              //classe du label
    'labelHelp' => 'Aide sur le label Enfant', //aide sur le label
    'lpos' => 'after',               //position label par rapport à checkbox : before (défaut) / after
    'clong' => 'col-1',              //longueur du champ en colonnes bootstrap
    'cclass' => '',                  //classe de la checkbox
    'border' => true,                //défaut : false. A définir une seule fois sur le premier élément
                                      //checkbox (false /true (encadrement par défaut) ou bordure
                                      //personnalisée : voir ligne suivante pour en voir 1 exemple.

    'border' => 'border-bottom:2px dotted silver;', //bordure personnalisée
    'value' => 'oui',                //valeur renvoyée dans dbfield si checkbox cliquée
    'valueInverse' => 'non',         //valeur renvoyée dans dbfield si checkbox non cliquée
    'checked' => true,               //cochée (true) / décochée (false)
    'erreur' => true,                //A définir une seule fois sur le premier élément.
                                      //Ignoré sur les autres. Noter qu'il n'est pas usuel de monter
                                      //une erreur à la construction du champ. Celle-ci doit se faire à
                                      //la validation du code par sur des tests supplémentaires.

    'liberreur' => 'Test erreur',    //idem
    'liberreurHelp' => 'libelle aide erreur', //idem
    'javascript' => '',              //code javascript associé
    'enable' => true,                //disponible (dbfield renvoie NULL sinon)
    'readonly' => true,              //lecture seule (dbfield renvoie value si readonly)
    'invisible' => false             //rend invisible (true) le champ
));
```

Type de champ « switch »

Un switch a la même fonction finale qu'une case à cocher : la validation d'un option. Mais ici à la place la case à cocher est dessiné un petit interrupteur. Ce composant offre cependant moins de paramètres que la checkbox. En contrepartie, et contrairement à la case à cocher, il n'est pas nécessaire de faire appartenir un switch à un groupe. S'agissant d'une gestion directe par Bootstrap (vu du côté Bootstrap, le switch est une case à cocher customisée) le modèle HTML du switch est réduit au bloc de champ. Aucune bordure n'est possible sur un switch.

Modèle html :

```
<legend : tlong tclass>titre</legend>
<bloc champ : invisible clong>
  <input checkbox : enable checked value javascript></input>
  <label : lclass>label</label>
</bloc champ>
```

Propriété	Informations	Oblig.	Défaut
newLine	Booléen. Si true , l'élément doit être dessiné sur une nouvelle ligne du formulaire et deviendra de facto le premier champ de cette nouvelle ligne.	non	false
dbfield	Nom de la donnée affectée au switch. La donnée pourra ainsi être récupérée lors de l'appel à la méthode getData . Si cette propriété n'est pas renseignée, la saisie du champ ne pourra pas être récupérée.	non	
custom	Apparence du switch. Trois possibilités. switch : Apparence sous forme de switch (valeur par défaut) checkbox : Le champ a l'apparence d'une checkbox customisée au sens Bootstrap. radio : Le champ a l'apparence d'un bouton radio customisé au sens Bootstrap.	non	switch
titre	Titre optionnel (libellé encore appelé légende) donné au switch affiché avant le switch.	non	" (vide)
tlong	Longueur du titre dans le système de grille Bootstrap (ex : col-5 pour une largeur de 5 colonnes sur 12). Si cette propriété est omise, alors la longueur de la zone de titre s'étend sur la totalité de la ligne du formulaire (100%). On peut utiliser cette propriété pour passer un décalage. Ex ' col-2 offset-3 ' définit un Titre sur 2 colonnes décalé de 3 colonnes sur la droite.	non	100% de la ligne du formulaire
tclass	Classe CSS personnalisée du titre du switch.	non	" (vide)
talign	(TitreAlignement). Alignement du titre. Choisir parmi left , center , right , justify .	non	left

Classe UniversalForm

titreHelp	Chaine de caractère qui apparait au survol du titre du switch par la souris.	non	" (vide)
titreHelpPos	Position du titreHelp par rapport au titre. Choisir parmi : auto , left , top , right , bottom .	non	auto
decalage	Décalage vers la droite du switch (et de son libellé) sur la ligne par ajout d'un certain nombre de colonnes Bootstrap vides (ex : col-3 pour un décalage de 3 colonnes sur 12). Le décalage ne fait pas partie du bloc de champ.	non	" (vide)
label	Libellé du switch. NB : la propriété llong n'est pas utilisée pour les champs switch. Le switch et son libellé étant indissociables, la longueur du champ est donnée par la propriété clong . Astuce : Par défaut le libellé est 'champ' . Si on ne souhaite pas de libellé alors il faut entrer un label vide (").	non	'champ'
lclass	Classe CSS personnalisée du label.	non	" (vide)
labelHelp	Chaine de caractère qui apparaitre au survol du label par la souris.	non	" (vide)
labelHelpPos	Position du labelHelp par rapport au label. Choisir parmi : auto , left , top , right , bottom . La valeur auto laisse l'application décider de la meilleure position en fonction des autres éléments alentour.	non	auto
clong	Longueur du switch (switch + label compris). Elle est exprimée en colonnes de grille Bootstrap (ex : col-5 pour une longueur de champ de 5 colonnes sur 12).	oui	" (vide)
value	Valeur qui sera renvoyée dans dbfield le switch est allumé (coché). Cela peut être n'importe quelle valeur : une chaîne, un entier. Important : ne pas donner de booléen true / false - ou alors sous forme d'une chaîne de caractères : 'true' / 'false' . Important : Ne pas confondre avec la propriété checked qui sert à donner l'état du switch !	non	1
valueInverse	Valeur qui sera renvoyée dans dbfield si le switch n'est pas cochée. Cela peut être n'importe quelle valeur : une chaîne, un entier. Important : ne pas donner de booléen true / false - ou alors sous forme d'une chaîne de caractères : 'true' / 'false' .	non	0

Classe UniversalForm

checked	Booléen. Allume / coche (true) ou éteint / décoche (false) le switch. Attention : Entrer le booléen true / false et non ' true ' / ' false '. Important : Ne pas confondre checked et value .	non	false
erreur	Booléen. Indique si une erreur est à produire sur le switch.	non	false
liberreur	Libellé de l'erreur à lever sur le switch.	non	" (vide)
liberreurHelp	Libellé à afficher sur le texte d'erreur au survol de la souris. Sert par exemple à donner des explications supplémentaire sur une erreur.	non	" (vide)
javascript	Code javascript à associer au champ	non	" (vide)
enable	Booléen. Active ou désactive le switch. Dans ce dernier cas, le switch (et son libellé) reste visible, grisé et ne peut plus être cliqué. NB : il n'est effectué aucun test sur les champs désactivés. NB : un champ désactivé renvoie NULL à l'appel de la méthode getData .	non	true
readonly	Booléen. Positionne le switch en lecture seule. En lecture seule on ne peut pas basculer le switch. Il est alors grisé. Un petit '1' est ajouté au label correspondant pour confirmer cet état. NB : même en lecture seule, la valeur du switch est renvoyée à l'appel de la méthode getData .	non	false
invisible	Booléen. Si true, le bloc champ est rendu invisible. NB : Le contenu du champ (même invisible) est toutefois renvoyé par la méthode getData . Donc si une checkbox invisible est cochée, sa valeur sera quand même renvoyée à dbfield .	non	false

Exemple de construction

```

$checked = ($this->_tab_donnees['port_gratuit'] == 0);
$this->createField('switch', 'port_gratuit', array(
    'newLine' => true, //nouvelle ligne ? false par défaut
    'titre' => 'Offrir les frais de port', //titre sur le switch
    'tlong' => 'col-12 col-md-2', //longueur du titre
    'talign' => 'left', //alignement du titre
    'titreHelp' => 'Basculez si vous souhaitez offrir les frais de port', //aide affichée sur titre
    'titreHelpPos' => 'bottom', //position de l'aide
    'tclass' => 'text-success', //classe CSS du titre
    'dbfield' => 'ptgratuit', //indice du retour de la saisie
    'decalage' => 'col-2', //décallage affiché en cols bootstrap
    'label' => getLib('OFFRIR_FRAIS_PORT'), //label du switch
    'lclass' => 'text-warning', //classe du label
    'labelHelp' => 'Permettre d'offrir les frais de port', //aide sur le label
    'labelHelpPos' => 'bottom', //position de l'aide sur le label
    'clong' => 'col-12 col-md-4', //longueur de la zone de champ
    'value' => 'oui', //valeur de ptgratuit si switch allumé
    'valueInverse' => 'non', //valeur de ptgratuit si switch éteint
    'readonly' => false, //readonly
    'invisible' => false, //invisible
    'erreur' => false, //montée erreur
    'libErreur' => 'Ceci est une erreur', //libellé de l'erreur

```

Classe UniversalForm

```
'libErreurHelp' => 'Ceci est une aide sur erreur',      //aide sur le libellé d'erreur
'javascript' => '',                                       //code javascript à associer
'enable' => true,                                         //activé
'checked' => $checked                                    //allumé (coché) / éteint (décoché)
});
```


Type de champ « select »

Liste déroulante ou liste. Il n'y a pas de notion de groupe sur les listes déroulantes, donc pas de titre (ou légende) non plus.

Modèle html « inline » :

```
<zone titre : llong lclass lalign invisible>
  <label>label</label>
</zone titre>
<zone champ : clong invisible>
  <select : enable cclass>complement value</select>
</zone champ>
```

Modèle html « online » :

```
<bloc champ : clong invisible>
  <zone titre : lclass lalign>
    <label>label</label>
  </zone titre>
  <zone champ>
    <select : enable cclass>complement value</select>
  </zone champ>
</bloc champ>
```

Propriété	Informations	Oblig.	Défaut
newLine	Booléen. Si true , l'élément doit être dessiné sur une nouvelle ligne du formulaire et deviendra de facto le premier champ de cette nouvelle ligne.	non	false
flexLine	Permet d'utiliser les possibilités flex de Bootstrap sur la totalité de la ligne du formulaire. Pour en savoir plus, lire la définition de cette propriété au chapitre 0.	non	" (vide)
dbfield	Nom de la donnée affectée à la liste déroulante. La donnée pourra ainsi être récupérée lors de l'appel à la méthode getData . Si cette propriété n'est pas renseignée, la sélection ne pourra pas être récupérée.	non	
design	Design du champ. inline : la zone de titre est positionnée avant ou après la zone de champ. online : la zone de titre est positionnée au-dessus ou au-dessous de la zone de champ.	non	inline
multiple	Booléen. Permet une sélection multiple. En cas de sélection multiple le POST renvoie alors un tableau de valeurs (array)	non	false

Classe UniversalForm

size	Entier. Si cette propriété est positionnée et > 1 alors le sélecteur n'est plus une liste déroulante mais une liste simple d'une hauteur de size lignes. NB : En cas de sélection multiple, la valeur par défaut est de 4 lignes, la valeur 1 est impossible et la valeur 2 est la valeur minimum.	non	4
decalage	Décalage préliminaire vers la droite de la liste (et de son libellé) sur la ligne par ajout d'un certain nombre de colonnes Bootstrap vides (ex : col-3 pour un décalage de 3 colonnes sur 12). Astuce : le même effet pourrait être obtenu en ajoutant un offset à la propriété llong (longueur de la zone de titre) dans le cas d'un design inline ou à la propriété clong (longueur de la zone de champ) dans le cas d'un design online . (ex : offset-3). Le décalage ne fait pas partie de la zone de champ.	non	" (vide)
label	Libellé de la zone de titre.	non	'champ'
llong	Longueur de la zone de titre dans le système de grille Bootstrap (ex : col-5 pour une largeur de 5 colonnes sur 12). Si cette propriété est omise, alors la longueur de la zone de titre s'adapte au libellé fourni dans la propriété label . Cette propriété est seulement prise en compte dans le cas d'un design inline .	non	Longueur du libellé fourni à 'label'
lclass	Classe CSS personnalisée du libellé fourni dans label . A savoir : Le code CSS s'applique à la « zone de titre ».	non	" (vide)
lalign	(LibelleAlignement). Alignement du libellé (label) dans la zone de titre. Choisir parmi left , center , right , justify .	non	left
labelHelp	Chaine de caractère qui apparait au survol du label par la souris.	non	" (vide)
labelHelpPos	Position du labelHelp par rapport au label. Choisir parmi : auto , left , top , right , bottom . La valeur auto laisse l'application décider de la meilleure position en fonction des autres éléments alentour.	non	auto
lpos	Position de la zone de titre par rapport à la zone de champ. before : dans le cas d'un design inline , la zone de titre est affichée devant la zone de champ. Dans le cas d'un design online , la zone de titre est affichée au-dessus de la zone de champ. after : dans le cas d'un design inline , la zone de titre est affichée après la zone de champ. Dans le cas d'un design online , la zone de titre est affichée au-dessous de la zone de champ.	non	before

Classe UniversalForm

clong	Longueur de la zone de champ dans le système de grille Bootstrap. (ex : <code>col-5</code> pour une longueur de champ de 5 colonnes sur 12). Si cette propriété est omise, alors la longueur de la zone de champ s'adapte au plus grand contenu de la liste déroulante..	non	Longueur du plus grand contenu de la liste
cclass	Classe CSS personnalisée du champ de saisie. A savoir : le code CSS n'est pas appliqué à la zone de champ mais au tag html <code>select</code> .	non	" (vide)
testMatches	Conditions de test (voir paragraphe détaillé sur les tests réalisables sur les champs). Attention : Aucun test n'est effectué sur une liste à choix multiples (propriété <code>multiple</code> à <code>true</code>)	non	null
value	Valeur (option) sélectionnée dans la liste. C'est une des valeurs de la liste qui doit être créée par la fonction de callback déclarée dans la propriété <code>complement</code> et à écrire par le développeur. Cette valeur peut être une chaîne, un entier, etc. Dans le cas d'un sélecteur multiple (liste à choix multiple), cette propriété doit recevoir un tableau de valeurs. NB : ne pas donner de booléen <code>true</code> / <code>false</code> - ou alors sous forme d'une chaîne de caractères : <code>'true'</code> / <code>'false'</code> . NB : dans le cas d'un sélecteur simple, si aucune valeur n'est donnée (propriété non renseignée), alors la liste renverra la dernière option (valeur de la liste) disponible. NB : si la valeur donnée n'existe pas dans la liste (aucun mot clé <code>selected</code> présent), alors la liste renverra la première option (valeur de la liste). NB : dans le cas d'un sélecteur multiple, il est obligatoire de renseigner cette propriété en lui donnant un tableau de valeurs à sélectionner. Dans le cas contraire, PHP lèvera une erreur. Un tableau vide <code>array()</code> aura pour conséquence de ne sélectionner aucune option. De même, à la validation, si aucune option n'est sélectionnée, le champ <code>dbfield</code> renverra <code>NULL</code> . Si une valeur du tableau n'existe pas, elle est ignorée.	non	0
complement	Information complémentaire. Cette propriété reçoit le nom de la fonction callback qui est chargée de remplir la liste. Si la propriété est laissée vide, une erreur est levée.	oui	" (vide)
erreur	Booléen. Indique si une erreur est à produire sur le champ.	non	false
liberreur	Libellé de l'erreur à lever.	non	" (vide)
liberreurHelp	Libellé à afficher sur le texte d'erreur au survol de la souris. Sert par exemple à donner des explications supplémentaire sur une erreur.	non	" (vide)
javascript	Code javascript à associer au champ	non	" (vide)

Classe UniversalForm

enable	<p>Booléen. Active ou désactive le champ. Dans ce dernier cas, le champ reste visible, grisé et ne peut plus recevoir de saisie. Le curseur de la souris se change en panneau sens interdit au survol du champ désactivé.</p> <p>NB : il n'est effectué aucun test sur les champs désactivés.</p> <p>NB : un champ désactivé renvoie NULL à l'appel de la méthode getData.</p>	non	true
readonly	<p>Booléen.</p> <p>Positionne le champ est en lecture seule. En lecture seule on ne peut pas modifier le contenu du champ. Il est alors grisé. Un petit '1' est ajouté au label correspondant pour confirmer cet état.</p> <p>NB : même en lecture seule, la valeur du champ (son contenu) est renvoyée à l'appel de la méthode getData.</p>	non	false
Invisible	<p>Booléen.</p> <p>Si true, zone de titre et zone de champ sont rendues invisibles.</p> <p>NB : Le contenu du champ (même invisible) est toutefois renvoyé par la méthode getData.</p>	non	false

Exemple de construction

```
$this->createField('select', 'genre_film', array(
    'newLine' => true,                //nouvelle ligne ? false par défaut
    'dbfield' => 'genre_film',        //retour de la saisie
    'design' => 'online',              //inline (default) / online
    'multiple' => true,               //Sélection multiple est autorisée ? (false par défaut)
    'size' => 4,                      //hauteur du select en nombre de lignes visibles (1 par défaut)
    'decalage' => 'col-2',            //décalage en colonnes bootstrap
    'label' => 'Genre',               //label
    'llong' => 'col-2',               //longueur zone de titre (inutile dans le cas d'un design online)
    'lclass' => 'bleu',               //classe du label
    'lalign' => 'left',               //left (défaut) / right / center / justify
    'labelHelp' => 'Aide sur label genre film', //aide sur le label
    'lpos' => 'after',                //position label par rapport au champ : before (défaut) / after
    'clong' => 'col-3',               //longueur de la zone de champ
    'cclass' => '',                   //classe de la zone de champ
    'testMatches' => array('REQUIRED_SELECTION'), //test de la saisie
    'value' => $this->_tab_donnees['genrel'], //valeur de la saisie
    'complement' => 'fillSelect',     //fonction de callback qui doit remplir le select
    'erreur' => true,                 //montée erreur
    'liberreur' => 'TEST ERREUR',     //libellé de l'erreur
    'liberreurHelp' => 'libelle erreur', //Aide sur l'erreur
    'javascript' => '',               //code javascript associé
    'enable' => true,                 //active, désactive le champ (dbfield renvoie NULL si false)
    'readonly' => true,               //lecture seule (defield renvoi value si readonly)
    'invisible' => false              //rend invisible le champ
));
```

Type de champ « area »

Zone de saisie multi-lignes (mémo). Pas de notion de groupe.

Modèle html « inline » :

```
<zone titre : llong lclass lalign invisible>
  <label>label</label>
</zone titre>
<zone champ : clong invisible>
  <textarea : enable cclass rows>value</textarea>
</zone champ>
```

Modèle html « online » :


```
<bloc champ : clong cclass invisible>
  <zone titre : lclass lalign>
    <label>label</label>
  </zone titre>
  <zone champ>
    <textarea : enable cclass rows>value</textarea>
  </zone champ>
</bloc champ>
```

Propriété	Informations	Oblig.	Défaut
newLine	Booléen. Si true , l'élément doit être dessiné sur une nouvelle ligne du formulaire et deviendra de facto le premier champ de cette nouvelle ligne.	non	false
flexLine	Permet d'utiliser les possibilités flex de Bootstrap sur la totalité de la ligne du formulaire. Pour en savoir plus, lire la définition de cette propriété au chapitre 0.	non	" (vide)
dbfield	Nom de la donnée affectée au mémo. La donnée pourra ainsi être récupérée lors de l'appel à la méthode getData . Si cette propriété n'est pas renseignée, la sélection ne pourra pas être récupérée.	non	
design	Design du champ. inline : la zone de titre est positionnée avant ou après la zone de champ. online : la zone de titre est positionnée au-dessus ou au-dessous de la zone de champ.	non	inline

Classe UniversalForm

decalage	<p>Décalage préliminaire vers la droite du mémo sur la ligne par ajout d'un certain nombre de colonnes Bootstrap vides (ex : <code>col-3</code> pour un décalage de 3 colonnes sur 12).</p> <p>Astuce : le même effet pourrait être obtenu en ajoutant un offset à la propriété <code>llong</code> (longueur de la zone de titre) ou à la propriété <code>clong</code> (longueur de la zone de champ) (ex : <code>offset-3</code>).</p> <p>Le décalage ne fait pas partie de la zone de champ.</p>	non	" (vide)
label	<p>Label de la zone de titre.</p> <p>Si on ne veut pas afficher de label, il faut obligatoirement renseigner ce champ avec la valeur " (vide), sinon la valeur par défaut sera affichée.</p>	non	'champ'
llong	<p>Longueur de la zone de titre dans le système de grille Bootstrap (ex : <code>col-5</code> pour une largeur de 5 colonnes sur 12).</p> <p>Si cette propriété est omise, alors la longueur de la zone de titre s'adapte au contenu de la propriété <code>label</code>.</p> <p>Cette propriété est seulement prise en compte dans le cas d'un design <code>inline</code>.</p>	non	S'adapte au contenu de 'label'
lclass	<p>Classe CSS personnalisée du libellé fourni dans <code>label</code>.</p> <p>A savoir : Le code CSS s'applique à la « zone de titre ».</p>	non	" (vide)
lalign	<p>Alignement du libellé (label) dans la zone de titre. Choisir parmi <code>left</code>, <code>center</code>, <code>right</code>, <code>justify</code>.</p>	non	<code>left</code>
labelHelp	<p>Chaine de caractère qui apparait au survol du label par la souris.</p>	non	" (vide)
labelHelpPos	<p>Position du <code>labelHelp</code> par rapport au label. Choisir parmi : <code>auto</code>, <code>left</code>, <code>top</code>, <code>right</code>, <code>bottom</code>. La valeur <code>auto</code> laisse l'application décider de la meilleure position en fonction des autres éléments alentour.</p>	non	<code>auto</code>
lpos	<p>Position de la zone de titre par rapport à la zone de champ.</p> <p><code>before</code> : dans le cas d'un design <code>inline</code>, la zone de titre est affichée devant la zone de champ. Dans le cas d'un design <code>online</code>, la zone de titre est affichée au-dessus de la zone de champ.</p> <p><code>after</code> : dans le cas d'un design <code>inline</code>, la zone de titre est affichée après la zone de champ. Dans le cas d'un design <code>online</code>, la zone de titre est affichée au-dessous de la zone de champ.</p>	non	<code>before</code>

Classe UniversalForm

labelPlus	<p>Si cette propriété est spécifiée (n'est pas vide) elle permet de scinder le libellé de la zone de titre en 2 libellés distincts, chacun positionné de part et d'autre de la zone de champ. Elle peut être par exemple utilisée pour ajouter une icône en plus du libellé et à l'opposé du libellé ... et pourquoi pas cliquable !</p> <p>Attention : la propriété est seulement valable pour un design online.</p> 	non	" (vide)
labelPlusHelp	Chaine de caractère qui apparaît au survol du libellé de droite.	non	" (vide)
labelPlusHelpPos	Position du cadre labelPlusHelp par rapport au label. Choisir parmi : left , top , right , bottom , auto . La valeur auto laisse l'application décider de la meilleure position en fonction des autres éléments alentour.	non	auto
clong	<p>Longueur de la zone de champ dans le système de grille Bootstrap. (ex : col-5 pour une longueur de champ de 5 colonnes sur 12).</p> <p>Si cette propriété est omise, alors la longueur de la zone de champ est donnée par le navigateur.</p>	non	Donné par le navigateur
rows	Nombre de lignes de hauteur du mémo.	non	0
cclass	<p>Classe CSS personnalisée du mémo.</p> <p>A savoir : le code CSS n'est pas appliqué à la zone de champ mais au tag html textarea.</p>	non	" (vide)
maxlength	Nombre maximum de caractères que l'on peut saisir dans le champ. La valeur par défaut 0 signifie « aucune limite ».	non	0
spellcheck	Booléen. Active ou désactive le correcteur orthographique du navigateur pour le champ (les éventuelles fautes d'orthographe sont soulignées en rouge). Par défaut à true , ce qui signifie que cette option dépend du paramétrage du navigateur .	non	true
placeholder	Texte qui apparaît par défaut dans le champ.	non	" (vide)
testMatches	Conditions de test (voir paragraphe détaillé sur les tests réalisables sur les champs).	non	null
value	<p>Contenu (valeur) du mémo.</p> <p>Important : ne pas donner de booléen true / false - ou alors sous forme d'une chaîne de caractères : 'true' / 'false'.</p>	non	" (vide)
erreur	Booléen. Indique si une erreur est à produire sur le champ.	non	false
liberreur	Libellé de l'erreur à lever.	non	" (vide)

Classe UniversalForm

liberreurHelp	Libellé à afficher sur le texte d'erreur au survol de la souris. Sert par exemple à donner des explications supplémentaire sur une erreur.	non	" (vide)
javascript	Code javascript à associer au champ	non	" (vide)
enable	Booléen. Active ou désactive le champ. Dans ce dernier cas, le champ reste visible, grisé et ne peut plus recevoir de saisie. NB : il n'est effectué aucun test sur les champs désactivés. NB : un champ désactivé renvoie NULL à l'appel de la méthode getData .	non	true
readonly	Booléen. Positionne le champ est en lecture seule. En lecture seule on ne peut pas modifier le contenu du champ. Il est alors grisé. Un petit '1' est ajouté au label correspondant pour confirmer cet état. NB : même en lecture seule, la valeur du champ (son contenu) est renvoyée à l'appel de la méthode getData .	non	false
Invisible	Booléen. Si true , zone de titre et zone de champ sont rendues invisibles. NB : Le contenu du champ (même invisible) est toutefois renvoyé par la méthode getData .	non	false

Exemple de construction

```
$this->createField('area', 'commentaires', array(
    'newLine' => true,                //nouvelle ligne ? false par défaut
    'dbfield' => 'comments',          //retour de la saisie
    'design' => 'online',              //inline (défaut) / online
    'decalage' => 'col-2',            //décalage en colonnes bootstrap
    'label' => 'Commentaires',        //label
    'llong' => 'col-2',               //longueur zone de titre (inutile dans le cas d'un design online)
    'lclass' => 'bleu',               //classe du label
    'lalign' => 'right',               //left (défaut) / right / center / justify
    'labelHelp' => 'Aide sur commentaires', //aide sur le label
    'lpos' => 'before',               //position label par rapport au champ : before (défaut) / after
    'clong' => 'col-5',               //longueur de la zone de champ
    'rows' => 7,                      //hauteur du commentaire en nombre de lignes
    'cclass' => '',                   //classe de la zone de champ
    'maxlength' => 128,               //nb caractères max en saisie
    'spellcheck' => true,             //correction orthographique
    'placeholder' => 'Commentaire online', //texte pré-affiché
    'testMatches' => array('REQUIRED'), //test de la saisie
    'value' => $this->_tab_donnees['commentaires'], //valeur de la saisie
    'erreur' => true,                 //montée erreur
    'liberreur' => 'TEST ERREUR',     //libellé de l'erreur
    'liberreurHelp' => 'libelle erreur', //Aide sur l'erreur
    'javascript' => '',               //code javascript associé
    'enable' => true,                 //active, désactive le champ (dbfield renvoie NULL si false)
    'readonly' => false,              //lecture seule (defield renvoi value si readonly)
    'invisible' => false              //rend invisible le champ
));
```


Type de champ « bouton »

Bouton. Le design d'un bouton est systématiquement **inline**. Un bouton ne possède pas de zone de titre (en réalité il en possède une mais aucune propriété ne permet de la personnaliser).

Modèle html :

```
<zone champ : clong border invisible>
    <button : inputType enable llong lclass value>label</textarea>
</zone champ>
```

Propriété	Informations	Oblig.	Défaut
newLine	Booléen. Si true , l'élément doit être dessiné sur une nouvelle ligne du formulaire et deviendra de facto le premier champ de cette nouvelle ligne.	non	false
flexLine	Permet d'utiliser les possibilités flex de Bootstrap sur la totalité de la ligne du formulaire. Cette propriété est tout à fait appropriée pour le positionnement des boutons d'un formulaire. Pour en savoir plus, lire la définition de cette propriété au chapitre 0.	non	" (vide)
dbfield	Nom de la donnée affectée au mémo. La donnée pourra ainsi être récupérée lors de l'appel à la méthode getData . Si cette propriété n'est pas renseignée, la sélection ne pourra pas être récupérée.	non	
inputType	Type de bouton : submit : le bouton permet de valider le formulaire. Un formulaire peut posséder plusieurs boutons submit , ce qui peut permettre d'effectuer différentes actions. Pour savoir quel bouton submit aura été cliqué il faudra interroger la propriété dbfield . button : simple bouton qui permettra par exemple de gérer une action via javascript. reset : bouton dont la principale fonctionnalité est de vider le contenu des champs du formulaire. NB : si cette propriété ne reçoit pas l'un de ces type de boutons, une erreur est levée.	oui	text
decalage	Décalage vers la droite du dessin du champ sur la ligne par ajout d'un certain nombre de colonnes Bootstrap vides (ex : col-3 pour un décalage de 3 colonnes sur 12).	non	" (vide)
label	Libellé du bouton. En guise de libellé il est possible d'envoyer le code d'un glyphe du type font-awesome . Exemple : <code></code>	non	'champ'
labelHelp	Chaine de caractère qui apparait au survol de l'objet par la souris.	non	" (vide)

Classe UniversalForm

labelHelpPos	Position du labelHelp par rapport au label. Choisir parmi : auto, left, top, right, bottom .	non	auto
llong	<p>La longueur d'un bouton est donnée en nombre de colonnes Bootstrap dans la propriété clong. Cependant, llong permet de définir une sous-longueur à l'intérieur même de la longueur clong.</p> <p>Ainsi, une valeur de col-12 dessinera le bouton sur la longueur maximale réservée au bouton.</p> <p>Exemple 1 : si clong possède une longueur de 5 colonnes, une valeur llong de 12 colonnes dessinera le bouton sur les 5 colonnes (100%).</p> <p>Exemple 2 : si llong valait col-6, le bouton serait dessiné sur une longueur de 2.5 colonnes, c'est-à-dire la moitié des 5 colonnes (50%).</p> <p>Exemple 3 : si la propriété llong est omise, le bouton sera dessiné sur la largeur du libellé.</p>	non	Largeur du libellé
lclass	<p>Classe CSS personnalisée du bouton. On pourra très bien utiliser les classe standard Bootstrap btn btn-primary, btn btn-default etc.</p> <p>A noter : la classe CSS est fournie au tag html button.</p>	non	" (vide)
clong	Longueur du bouton exprimée en nombre de colonnes Bootstrap (ex : col-5). Voir llong pour affiner la longueur du bouton.	oui	" (vide)
cclass	<p>Classe CSS personnalisée du bloc de champ (et non du bouton fournie par lclass).</p> <p>A savoir : le code CSS est appliqué au bloc de champ et non au tag html button.</p>	non	" (vide)
border	<p>Booléen ou code CSS.</p> <p>Bordure affichée autour du bouton. Le décalage éventuel n'est pas concerné. Plusieurs entrées possible :</p> <p>false : aucune bordure n'est dessinée autour du bouton.</p> <p>true : le bouton est entouré d'une bordure standard.</p> <p>CSS : code CSS personnalisé pour la bordure. Exemple : 'border' => 'border-bottom:2px dotted silver;'</p>	non	false
value	Valeur renvoyée par dbfield à la validation du formulaire. Ne pas confondre avec label qui fournit le libellé au bouton.	non	" (vide)
showErreur	Booléen. Indique si les erreurs du champ doivent être affichées.	non	true
erreur	Booléen. Indique si une erreur est à produire sur le champ.	non	false
liberreur	Libellé de l'erreur à lever.	non	" (vide)

Classe UniversalForm

liberreurHelp	Libellé à afficher sur le texte d'erreur au survol de la souris. Sert par exemple à donner des explications supplémentaire sur une erreur.	non	" (vide)
javascript	Code javascript à associer au champ	non	" (vide)
enable	Booléen. Active ou désactive le champ. Dans ce dernier cas, le champ reste visible, sa couleur est estompée et il ne peut plus être cliqué. NB : un champ désactivé renvoie NULL à l'appel de la méthode getData .	non	true
invisible	Booléen. Si true , la zone de champ est rendue invisible. NB : le contenu du champ (même invisible) est toutefois renvoyé par la méthode getData .	non	false

Exemple de construction

```
//construction bouton Submit
$this->createField('bouton', 'submit', array(
    'newLine' => true,
    'dbfield' => 'btvalide',
    'decalage' => 'col-2',
    'inputType' => 'submit',
    'label' => 'Valider le formulaire',
    'llong' => 'col-12',

    'lclass' => 'btn btn-primary',
    'clong' => 'col-5',
    'cclass' => '',
    'border' => false,
    'border' => 'border-top:2px dotted silver;',
    'value' => 'Valider',
    'erreur' => true,
    'liberreur' => 'TEST ERREUR',
    'liberreurHelp' => 'libelle erreur',
    'javascript' => '',
    'enable' => true,

    'invisible' => false
));

//nouvelle ligne ? false par défaut
//retour de la saisie
//décalage en colonnes bootstrap
//submit (default), button, reset
//label
//col-12 dessine le bouton sur la totalité de sa largeur
//clong. Si vide, bouton dessiné à la largeur du libellé
//classes graphique du bouton
//longueur de la zone de champ (du bouton)
//classe personnalisée du bloc de champ
//défaut : false. false / true (encadrement par défaut)
//ou bordure personnalisée
//valeur renvoyée dans dbfield
//montée erreur
//libellé de l'erreur
//Aide sur l'erreur
//code javascript associé
//active/désactive le champ (dbfield renvoie NULL si
//false)
//rend invisible le champ
```

Type de champ « image »

Ce champ permet d'intégrer une image dans le formulaire dans un but "décoratif". Ce champ est en effet un petit peu particulier dans le sens où il ne permet aucune saisie. Il permet cependant de faire suivre une image lors de la manipulation du formulaire.

Modèle html « inline » :

```
<zone titre : llong lclass lalign invisible>
  <label>label</label>
</zone titre>
<zone champ : clong invisible>
  <img : cclass value border></img>
</zone champ>
```

Modèle html « online » :

```
<bloc champ : clong invisible>
  <zone titre : lclass lalign>
    <label>label</label>
  </zone titre>
  <zone champ>
    <img : cclass value border></img>
  </zone champ>
</bloc champ>
```

Propriété	Informations	Oblig.	Défaut
newLine	Booléen. Si true , l'élément doit être dessiné sur une nouvelle ligne du formulaire et deviendra de facto le premier champ de cette nouvelle ligne.	non	false
flexLine	Permet d'utiliser les possibilités flex de Bootstrap sur la totalité de la ligne du formulaire. Pour en savoir plus, lire la définition de cette propriété au chapitre 0.	non	" (vide)
dbfield	Nom de la donnée affectée à la zone. La donnée (c'est-à-dire le contenu du champ value) pourra ainsi être récupérée lors de l'appel à la méthode getData . Si cette propriété n'est pas renseignée, le contenu de value ne pourra pas être récupéré.	non	
design	Design du champ. inline : la zone de titre est positionnée avant ou après la zone de champ. online : la zone de titre est positionnée au-dessus ou au-dessous de la zone de champ.	non	inline

Classe UniversalForm

decalage	<p>Décalage vers la droite du bloc de champ par ajout d'un certain nombre de colonnes Bootstrap vides (ex : <code>col-3</code> pour un décalage de 3 colonnes sur 12).</p> <p>Astuce : le même effet pourrait être obtenu en ajoutant un <code>offset</code> à la propriété <code>llong</code> (longueur de la zone de titre) ou à la propriété <code>clong</code> (longueur de la zone de champ) (ex : <code>offset-3</code>).</p> <p>Le décalage ne fait pas partie de la zone de champ.</p>	non	" (vide)
label	Label de la zone de titre. Si on ne veut pas afficher de label, il faut obligatoirement renseigner ce champ avec la valeur " (vide).	non	'champ'
llong	<p>Longueur de la zone de titre dans le système de grille Bootstrap (ex : <code>col-5</code> pour une largeur de 5 colonnes sur 12).</p> <p>Si cette propriété est omise, alors la longueur de la zone de titre s'adapte au contenu de la propriété <code>label</code>.</p> <p>Cette propriété est seulement prise en compte dans le cas d'un design <code>inline</code>.</p>	non	" (vide)
lclass	<p>Classe CSS personnalisée du libellé fourni dans <code>label</code>.</p> <p>A savoir : Le code CSS s'applique à la « zone de titre ».</p>	non	" (vide)
lalign	(LibelleAlignement). Alignement du libellé (<code>label</code>) dans la zone de titre. Choisir parmi <code>left</code> , <code>center</code> , <code>right</code> , <code>justify</code> .	non	<code>left</code>
labelHelp	Chaine de caractère qui apparait au survol du label par la souris.	non	" (vide)
labelHelpPos	Position du <code>labelHelp</code> par rapport au label. Choisir parmi : <code>auto</code> , <code>left</code> , <code>top</code> , <code>right</code> , <code>bottom</code> . La valeur <code>auto</code> laisse l'application décider de la meilleure position en fonction des autres éléments alentour.	non	<code>auto</code>
lpos	<p>Position de la zone de titre par rapport à la zone de champ.</p> <p>before : dans le cas d'un design <code>inline</code>, la zone de titre est affichée devant la zone de champ. Dans le cas d'un design <code>online</code>, la zone de titre est affichée au-dessus de la zone de champ.</p> <p>after : dans le cas d'un design <code>inline</code>, la zone de titre est affichée après la zone de champ. Dans le cas d'un design <code>online</code>, la zone de titre est affichée au-dessous de la zone de champ.</p>	non	<code>before</code>
clong	<p>Longueur de la zone de champ dans le système de grille Bootstrap. (ex : <code>col-5</code> pour une longueur de champ de 5 colonnes sur 12).</p> <p>Si cette propriété est omise, alors la longueur de la zone de champ s'adapte à la taille de l'image.</p>	non	" (vide)

Classe UniversalForm

cclass	<p>Classe CSS personnalisée de l'image.</p> <p>A savoir : le code CSS est appliqué à l'image définie par le tag html img et non à la zone de champ.</p> <p>A savoir : les images sont responsives, c'est-à-dire qu'elles adaptent leur taille selon l'écran visualisant le site (ordinateur, tablette, téléphone)</p>	non	" (vide)
border	<p>Booléen ou code CSS.</p> <p>Bordure affichée autour de l'image. Le décalage éventuel n'est pas concerné. Plusieurs entrées possible :</p> <p>false : aucune bordure n'est dessinée autour de l'image. true : l'image est entouré d'une bordure standard. CSS : code CSS personnalisé pour la bordure. Exemple : 'border' => 'border-bottom:2px dotted silver;'</p>	non	false
value	url de l'image à afficher	non	" (vide)
erreur	Booléen. Indique si une erreur est à produire sur le champ.	non	false
liberreur	Libellé de l'erreur à lever.	non	" (vide)
liberreurHelp	Libellé à afficher sur le texte d'erreur au survol de la souris. Sert par exemple à donner des explications supplémentaire sur une erreur.	non	" (vide)
javascript	Code javascript à associer à l'image	non	" (vide)
invisible	<p>Booléen.</p> <p>Si true, zone de titre et zone de champ sont rendues invisibles.</p> <p>NB : Le contenu du champ (même invisible) est toutefois renvoyé par la méthode getData.</p>	non	false


Exemple de construction

```
$this->createField('image', 'image1', array(
    'newLine' => true,                //nouvelle ligne ? False par défaut
    'dbfield' => 'image1',           //retour de la saisie
    'design' => 'inline',             //inline (défaut) / online
    'decalage' => 'col-2',           //décalage en colonnes bootstrap
    'label' => 'Avion',              //label
    'llong' => 'col-2',              //longueur zone de titre (pour design inline seulement)
    'lclass' => 'rouge',             //classe du label
    'lalign' => 'right',             //left (défaut) / right / center / justify (alignement label)
    'labelHelp' => 'Aide sur label Avion', //aide sur le label
    'lpos' => 'before',              //position label par rapport à l'image : before (default) / after
    'clong' => 'col-5',              //longueur du champ en colonnes bootstrap
    'cclass' => '',                  //classe de l'image
    'border' => true,                 //défaut : false. false / true (encadrement par défaut)
    'border' => 'border-bottom:2px dotted silver;', //... ou bordure personnalisée
    'value' => 'images/alphaJet.jpg', //valeur renvoyée dans dbfield
    'erreur' => true,                 //montée erreur
    'liberreur' => 'TEST ERREUR',     //libellé de l'erreur
    'liberreurHelp' => 'libelle erreur', //Aide sur l'erreur
    'javascript' => 'onclick="alert(\'Avion cliqué\');"', //code javascript associé
    'invisible' => false              //rend invisible (true) le champ
));
```

Type de champ « search »

Ce champ fournit une zone de recherche complète formée d'un champ de saisie texte (La zone de champ) et d'un bouton de validation de la recherche (la zone de titre). Comme les boutons, le design d'un champ **search** est toujours **inline**.

Il est possible d'ajouter au champ une extension - un « addon » - (ici à gauche), c'est-à-dire une petite liste déroulante pour cibler une recherche.

A screenshot of a search field. On the left, there is a green dropdown menu labeled 'Choix 3' with a downward arrow. To its right is a text input field containing the word 'recherche'. On the far right of the input field is a green button with a white magnifying glass icon.

Modèle html :

```
<zone champ : clong invisible>
  <addon></addon>
  <input : inputType enable cclass value></input>
  <button type:submit enable llong lclass>label</button>
</bloc champ>
```

Propriété	Informations	Oblig.	Défaut
newLine	Booléen. Si true , l'élément doit être dessiné sur une nouvelle ligne du formulaire et deviendra de facto le premier champ de cette nouvelle ligne.	non	false
flexLine	Permet d'utiliser les possibilités flex de Bootstrap sur la totalité de la ligne du formulaire. Pour en savoir plus, lire la définition de cette propriété au chapitre 0.	non	" (vide)
dbfield	<p>Nom de la donnée affectée au champ. La donnée (c'est à dire le contenu du champ value) pourra ainsi être récupérée lors de l'appel à la méthode getData. Si cette propriété n'est pas renseignée, la valeur donnée au champ (donc saisie par l'utilisateur) ne pourra pas être restituée.</p> <p>Si l'objet contient une extension (addon = true), la valeur sera renvoyée sous forme d'un tableau à 2 entrées. L'entrée à la clé 0 correspond au choix dans la liste déroulante d'extension, l'entrée à la clé 1 correspond à la saisie faite par l'utilisateur dans le champ de saisie. Exemple :</p> <pre>Array ([0] => choix3 [1] => toto)</pre>	non	

Classe UniversalForm

inputType	<p>Choix parmi : button, checkbox, color, date, datetime, datetime-local, email, file, hidden, image, month, number, password, radio, range, reset, search, submit, tel, text, time, url, week.</p> <p>Voir la signification de chaque valeur au niveau HTML.</p> <p>NOTE : Les valeurs possibles de ce champ sont strictement les même que pour un champ de type text. Cependant, s'agissant ici d'un champ search certaines valeurs comme file, submit ou autres sont tout à fait inappropriées... Qui peut le plus peut le moins ! La valeur par défaut text devrait correspondre 99.99% du temps !</p>	non	text
addon	<p>Booléen. Si true, l'objet va afficher une extension (petite liste déroulante d'informations supplémentaires). Il faut alors renseigner les propriétés apos, aclass, complement ainsi que value.</p>	non	false
apos	<p>(AddonPosition). Position de l'extension par rapport à la zone de saisie. Choisir parmi before ou after.</p>	non	before
aclass	<p>(AddonClasse). Classe CSS personnalisée de l'extension. S'agissant en réalité d'un bouton de type button il est possible de styler l'extension avec du CSS standard Bootstrap tel que 'btn btn-primary', 'btn btn-default' etc.</p> <p>A noter : la classe CSS est fournie au tag html button.</p>	non	" (vide)
complement	<p>Information complémentaire.</p> <p>Ici la propriété complement doit recevoir les informations qui vont permettre de construire la petite liste déroulante de l'extension. Il faut lui passer un tableau de couples valeur => libelle.</p> <p>Exemple : <code>array('choix1' => 'Choix 1', 'choix2' => 'Choix 2')</code>.</p> <p>Note : Il est possible d'ajouter un séparateur entre deux options. Pour se faire, ajouter separateur ou separator dans le tableau.</p> <p>Exemple : <code>array('choix1' => 'Choix 1', 'separateur', 'choix2' => 'Choix 2')</code>.</p>	oui si addon est 'true'	" (vide)
titre	<p>Titre optionnel (libellé encore appelé légende).</p>	non	" (vide)
tlong	<p>Longueur du titre dans le système de grille Bootstrap (ex : col-5 pour une largeur de 5 colonnes sur 12). Si cette propriété est omise, alors la longueur de la zone de titre s'étend sur la totalité de la ligne du formulaire (100%).</p> <p>On peut aussi passer à cette propriété un décalage. Ex col-2 offset-3 définit un Titre sur 2 colonnes décalé de 3 colonnes sur la droite.</p>	non	100% de la ligne du formulaire

Classe UniversalForm

tclass	Classe CSS personnalisée du titre.	non	" (vide)
talign	(TitreAlignement). Alignement du titre dans la zone de titre. Choisir parmi left , center , right , justify .	non	left
titreHelp	Chaine de caractère qui apparait au survol du titre par la souris.	non	" (vide)
titreHelpPos	Position du titreHelp par rapport au titre. Choisir parmi : auto , left , top , right , bottom . La valeur auto laisse l'application décider de la meilleure position en fonction des autres éléments alentour.	non	auto
decalage	Décalage préliminaire vers la droite du dessin du champ sur la ligne par ajout d'un certain nombre de colonnes Bootstrap vides (ex : col-3 pour un décalage de 3 colonnes sur 12). Le même effet pourrait être obtenu en ajoutant un offset au champ (ex : offset-3) via la propriété clong (et non pas llong !) qui détermine la longueur de la zone de titre. Le décalage ne fait pas partie de la zone de champ.	non	" (vide)
label	Libellé de la zone de titre. Si la propriété n'est pas renseignée, alors la valeur par défaut s'applique et affiche champ . Si le libellé est vide alors le bouton affiche une icône de loupe (font-awesome). On peut aussi affiche le texte de son choix, y compris une icône du type font-awesome en passant le code suivant " " (qui correspond ici d'ailleurs à la loupe).	non	champ
llong	La longueur d'un champ search est donnée en nombre de colonnes Bootstrap à la propriété clong . Cependant, llong permet de définir la sous-longueur du bouton de validation associé. Ainsi, une valeur de col-12 dessinera le bouton sur la longueur maximale réservée au bouton. Exemple 1 : une valeur llong de 12 colonnes dessinera le bouton sur la totalité de sa zone réservée (100%). Exemple 2 : une valeur llong de 6 colonnes dessinera le bouton à moitié. (50%). Exemple 3 : si la propriété llong est omise, le bouton sera dessiné sur la largeur du libellé.	non	Largeur du libellé
lclass	Classe CSS personnalisé du bouton de l'objet search . On pourra très bien utiliser les classe standard Bootstrap ' btn btn-primary ', ' btn btn-default ' etc. A noter : la classe CSS est fournie au tag html button .	non	" (vide)
labelHelp	Chaine de caractère qui apparait au survol de l'objet par la souris.	non	" (vide)

Classe UniversalForm

labelHelpPos	Position du labelHelp par rapport au label. Choisir parmi : auto , left , top , right , bottom . La valeur auto laisse l'application décider de la meilleure position en fonction des autres éléments alentour.	non	auto
lpos	Position du champ de saisie par rapport au bouton. before : le champ de saisie est affiché devant le bouton. after : le champ de saisie est affiché après le bouton.	non	before
clong	Longueur de l'objet search (bloc de champ) dans le système de grille Bootstrap (ex : col-5 pour une longueur de champ de 5 colonnes sur 12). Si cette propriété est omise, alors la longueur de la zone de champ est définie par le navigateur.	non	" (vide)
cclass	Classe CSS personnalisée du champ de saisie. A savoir : le code CSS n'est pas appliqué à la zone de champ mais au tag html input .	non	" (vide)
maxlength	Nombre maximum de caractères que l'on peut saisir dans le champ. La valeur pas défaut 0 signifie « aucune limite ».	non	0
spellcheck	Booléen. Active ou désactive le correcteur orthographique du navigateur pour le champ (les éventuelles fautes d'orthographe sont soulignées en rouge). Par défaut à true , ce qui signifie que cette option dépend du paramétrage du navigateur .	non	true
placeholder	Texte qui apparaît par défaut dans le champ.	non	" (vide)
value	Contenu (valeur) du champ de saisie. Important : ne pas donner de booléen true/false - ou alors sous forme d'une chaîne de caractères : 'true'/'false' . 1 - Si l'objet n'affiche pas d'extension (propriété addon = false), alors la valeur du champ est une valeur simple. 2 - Si l'objet affiche une extension (propriété addon = true), alors la valeur retournée (et celle saisie) doit être un tableau à 2 entrées contenant la valeur de l'extension, et la valeur du champ de recherche. Exemple : array('choix2', '') . Sur cet exemple, le libellé de l'option du menu déroulant d'extension dont la valeur est choix2 sera affiché sur l'extension ; la zone de saisie sera vide. NB : dans le choix d'une extension, le contenu de dbfield renvoie lui aussi un tableau.	oui si addon est true	" (vide)
erreur	Booléen. Indique si une erreur est à produire sur le champ.	non	false
liberreur	Libellé de l'erreur à lever.	non	" (vide)

Classe UniversalForm

liberreurHelp	Libellé à afficher sur le texte d'erreur au survol de la souris. Sert par exemple à donner des explications supplémentaire sur une erreur.	non	" (vide)
javascript	Code javascript à associer au champ	non	" (vide)
enable	Booléen. Active ou désactive le champ. Dans ce dernier cas, le champ reste visible, grisé et ne peut plus recevoir de saisie. Le curseur de la souris se change en panneau sens interdit au survol du champ désactivé. NB : il n'est effectué aucun test sur les champs désactivés. NB : un champ désactivé renvoie NULL à l'appel de la méthode getData .	non	true
readonly	Booléen. Positionne le champ en lecture seule. En lecture seule on ne peut pas modifier le contenu du champ. Il est alors grisé. Un petit '1' est ajouté au label correspondant pour confirmer cet état. NB : Même en lecture seule, la valeur du champ (son contenu) est renvoyée à l'appel de la méthode getData .	non	false
Invisible	Booléen. Si true , la zone de champ est rendue invisible. NB : Le contenu du champ (même invisible) est toutefois renvoyé par la méthode getData .	non	false

Exemple de construction

```
//ajout d'un champ de recherche dans un champ de type recherche, le bouton fait office de label
$this->createField('search', 'recherche', array(
    'newLine' => true,                                //nouvelle ligne ? False par défaut
    'addon' => true,                                  //ajout d'une liste déroulante addon
    'aclass' => 'btn btn-success',                    //classe CSS de l'addon
    'apos' => 'before',                               //position de l'addon par rapport à la saisie
    'complement' => array(                            //contenu de l'addon (liste déroulante composée de la valeur
        'choix1' => 'Choix 1',                        //de la liste et du libellé correspondant.
        'separateur',                                //affiche une ligne de séparation
        'choix2' => 'Choix 2',
        'choix3' => 'Choix 3',
        'separateur',
        'choix4' => 'Choix 4'),
    'value' => array('choix3', ''),                    //valeur de la saisie
    'dbfield' => 'btrecherche',                       //retour de la saisie
    'inputType' => '',                                //search(default), text, time, date, etc.
    'decalage' => 'col-2',                            //décalage en colonnes bootstrap
    'label' => '',                                     //libellé du bouton (par défaut "champ") ou une icône loupe si
                                                    //vide ou icône font-awesome "<span class='fa fa-search'></span>"
    'lpos' => 'before',                               //position du champ de saisie par rapport au bouton
    'labelHelp' => 'aide sur le champ de recherche', //aide sur le champ
    'llong' => 'col-12',                               //col-12 dessine le bouton sur la totalité de sa largeur. Si
                                                    //vide, le bouton est dessiné à la largeur du libellé
    'lclass' => 'btn btn-success',                     //classe du bouton
    'clong' => 'col-4 offset-8',                       //longueur du bloc champ (ici cadré à droite)
    'cclass' => '',                                    //classe de la zone de saisie
    'maxlength' => 10,                                //taille maximum de la saisie en nombre de caractères
    'placeholder' => 'recherche',                     //placeholder de la saisie
    'spellcheck' => false,                             //correction orthographique ?
    'erreur' => true,                                  //montée erreur
    'liberreur' => 'TEST ERREUR',                     //libellé de l'erreur
    'liberreurHelp' => 'libelle erreur',              //Aide sur l'erreur
    'javascript' => $javascript,                      //code javascript associé
    'enable' => true,                                  //active, désactive le champ (dbfield renvoie NULL si false)
    'readonly' => false,                              //lecture seule (defield renvoi value si readonly)
    'invisible' => false                              //rend invisible le champ
));
```

Type de champ « filtretext »

Ce champ est composé d'une zone de saisie de texte et d'un menu d'options associées appelé **addon**. Contrairement à un champ **search** il n'y a pas de bouton de validation. Ce type de champ peut par exemple être utilisé pour créer des champ de type filtre. Il est d'ailleurs utilisé comme filtre dans les listes créées via la classe UniversalList. Le design d'un champ **filtretext** est toujours **online**. L'addon sera donc toujours positionné au-dessus ou au dessous du champ de saisie. Notons que l'addon est facultatif. Sans addon, ce type de champ se comporte comme une simple champ de type **text** avec – à la place de l'addon - un bouton inopérant qui fait office de label.

Voici un exemple de ce type de champ, sans menu déroulé et avec le menu déroulé.



Modèle html :

```
<bloc champ : clong invisible>
  <button : lclass/aclass lalign>
    label
  </button>
  <input : inputType enable cclass value javascript></input>
</bloc champ>
```

Propriété	Informations	Oblig.	Défaut
newLine	Booléen. Si true , l'élément doit être dessiné sur une nouvelle ligne du formulaire et deviendra de facto le premier champ de cette nouvelle ligne.	non	false
flexLine	Permet d'utiliser les possibilités flex de Bootstrap sur la totalité de la ligne du formulaire. Pour en savoir plus, lire la définition de cette propriété au chapitre 0.	non	" (vide)

Classe UniversalForm

dbfield	<p>Nom de la donnée affectée au champ. La donnée (c'est à dire le contenu du champ value) pourra ainsi être récupérée lors de l'appel à la méthode getData. Si cette propriété n'est pas renseignée, la valeur donnée au champ (donc saisie par l'utilisateur) ne pourra pas être restituée.</p> <p>Si l'objet contient une extension (addon = true), la valeur sera renvoyée sous forme d'un tableau à 2 entrées. L'entrée à la clé 0 correspond au choix dans la liste déroulante d'extension, l'entrée à la clé 1 correspond à la saisie faite par l'utilisateur dans le champ de saisie. Exemple :</p> <pre>Array ([0] => choix3 [1] => toto)</pre>	non	
inputType	<p>Choix parmi : button, checkbox, color, date, datetime, datetime-local, email, file, hidden, image, month, number, password, radio, range, reset, search, submit, tel, text, time, url, week. Voir la signification de chaque valeur au niveau HTML.</p> <p>Note : Les valeurs possibles de ce champ sont strictement les mêmes que pour un champ de type text. Cependant, s'agissant ici d'un champ filtersearch certaines valeurs comme file, submit ou autres sont tout à fait inappropriées... Qui peut le plus peut le moins ! La valeur par défaut text devrait correspondre 99.99% du temps !</p>	non	text
addon	<p>Booléen. Si true, l'objet va afficher une extension (petite liste déroulante d'informations supplémentaires). Il faut alors renseigner les propriétés apos, aclass, complement ainsi que value.</p>	non	false
apos	<p>(AddonPosition). Position de l'extension par rapport à la zone de saisie. Choisir parmi before ou after.</p> <p>Même si la propriété addon est à false, cette propriété est prise en compte pour positionner le label qui remplace l'addon au-dessus ou au-dessous du champ de saisie texte.</p>	non	before
aclass	<p>(AddonClasse). Classe CSS personnalisée de l'extension. S'agissant en réalité d'un bouton de type button il est possible de styler l'extension avec du CSS standard Bootstrap tel que 'btn btn-primary', 'btn btn-default' etc.</p> <p>A noter : la classe CSS est fournie au tag html button.</p>	non	" (vide)

Classe UniversalForm

complement	<p>Information complémentaire. Ici la propriété complement doit recevoir les informations qui vont permettre de construire la petite liste déroulante de l'extension. Il faut lui passer un tableau de couples valeur => libelle.</p> <p>Exemple :</p> <pre>array('choix1' => 'Choix 1', 'choix2' => 'Choix 2').</pre> <p>Note : Il est possible d'ajouter un séparateur entre deux options. Pour se faire, ajouter separateur ou separator dans le tableau.</p> <pre>array('choix1' => 'Choix 1', 'separateur', 'choix2' => 'Choix 2').</pre>	oui si addon est true	" (vide)
titre	Titre optionnel (libellé encore appelé légende).	non	" (vide)
tlong	<p>Longueur du titre dans le système de grille Bootstrap (ex : col-5 pour une largeur de 5 colonnes sur 12). Si cette propriété est omise, alors la longueur de la zone de titre s'étend sur la totalité de la ligne du formulaire (100%).</p> <p>On peut aussi passer à cette propriété un décalage. Ex col-2 offset-3 définit un Titre sur 2 colonnes décalé de 3 colonnes sur la droite.</p>	non	100% de la ligne du formulaire
tclass	Classe CSS personnalisée du titre.	non	" (vide)
talign	(TitreAlignement). Alignement du titre dans la zone de titre. Choisir parmi left , center , right , justify .	non	left
titreHelp	Chaine de caractère qui apparait au survol du titre par la souris.	non	" (vide)
titreHelpPos	Position du titreHelp par rapport au titre. Choisir parmi : auto , left , top , right , bottom . La valeur auto laisse l'application décider de la meilleure position en fonction des autres éléments alentour.	non	auto
decalage	<p>Décalage préliminaire vers la droite du dessin du champ sur la ligne par ajout d'un certain nombre de colonnes Bootstrap vides (ex : col-3 pour un décalage de 3 colonnes sur 12).</p> <p>Le même effet pourrait être obtenu en ajoutant un offset au champ (ex : offset-3) via la propriété clong (et non pas llong !) qui détermine la longueur de la zone de titre.</p> <p>Le décalage ne fait pas partie de la zone de champ.</p>	non	" (vide)

Classe UniversalForm

label	Libellé du bouton faisant office de label si la propriété addon est à false . Dans le cas contraire, cette propriété n'est pas prise en compte. Si la propriété n'est pas renseignée, alors la valeur par défaut s'applique et affiche champ . On peut aussi afficher le texte de son choix, y compris une icône du type font-awesome en passant le code suivant " " (qui correspond ici à la loupe).	non	champ
lclass	Classe CSS personnalisé du bouton faisant office de libellé si la propriété addon du champ est false . On pourra très bien utiliser les classe standard Bootstrap ' btn btn-primary ', ' btn btn-default ' etc. A noter : la classe CSS est fournie au tag html button .	non	" (vide)
lalign	(LibelleAlignement). Alignement du libellé (label) dans le bouton addon . Choisir parmi left , center , right , justify .	non	left
labelHelp	Chaine de caractère qui apparait au survol de l'objet par la souris.	non	" (vide)
labelHelpPos	Position du labelHelp par rapport au label. Choisir parmi : auto , left , top , right , bottom . La valeur auto par laisse l'application décider de la meilleure position en fonction des autres éléments alentour.	non	auto
clong	Longueur de l'objet filtreText dans le système de grille Bootstrap (ex : col-5 pour une longueur de champ de 5 colonnes sur 12). Si cette propriété est omise, alors la longueur de la zone de champ s'étend sur la totalité de la ligne du formulaire (100%).	non	100% de la ligne du formulaire
cclass	Classe CSS personnalisée du champ de saisie. A savoir : le code CSS n'est pas appliqué à la zone de champ mais au tag html input .	non	" (vide)
maxlength	Nombre maximum de caractères que l'on peut saisir dans le champ. La valeur par défaut 0 signifie « aucune limite ».	non	0
spellcheck	Booléen. Active ou désactive le correcteur orthographique du navigateur pour le champ (les éventuelles fautes d'orthographe sont soulignées en rouge). Par défaut à true , ce qui signifie que cette option dépend du paramétrage du navigateur .	non	true
placeholder	Texte qui apparaît par défaut dans le champ.	non	" (vide)

Classe UniversalForm

value	<p>Contenu (valeur) du champ de saisie.</p> <p>Important : ne pas donner de booléen true/false - ou alors sous forme d'une chaîne de caractères : 'true'/'false'.</p> <p>1 - Si l'objet n'affiche pas d'extension (propriété addon = false), alors la valeur du champ est une valeur simple.</p> <p>2 - Si l'objet affiche une extension (propriété addon = true), alors la valeur retournée (et celle saisie) doit être un tableau à 2 entrées contenant la valeur de l'extension, et la valeur du champ de recherche. Exemple : array('choix2', ''). Sur cet exemple, le libellé de l'option du menu déroulant d'extension dont la valeur est choix2 sera affiché sur l'extension ; la zone de saisie sera vide.</p> <p>NB : dans le choix d'une extension, le contenu de dbfield renvoie lui aussi un tableau.</p>	oui si addon est true	" (vide)
erreur	Booléen. Indique si une erreur est à produire sur le champ.	non	false
liberreur	Libellé de l'erreur à lever.	non	" (vide)
liberreurHelp	Libellé à afficher sur le texte d'erreur au survol de la souris. Sert par exemple à donner des explications supplémentaire sur une erreur.	non	" (vide)
javascript	Code javascript à associer au champ (input)	non	" (vide)
enable	<p>Booléen. Active ou désactive le champ. Dans ce dernier cas, le champ reste visible, grisé et ne peut plus recevoir de saisie. Le curseur de la souris se change en panneau sens interdit au survol du champ désactivé.</p> <p>NB : il n'est effectué aucun test sur les champs désactivés.</p> <p>NB : un champ désactivé renvoie NULL à l'appel de la méthode getData.</p>	non	true
readonly	<p>Booléen. Positionne le champ en lecture seule. En lecture seule on ne peut pas modifier le contenu du champ. Il est alors grisé. Un petit '1' est ajouté au label correspondant pour confirmer cet état.</p> <p>NB : Même en lecture seule, la valeur du champ (son contenu) est renvoyée à l'appel de la méthode getData.</p>	non	false
Invisible	<p>Booléen.</p> <p>Si true, la zone de champ est rendue invisible.</p> <p>NB : Le contenu du champ (même invisible) est toutefois renvoyé par la méthode getData.</p>	non	false

Exemple de construction

```
//ajout d'un champ de recherche spécialisé pour les entêtes de listings
$this->createField('filtretext', 'filtrel', array(
    'newLine' => true,                //nouvelle ligne ? False par défaut
    'addon' => true,                  //ajout d'une liste déroulante addon
    'aclass' => 'btn btn-success',    //classe CSS de l'addon
    'apos' => 'before',               //position de l'addon par rapport à la saisie
));
```


Classe UniversalForm

```
'complement' => array(                                //contenu de l'addon (liste déroulante composée de la valeur
'tout' => 'Tout',                                    //contenu de l'addon (liste déroulante composée de la valeur
'commence' => 'Commence par',                          //de la liste et du libellé correspondant.
'separateur',                                         //affiche une ligne de séparation
'egal' => 'Egal à',
'contient' => 'Ressemble à',
'separator',                                         //autre séparateur (nom différent marche aussi)
'finit' => 'Finit par'),
'titre' => 'Filtre textuel',
'tlong' => 'col-2 offset-6',
'tclass' => 'vert adroite',                            //classe du titre
'titreHelp' => 'Aide sur le titre',                   //Aide sur le titre
'label' => 'Juste le label',                           //label
'lclass' => 'btn btn-success',                         //classe CSS du bouton
'lalign' => 'right',                                   //alignement du libellé dans le bouton
'value' => '',                                         //valeur de la saisie dans le cas où addon = false
'value' => array('egal', 'test'),                     //valeur de la saisie (tableau) dans le cas ou addon = true
'dbfield' => 'filtrel',                               //retour de la saisie
'inputType' => '',                                    //search (default), text, time, date, etc.
'decalage' => 'col-2',                                //décalage en colonnes bootstrap
'labelHelp' => 'Aide sur le filtre',                   //aide sur le champ
'clong' => 'col-4',                                   //longueur du bloc champ (ici cadré à droite)
'cclass' => '',                                       //classe de la zone de saisie
'maxlength' => 10,                                   //taille maximum de la saisie en nombre de caractèresz
'placeholder' => 'filtre',                            //placeholder de la saisie
'spellcheck' => false,                                //correction orthographique ?
'erreur' => true,                                     //montée erreur
'liberreur' => 'TEST ERREUR',                          //libellé de l'erreur
'liberreurHelp' => 'libelle erreur',                  //Aide sur l'erreur
'javascript' => $javascript,                          //code javascript associé
'enable' => true,                                     //active, désactive le champ (dbfield renvoie NULL si false)
'readonly' => false,                                  //lecture seule (defield renvoi value si readonly)
'invisible' => false                                  //rend invisible le champ
));
```

Type de champ « filtreset »

Ce champ est très proche du champ de type `select`. C'est d'ailleurs un champ `select` agrémenté d'un bouton type « addon » inopérant qui fait office de label afin d'obtenir le même style graphique qu'un champ de type `filtertext`. Il peut donc être conjointement utilisé avec ce type de champ pour créer des champs de type filtre. Lui aussi est d'ailleurs utilisé par la classe `UniversalList` comme filtre.

Voici deux exemples de restitution d'un champ `filterselect`, le premier une liste avec un choix multiple, le deuxième, une liste déroulante.



Il n'y a pas de notion de groupe sur les listes déroulantes, donc pas de titre (ou légende) non plus.

Modèle html « inline » :

```
<bloc champ : clong invisible>
  <button zone titre : lclass lalign>
    label
  </button>
  <select : enable cclass javascript>complement value</select>
</bloc champ>
```

Modèle html « online » :

```
<bloc champ : clong invisible>
  <button zone titre : lclass lalign>
    label
  </button>
  <zone champ>
    <select : enable cclass javascript>complement value</select>
  </zone champ>
</bloc champ>
```

Propriété	Informations	Oblig.	Défaut
newLine	Booléen. Si <code>true</code> , l'élément doit être dessiné sur une nouvelle ligne du formulaire et deviendra de facto le premier champ de cette nouvelle ligne.	non	<code>false</code>
flexLine	Permet d'utiliser les possibilités <code>flex</code> de Bootstrap sur la totalité de la ligne du formulaire. Pour en savoir plus, lire la définition de cette propriété au chapitre 0.	non	" (vide)

Classe UniversalForm

dbfield	<p>Nom de la donnée affectée à la liste déroulante. La donnée (c'est à dire l'entrée de liste sélectionnée) pourra ainsi être récupérée lors de l'appel à la méthode getData dans la propriété value. Si cette propriété n'est pas renseignée, la sélection de l'utilisateur ne pourra pas être restituée.</p> <p>Si la propriété multiple indiquant une sélection multiple possible est positionnée à true, alors la valeur sera renvoyée sous forme d'un tableau possédant autant d'entrées que de d'items sélectionnés.</p>	non	
design	<p>Design du champ.</p> <p>inline : l'addon est positionné avant ou après la liste déroulante. online : l'addon est positionné au-dessus ou au-dessous de la liste déroulante.</p>	non	inline
multiple	<p>Booléen.</p> <p>Permet une sélection multiple. En cas de sélection multiple le POST renvoie alors un tableau de valeurs (array)</p>	non	false
size	<p>Entier. Si cette propriété est positionnée et > 1 alors le sélecteur n'est plus une liste déroulante mais une liste simple d'une hauteur de size lignes.</p> <p>NB : En cas de sélection multiple, la valeur par défaut est de 4 lignes, la valeur 1 est impossible et la valeur 2 est la valeur minimum.</p>	non	4
decalage	<p>Décalage préliminaire vers la droite de la liste (et de son libellé) sur la ligne par ajout d'un certain nombre de colonnes Bootstrap vides (ex : col-3 pour un décalage de 3 colonnes sur 12).</p> <p>Astuce : le même effet pourrait être obtenu en ajoutant un offset à la propriété llong (longueur de la zone de titre) ou à la propriété clong (longueur de la zone de champ) (ex : offset-3).</p> <p>Le décalage ne fait pas partie de la zone de champ.</p>	non	" (vide)
label	Libellé de la zone de titre.	non	champ
lclass	<p>Classe CSS personnalisée du libellé fourni dans label.</p> <p>A savoir : Le code CSS s'applique à la button.</p>	non	" (vide)
lalign	(LibelleAlignement). Alignement du libellé (label) du bouton. Choisir parmi left , center , right , justify .	non	left
labelHelp	Chaine de caractère qui apparait au survol du label par la souris.	non	" (vide)
labelHelpPos	Position du labelHelp par rapport au label. Choisir parmi : auto , left , top , right , bottom . La valeur auto laisse l'application décider de la meilleure position en fonction des autres éléments alentour.	non	auto

Classe UniversalForm

lpos	<p>Position de la zone de titre par rapport à la zone de champ.</p> <p>before : dans le cas d'un design inline, la zone de titre est affichée devant la zone de champ. Dans le cas d'un design online, la zone de titre est affichée au-dessus de la zone de champ.</p> <p>after : dans le cas d'un design inline, la zone de titre est affichée après la zone de champ. Dans le cas d'un design online, la zone de titre est affichée au-dessous de la zone de champ.</p>	non	before
clong	<p>Longueur du bloc de champ dans le système de grille Bootstrap. (ex : <code>col-5</code> pour une longueur de champ de 5 colonnes sur 12).</p> <p>Si cette propriété est omise, alors la longueur de la zone de champ s'étend sur la totalité de la ligne du formulaire (100%).</p>	non	100% de la ligne du formulaire
cclass	<p>Classe CSS personnalisée de la liste déroulante</p> <p>A savoir : le code CSS n'est pas appliqué à la zone de champ mais au tag html select.</p>	non	" (vide)
testMatches	<p>Conditions de test (voir paragraphe détaillé sur les tests réalisables sur les champs).</p> <p>Attention : Aucun test n'est effectué sur une liste à choix multiples (propriété multiple à true)</p>	non	null
value	<p>Valeur (option) sélectionnée dans la liste. C'est une des valeurs de la liste qui doit être créée par la fonction de callback déclarée dans la propriété complement et à écrire par le développeur. Cette valeur peut être une chaîne, un entier, etc.</p> <p>Dans le cas d'un sélecteur multiple (liste à choix multiple), cette propriété doit recevoir un tableau de valeurs.</p> <p>NB : ne pas donner de booléen true/false - ou alors sous forme de chaîne de caractère : 'true' / 'false'.</p> <p>NB : dans le cas d'un sélecteur simple, si aucune valeur n'est donnée (propriété non renseignée), alors la liste renverra la dernière option (valeur de la liste) disponible.</p> <p>NB : si la valeur donnée n'existe pas dans la liste (aucun mot clé selected présent), alors la liste renverra la première option (valeur de la liste).</p> <p>NB : dans le cas d'un sélecteur multiple, il est obligatoire de renseigner cette propriété en lui donnant un tableau de valeurs à sélectionner. Dans le cas contraire, PHP lèvera une erreur. Un tableau vide array() aura pour conséquence de ne sélectionner aucune option. De même, à la validation, si aucune option n'est sélectionnée, le champ dbfield renverra NULL. Si une valeur du tableau n'existe pas, elle est ignorée.</p>	non	0
complement	<p>Information complémentaire.</p> <p>Cette propriété reçoit le nom de la fonction callback qui est chargée de remplir la liste. Si la propriété est laissée vide, une erreur est levée.</p>	oui	" (vide)

Classe UniversalForm

erreur	Booléen. Indique si une erreur est à produire sur le champ.	non	False
liberreur	Libellé de l'erreur à lever.	non	" (vide)
liberreurHelp	Libellé à afficher sur le texte d'erreur au survol de la souris. Sert par exemple à donner des explications supplémentaire sur une erreur.	non	" (vide)
javascript	Code javascript à associer au champ (positionné sur le tag HTML select)	non	" (vide)
enable	<p>Booléen. Active ou désactive le champ. Dans ce dernier cas, le champ reste visible, grisé et ne peut plus recevoir de saisie. Le curseur de la souris se change en panneau sens interdit au survol du champ désactivé.</p> <p>NB : il n'est effectué aucun test sur les champs désactivés.</p> <p>NB : un champ désactivé renvoie NULL à l'appel de la méthode getData.</p>	non	true
readonly	<p>Booléen. Positionne le champ est en lecture seule. En lecture seule on ne peut pas modifier le contenu du champ. Il est alors grisé. Un petit '1' est ajouté au label correspondant pour confirmer cet état.</p> <p>NB : même en lecture seule, la valeur du champ (son contenu) est renvoyée à l'appel de la méthode getData.</p>	non	false
Invisible	<p>Booléen. Si true, zone de titre et zone de champ sont rendues invisibles.</p> <p>NB : Le contenu du champ (même invisible) est toutefois renvoyé par la méthode getData.</p>	non	false

Exemple de construction

```
function fillSelect($value) {
    $html = '';
    ($value == '') ? $default = ' selected' : $default = '';
    $html.= '<option value="Indéfini"'.$default.'>Choose a genre</option>';
    ($value == 'action') ? $default = ' selected' : $default = '';
    $html.= '<option value="action"'.$default.'>Action</option>';
    ($value == 'comedy') ? $default = ' selected' : $default = '';
    $html.= '<option value="comedy"'.$default.'>Comedy</option>';
    ($value == 'horror') ? $default = ' selected' : $default = '';
    $html.= '<option value="horror"'.$default.'>Horror</option>';
    ($value == 'romance') ? $default = ' selected' : $default = '';
    $html.= '<option value="romance"'.$default.'>Romance</option>';
    return $html;
}

//ajout d'un champ de recherche spécialisé pour les entêtes de listings
$this->createField('filtreselect', 'filtre2', array(
    'newLine' => true, //nouvelle ligne ? false par défaut
    'dbfield' => 'filtre2', //retour de la saisie
    'design' => 'online', //inline (défaut) / online
    'multiple' => true, //sélection multiple autorisée ? (false par défaut)
    'size' => 4, //hauteur du select en nombre de lignes visibles (1 par défaut)
    'decalage' => 'col-2', //décalage en colonnes bootstrap
    'label' => 'Filtre Genre', //label
    'lclass' => 'btn btn-success', //classe du label
    'lalign' => 'right', //left (default) / right / center / justify
    'labelHelp' => 'Aide sur filtre2', //aide sur le label
    'lpos' => 'after', //position label par rapport au champ : before (défaut) / after
    'clong' => 'col-3', //longueur de la zone de champ
    'cclass' => '', //classe de la zone de champ
));
```

Classe UniversalForm

```
'testMatches' => array('REQUIRED_SELECTION'), //test de la saisie
'value' => $this->_tab_donnees['filtre2'], //valeur de la saisie
'complement' => 'fillSelect', //fonction de callback qui doit remplir le select
'erreur' => true, //montée erreur
'liberreur' => 'TEST ERREUR', //libellé de l'erreur
'liberreurHelp' => 'libelle erreur', //Aide sur l'erreur
'javascript' => '', //code javascript associé
'enable' => true, //active, désactive le champ (dbfield renvoie NULL si false)
'readonly' => false, //lecture seule (defield renvoi value si readonly)
'invisible' => false //rend invisible le champ
));
```

Type de champ « séparateur »

Zone de séparation de champs. Il s'agit principalement d'un type de champ destiné à afficher du texte pour affiner le look du formulaire (comme un paragraphe). Ce type de champ permet cependant de récupérer une valeur passée dans la propriété **value**. Un séparateur est toujours **online**.

Modèle html :

```
<bloc champ : clong cclass border invisible>
  <zone titre : lclass>
    <p>label</p>
  </zone titre>
  <zone champ>
  </zone champ>
</bloc champ>
```

Propriété	Informations	Oblig.	Défaut
newLine	Booléen. Si true , l'élément doit être dessiné sur une nouvelle ligne du formulaire et deviendra de facto le premier champ de cette nouvelle ligne.	non	false
flexLine	Permet d'utiliser les possibilités flex de Bootstrap sur la totalité de la ligne du formulaire. Pour en savoir plus, lire la définition de cette propriété au chapitre 0.	non	" (vide)
dbfield	Nom de la donnée affectée au champ. La donnée (c'est à dire le contenu du champ value) pourra ainsi être récupérée lors de l'appel à la méthode getData . Si cette propriété n'est pas renseignée, la valeur donnée au champ ne pourra pas être restituée.	non	
decalage	Décalage préliminaire vers la droite du dessin du champ sur la ligne par ajout d'un certain nombre de colonnes Bootstrap vides (ex : col-3 pour un décalage de 3 colonnes sur 12). Le même effet pourrait être obtenu en ajoutant un offset au champ (ex : offset-3) à la propriété clong . Le décalage ne fait pas partie de la zone de champ.	non	" (vide)
label	Label du séparateur. Si on ne veut pas afficher de label (ce qui est idiot), il faut renseigner ce champ avec la valeur " (vide), sinon la valeur par défaut sera affichée.	Non	champ
lclass	Classe CSS personnalisée du libellé fourni dans label . A savoir : Le code CSS s'applique à la « zone de titre ».	non	" (vide)
labelHelp	Chaine de caractère qui apparait au survol de l'objet par la souris.	non	" (vide)
labelHelpPos	Position du labelHelp par rapport au label. Choisir parmi : auto , left , top , right , bottom . La valeur auto laisse l'application décider de la meilleure position en fonction des autres éléments alentour.	non	auto

Classe UniversalForm

border	<p>Booléen ou code CSS. Bordure affichée autour du champ. Le décalage éventuel n'est pas concerné. Plusieurs entrées possible :</p> <p>false : aucune bordure n'est dessinée autour du champ. true : le champ est entouré d'une bordure standard. CSS : code CSS personnalisé pour la bordure. Exemple : <code>'border'</code> => <code>'border-bottom:2px dotted silver;'</code></p>	non	false
clong	<p>Longueur de de l'objet dans le système de grille Bootstrap (ex : <code>col-5</code> pour une longueur de champ de 5 colonnes sur 12).</p> <p>Si cette propriété est omise, alors la longueur de la zone de champ s'étend sur la totalité de la ligne du formulaire (100%), ce qui peut être intéressant pour ce type de champ.</p>	non	100% de la ligne du formulaire
cclass	Classe CSS personnalisée du « bloc de champ ».	non	" (vide)
value	<p>Valeur donnée au champ qui pourra être récupérée par <code>dbfield</code>.</p> <p>Important : ne pas donner de booléen <code>true/false</code> - ou alors sous forme d'une chaîne de caractères : <code>'true'/'false'</code>.</p>	non	" (vide)
Invisible	<p>Booléen. Si <code>true</code>, le bloc champ est rendu invisible.</p> <p>NB : La valeur du champ (même invisible) est toutefois renvoyé par la méthode <code>getData</code>.</p>	non	false

Exemple de construction

```
$this->createField('separateur', 'sep12', array(
    'newLine' => true,                //nouvelle ligne ? False par défaut
    'dbfield' => 'separateur10',      //retour de la saisie
    'decalage' => 'col-2',            //décalage en colonnes bootstrap
    'label' => 'FILTRE',              //libellé du séparateur
    'lclass' => 'gras bleu souligne_epais', //classe du séparateur
    'labelHelp' => 'Ceci est une aide sur le label', //aide sur le séparateur
    'border' => false,                //défaut : false. false / true (encadrement par défaut)
    'border' => 'border-top:2px dotted silver;', //... ou bordure personnalisée
    'clong' => 'col-5',               //longueur du séparateur en colonnes bootstrap
    'cclass' => '',                   //classe sur le bloc de champ
    'value' => 'Séparateur 10',       //valeur de la saisie
    'invisible' => false              //rend invisible le champ
));
```


Type de champ « comment »

Champ commentaires. Aucune saisie, il permet d'afficher un commentaire (paragraphe ou texte) qui pourra être récupéré via la méthode `getData()`. Le champ `comment` est très proche du type champ `image`.

Modèle html « inline » :

```
<zone titre : llong lclass lalign invisible>
  <label>label</label>
</zone titre>
<zone champ : clong invisible>
  <div : cclass border>value</div>
</zone champ>
```

Modèle html « online » :

```
<bloc champ : clong invisible>
  <zone titre : lclass lalign>
    <label>label</label>
  </zone titre>
  <zone champ>
    <div : cclass border>value</div>
  </zone champ>
</bloc champ>
```

Propriété	Informations	Oblig.	Défaut
newLine	Booléen. Si <code>true</code> , l'élément doit être dessiné sur une nouvelle ligne du formulaire et deviendra de facto le premier champ de cette nouvelle ligne.	non	<code>false</code>
flexLine	Permet d'utiliser les possibilités <code>flex</code> de Bootstrap sur la totalité de la ligne du formulaire. Pour en savoir plus, lire la définition de cette propriété au chapitre 0.	non	" (vide)
dbfield	Nom de la donnée affectée au champ. La donnée (c'est-à-dire le contenu du champ <code>value</code>) pourra ainsi être récupérée lors de l'appel à la méthode <code>getData</code> . Si cette propriété n'est pas renseignée, le contenu de valeur ne pourra pas être récupéré.	non	
design	Design du champ. inline : la zone de titre est positionnée avant ou après la zone de champ. online : la zone de titre est positionnée au-dessus ou au-dessous de la zone de champ.	non	<code>inline</code>
decalage	Décalage vers la droite du bloc de champ par ajout d'un certain nombre de colonnes Bootstrap vides (ex : <code>col-3</code> pour un décalage de 3 colonnes sur 12). Astuce : le même effet pourrait être obtenu en ajoutant un <code>offset</code> à la propriété <code>llong</code> (longueur de la zone de titre) dans le cas d'un design <code>inline</code> .	non	" (vide)

Classe UniversalForm

label	Label de la zone de titre, c'est à dire le commentaire. Si on ne veut pas afficher de label, il faut obligatoirement renseigner ce champ avec la valeur " (vide).	non	champ
lpos	Position de la zone de titre par rapport à la zone de champ. before : dans le cas d'un design inline , la zone de titre est affichée devant la zone de champ. Dans le cas d'un design online , la zone de titre est affichée au-dessus de la zone de champ. after : dans le cas d'un design inline , la zone de titre est affichée après la zone de champ. Dans le cas d'un design online , la zone de titre est affichée au-dessous de la zone de champ.	non	before
llong	Longueur de la zone de titre dans le système de grille Bootstrap (ex : col-5 pour une largeur de 5 colonnes sur 12). Si cette propriété est omise, alors la longueur de la zone de titre s'étend sur la totalité de la ligne du formulaire (100%). Cette propriété est seulement prise en compte dans le cas d'un design inline .	non	100% de la ligne du formulaire
lclass	Classe CSS personnalisée du libellé fourni dans label . A savoir : Le code CSS s'applique à la « zone de titre ».	non	" (vide)
lalign	Alignement du libellé (label) dans la zone de titre. Choisir parmi left , center , right , justify .	non	left
labelHelp	Chaine de caractère qui apparait au survol du label par la souris.	non	" (vide)
labelHelpPos	Position du labelHelp par rapport au label. Choisir parmi : auto , left , top , right , bottom . La valeur auto laisse l'application décider de la meilleure position en fonction des autres éléments alentour.	non	auto
clong	Longueur de la zone de champ dans le système de grille Bootstrap. (ex : col-5 pour une longueur de champ de 5 colonnes sur 12). Si cette propriété est omise, alors la longueur de la zone de champ s'étend sur la totalité de la ligne du formulaire (100%).	non	100% de la ligne du formulaire
cclass	Classe CSS personnalisée de l'objet. A savoir : le code CSS est appliqué à la balise html <div> et non à la zone de champ.	non	" (vide)

Classe UniversalForm

border	<p>Booléen ou code CSS. Bordure affichée autour du commentaire. Le décalage éventuel n'est pas concerné. Plusieurs entrées possible :</p> <p>false : aucune bordure n'est dessinée autour du commentaire. true : le commentaire est entouré d'une bordure standard. CSS : code CSS personnalisé pour la bordure. Exemple : 'border' => 'border-bottom:2px dotted silver;'</p>	non	false
value	Commentaire (texte) à afficher.	non	" (vide)
erreur	Booléen. Indique si une erreur est à produire sur le champ.	non	false
liberreur	Libellé de l'erreur à lever.	non	" (vide)
liberreurHelp	Libellé à afficher sur le texte d'erreur au survol de la souris. Sert par exemple à donner des explications supplémentaire sur une erreur.	non	" (vide)
invisible	<p>Booléen. Si true, zone de titre et zone de champ sont rendues invisibles.</p> <p>NB : Le commentaire (même invisible) est toutefois renvoyé par la méthode getData.</p>	non	false

Exemple de construction

```
$this->createField('area', 'commentaires-inline', array(
    'newLine' => true,                //nouvelle ligne ? False par défaut
    'dbfield' => 'comments',          //retour de la saisie
    'design' => 'inline',              //inline (default) / online
    //'decalage' => 'col-2',           //décalage en colonnes bootstrap
    'label' => 'Commentaires',        //label
    'llong' => 'col-2',               //long. zone de titre (inutile dans le cas d'un design online)
    'lclass' => 'bleu',               //classe du label
    'lalign' => 'right',              //left (défaut) / right / center / justify
    'labelHelp' => 'Aide sur commentaires', //aide sur le label
    'lpos' => 'before',               //position label par rapport au champ : before (défaut) / after
    'clong' => 'col-5',               //longueur de la zone de champ
    'rows' => 7,                      //hauteur du commentaire en nombre de lignes
    'cclass' => '',                   //classe de la zone de champ
    'maxlength' => 128,               //nb caractères max en saisie
    'spellcheck' => true,             //correction orthographique
    'placeholder' => 'Commentaire inline', //texte pré-affiché
    'testMatches' => array('REQUIRED'), //test de la saisie
    'value' => $this->_tab_donnees['commentaires'], //valeur de la saisie
    'erreur' => true,                 //montée erreur
    'liberreur' => 'TEST ERREUR',     //libellé de l'erreur
    'liberreurHelp' => 'libelle erreur', //Aide sur l'erreur
    'javascript' => '',               //code javascript associé
    'enable' => true,                 //active, désactive le champ (dbfield renvoie NULL si false)
    'readonly' => false,              //lecture seule (dbfield renvoie value si readonly)
    'invisible' => false              //rend invisible le champ
));
```

Type de champ « div »

Panel. Permet de rassembler d'autres champs sous une même `<div>`.

Astuce : Permet donc de cacher / afficher plusieurs champs UniversalForm en même temps

Attention : L'appel d'un champ `div` doit impérativement être fermé par l'appel d'un champ `divfin`.

NB : Un champ `div` n'est constitué que d'un « bloc champ ».

Modèle html :

```
<bloc champ : cclass invisible>
<zone titre></zone titre>
  <zone champ></zone champ>
</bloc champ>
```

Propriété	Informations	Oblig.	Défaut
newLine	Booléen. Si true , l'élément doit être dessiné sur une nouvelle ligne du formulaire et deviendra de facto le premier champ de cette nouvelle ligne.	non	false
cclass	Classe CSS personnalisée de la <code>div</code> (panel).	non	" (vide)
Invisible	Booléen. Rend le panel invisible.	non	false

Type de champ « divfin »

Ferme le panel. Obligatoire après l'ouverture d'un panel (champ `div`). Ce champ ne possède aucune propriété.

Type de champ « hidden »

Champ caché.

Modèle html :

```
<input type=hidden></input>
```

Propriété	Informations	Oblig.	Défaut
dbfield	Nom de la donnée affectée au champ caché. La donnée pourra ainsi être récupérée lors de l'appel à la méthode <code>getData</code> . Si cette propriété n'est pas renseignée, le champ caché ne pourra pas être récupéré. Permet de conserver des valeurs le temps de la gestion du formulaire.	non	
value	Valeur du champ caché. C'est cette valeur qui sera renvoyée par la méthode <code>getData</code> si la propriété <code>dbfield</code> a été spécifiée.	non	" (vide)

7. Test du formulaire

Nous voici maintenant en possession d'un formulaire complet qui permet de récupérer toute sorte de champ de saisie. Nous allons maintenant voir comment appliquer des tests sur les champs afin de valider le formulaire.

Tests unitaires (champ par champ)

Lors de la création d'un champ il est possible de spécifier certaines contraintes de saisie. Ceci va se faire via la propriété `testMatches`. Si vous l'avez remarqué dans notre exemple, le champ « titre » était construit ainsi :

```
$this->createField('text', 'titre', array(
    'newLine' => true,
    'dbfield' => 'titre',
    'label' => 'Titre du film',
    'llong' => 'col-2',
    'clong' => 'col-5',
    'testMatches' => array('REQUIRED'),
    'value' => $this->_tab_donnees['titre'],
));
```

Cette propriété doit être renseignée sous forme de tableau de tests auquel le champ doit satisfaire pour que le formulaire accepte sa saisie. Ici, dans cet exemple, cela signifie que la saisie du champ `titre` est obligatoire. Voici tous les tests prédéfinis à votre disposition :

Test	Informations	S'applique à
REQUIRED	Le champ ne doit pas être vide. Correspond à une saisie obligatoire.	text / checkbox ¹
NUMERIC	Le contenu du champ doit être du type numérique (ou vide). Les virgules sont donc acceptées.	text / area
REQUIRED_SELECTION	Le champ ne doit pas être vide et ne pas contenir la chaîne de caractère 'NULL' (et non pas NULL). Correspond à une saisie obligatoire.	text / area / select
NOT_ZERO	La saisie doit être différente de 0 (ou vide).	text / area
CHECK_INTEGER	Entier obligatoire (signes + et - autorisés) (ou vide).	text / area
CHECK_INTEGER_1OU2	Entier à 1 ou 2 chiffres obligatoire (ou vide).	text / area
CHECK_INTEGER_4	Année : 4 chiffres obligatoires (ou vide).	text / area
CHECK_INTEGER_8	8 chiffres obligatoires (ou vide).	text / area
CHECK_UNSIGNED_INTEGER	Entier obligatoire (signe interdit) (ou vide).	text / area

¹ Si la valeur de la case à cocher est vide.

Classe UniversalForm

CHECK_SIGNED_INTEGER	Entier obligatoire (signes + et - autorisés) (ou vide).	text / area
CHECK_FLOAT	Nombre à virgule flottante avec décimales optionnelles (ou vide).	text / area
CHECK_FLOAT_2DEC	Nombre à virgule flottante avec 2 décimales optionnelles (ou vide).	text / area
CHECK_BOOLEAN	0 ou 1 (ou vide).	text / area / checkbox
CHECK_DATETIME	Entrée Datetime au format 0000-00-00 00:00 (Y-m-d) (les secondes sont optionnelles) (ou vide).	text / area
CHECK_EMAIL	eMail (ou vide).	text / area
CHECK_EMAIL_APOSTROPHE	eMail avec prise en compte apostrophe (ou vide).	text / area
CHECK_ALPHA_CODE	majuscule + minuscules + "-" + "_" + "." ou vide	text / area
CHECK_ALPHA_SIMPLE	chiffres + minuscules + espace + "-" (ou vide).	text / area
CHECK_ALPHA_NOMS	majuscules + minuscules + accents + espace (ou vide).	text / area
CHECK_URL	url du style <code>http://serveur.ext/page/</code> (ou vide).	text / area
CHECK_IPV4	Adresse IP V4 (ex : 192.168.1.0)	text / area
CHECK_MAC	Adresse Mac (ex : 12:45:af:20:d8:00)	text / area
CHECK_SHA1	40 caractères hexadécimaux	text / area
CHECK_DOLLARS	Saisie de la forme \$9 999 999 (signe monétaire \$ suivi de chiffres séparés par un espace entre chaque millier). A utiliser en conjonction avec le transformateur USD	text / area
CHECK_MONNAIE	Saisie de la forme [signe]9 999 999 (signe monétaire \$ ou £ ou € suivi de chiffres séparés par un espace entre chaque millier). A utiliser en conjonction avec les transformateurs USD, EUR et GBP	text / area
CHECK_INTEGER_SPACED	Saisie d'un nombre dans lequel sont autorisés les espaces. A utiliser en conjonction avec le transformateur MILLE_SPACED	text / area

Classe UniversalForm

CHECK_FILE_NAME	Vérifie si saisie compatible avec le nommage d'un fichier	text / area
------------------------	---	-------------

Si l'un des test spécifié n'existe pas, une erreur est levée sur le champ concerné afin de prévenir le développeur (libellé et encadrement du champ en rouge).

- Les tests sont effectués sur la valeur de la propriété `value` (récupérée de la saisie).
- Aucun test n'est effectué sur les champs qui sont désactivés (`enable`), en lecture seule (`readonly`) ou invisible (`invisible`).

En plus de ces tests, on peut se servir de cette propriété pour transformer automatiquement une saisie via l'un des transformateurs suivants :

NB : Notons que la transformation du champ intervient juste avant les tests unitaires.

Transformateur	Informations	S'applique à
UPPERCASE	Transforme la saisie en majuscules	text / area
LOWERCASE	Transforme la saisie en minuscules	text / area
WORDSCASE	Mise en majuscule des premières lettres de chaque mot	text / area
FIRST-LETTER	Mise en majuscule de la première lettre de chaque mot de la phrase	text / area
FIRST-LETTER-ONLY	Mise en majuscule de la première lettre de la phrase	text / area
TRIM	Supprime les espaces avant et après le texte saisi	text / area
NOSPACE	Supprime tous les espaces du texte saisi	text / area
NODOUBLESPEACE	Supprime tous les espaces doublés (ou plus). Permet de ramener une suite d'espace à 1 seul espace, y compris à l'intérieur de la chaîne saisie.	text / area
USD	Transforme un nombre sous la forme \$9 999 999	text / area
EUR	Transforme un nombre sous la forme €9 999 999	text / area
GBP	Transforme un nombre sous la forme £9 999 999	text / area
MILLE_SPACED	Transforme un nombre sous la forme 9 999 999 (ajout d'un espace entre chaque millier)	text / area

Information : cette transformation ne fonctionne pas pour les listes à choix multiples (`select` avec propriété `multiple` à `true`), ni pour les champs `invisible` ou `readonly`.

NOTES :

- Les transformateurs **USD**, **EUR**, **GBP** et **MILLE_SPACED**, ne sont appliqués que si le texte saisi est du type numérique. Dans le cas contraire, la transformation n'est pas opérée.

Classe UniversalForm

- Pour tester la saisie sur ce type de champ, utiliser les tests unitaires `CHECK_DOLLARS`, `CHECK_MONNAIE` et `CHECK_INTEGER_SPACED`.
- Noter de plus que bien que la saisie de ces nombres soit ainsi transformée en chaîne de caractères, la méthode `getData` renvoie uniquement la valeur numérique sans espace et sans signe monétaire.

Ainsi, le champ année pourrait être amélioré afin de le rendre 4 chiffres obligatoires :

```
$this->createField('text', 'annee', array(
    'newLine' => true,
    'dbfield' => 'annee',
    'label' => 'Année de production',
    'llong' => 'col-2',
    'clong' => 'col-1',
    'value' => $this->_tab_donnees['annee'],
    'testMatches' => array('REQUIRED', 'CHECK_INTEGER_4')
));
```

Pour que le test soit exécuté (jusque-là nous avons juste paramétré les champs), il va maintenant falloir faire appel à la méthode `tester()` de notre formulaire. Pour prendre en compte ces tests, nous allons modifier notre code dans le gestionnaire du formulaire, ici le fichier `saisie_videotheque.php`.

```
if ($action == 'ajouter') {
    $monFormulaire->init();
    echo $monFormulaire->afficher();
}
elseif ($action == 'valid_ajouter') {
    if (!$monFormulaire->tester()) {
        echo $monFormulaire->afficher();
    }
    else {
        $lesDonnees = $monFormulaire->getData();
        echo '<pre>';
        print_r($lesDonnees);
        echo '</pre>';
    }
}
```

Explications : A la validation du formulaire (la variable `$action` contient la valeur `valid_ajouter`), nous testons le formulaire à l'aide de la méthode `tester`. Celle-ci renvoie un booléen :

true : le formulaire est valide (l'ensemble des tests du formulaire sont validés).

false : au moins un champ ne répond pas à une contrainte choisie. Dans ce cas, on réaffiche le formulaire par l'appel de la méthode `afficher`.

Au réaffichage du formulaire, un libellé rouge vous rappelle les erreurs commises.

Attention, les tests unitaires sont exécutés sur les champs les uns à la suite des autres. Si un champ est en erreur, seul ce champ sera signalé. S'il existe des erreurs sur un champ suivant, celle-ci sera signalée seulement lorsque le premier champ aura été corrigé.

Tests supplémentaires

En plus de la batterie de contraintes standard qu'il est possible de donner à un champ pour le formaliser à sa création, il est possible de créer des tests personnalisés. Par exemple, si l'on souhaite que l'année saisie soit comprise entre 1910 et 2030, nous allons appeler et surcharger la méthode `testsSupplementaires($champ)`. Voici à quoi peut ressembler le code :

```
protected function testsSupplementaires($champ) {
    if ($champ->idField() == 'annee') {
        if (($champ->value() < 1920) || ($champ->value() > 2030)) {
            $champ->setErreur(true);
            $champ->setLibErreur('L\'année doit être comprise entre 1910 et 2030');
            return true;
        }
    }
    return false; //pas d'erreur
}
```

Explication du code : On intercepte le test sur le champ « annee ». Si sa valeur n'est pas comprise dans la bonne fourchette, on lève une erreur via la méthode `setErreur(true)` et on associe un message d'erreur à cette erreur via la méthode `setLibErreur`. Enfin on renvoie `true` afin de signifier une erreur.

Autre exemple de code :

```
protected function testsSupplementaires($champ) {
    defined('CHECK_INTEGER_7') || define('CHECK_INTEGER_7', '#^[0-9]{7}$#'); //7 chiffres obligatoires
    //les champs 'idMere' et 'idOuvre' doivent être composés de 7 chiffres (ou vide)
    if (in_array($champ->idField(), array('idMere', 'idOuvre'))) {
        if ($champ->value() != '') {
            if (!preg_match(CHECK_INTEGER_7, $champ->value())) {
                $champ->setErreur(true);
                $champ->setLibErreur(getLib('UFC_INTEGER_X', 7));
                return true;
            }
        }
    }
    return false; //pas d'erreur
}
```

ATTENTION : (rappel) les champs sont testés unitairement et en séquence, en commençant par le premier dans l'ordre de construction du formulaire. Si une erreur est détectée sur le premier champ, les tests supplémentaires ultérieurs ne sont pas effectués sur les champs suivants. Ils le seront dès que la saisie aura été corrigée sur le champ en erreur. Ceci est illustré par l'exemple ci-dessous.

Les champs étant testés en séquence et unitairement, cette méthode ne permet donc pas d'effectuer des tests relatifs entre champs. Par exemple, il est impossible de comparer deux champs. Un appel à la méthode `testsSupplementairesPosterieurs()` va permettre de palier à ce problème.

Tests supplémentaires postérieurs

La méthode `testsSupplementairesPosterieurs` est automatiquement appelée lorsque **TOUS** les tests séquentiels (vu précédemment) ont été réalisés. Ainsi, connaissant la valeur de chaque champ saisi, il est possible, en surchargeant cette méthode, d'effectuer des tests supplémentaires de comparaison entre champs.

Voici un exemple (idiot) de test comparatif qui peut être réalisé via l'appel à cette méthode :

```
protected function testsSupplementairesPosterieurs() {
    if (($this->field('titre')->value() == 'La Guerre des étoiles') &&
        ($this->field('annee')->value() != '1977')){
        $this->field('annee')->setErreur(true);
        $this->field('annee')->setLibereur('La Guerre des étoiles ne peut pas être sorti à une autre année que 1977');
        return true;
    }
    return false; //pas d'erreur
}
```

Ce qui donne :



ATTENTION : Notons qu'une fois de plus les tests supplémentaires postérieurs sont exécutés seulement si les tests unitaires n'ont pas renvoyé d'erreur.

NOTE IMPORTANTE : Toutes les méthodes de test `testsSupplementaires` et `testsSupplementairesPosterieurs` s'appuient sur les valeurs déjà postées du formulaire. Pour cette raison, modifier la valeur d'un champ dans une de ces méthodes via la méthode `setValue` n'aura aucun effet. La donnée modifiée dans une de cette méthode ne sera pas restituée par la méthode `getData`.

Notons également qu'il serait éventuellement possible de modifier ce comportement en repostant la valeur (ex : `$_POST[$this->field('id_user')->postName()] = $nouvelleValeur`). Il n'est cependant pas conseillé d'utiliser cette méthode. Préférez la modification d'une donnée en dernière minute après la séquence de tests en appelant la méthode `doUltimateThings` décrite plus loin.

8. Construisons un beau formulaire !

Nous allons construire un formulaire complexe ! Ceci va servir de petits exercices !

Boutons radio

Ajoutons à notre formulaire 3 boutons radio sous le groupe nommé visuel : « noir & blanc », « couleur », « noir & blanc et couleur »... et deux autres boutons ...

Voici le code correspondant :

```
protected function initDonnees() {
    $this->_tab_donnees['titre'] = '';
    $this->_tab_donnees['annee'] = '';
    $this->_tab_donnees['visuel'] = 'couleur';
}

//-----
// construction des champs du formulaire
//-----

protected function construitChamps() {
    parent::construitChamps();

    $this->createField('text', 'titre', array(
        'newLine' => true,
        'dbfield' => 'titre',
        'label' => 'Titre du film',
        'llong' => 'col-2',
        'lalign' => 'right',
        'labelHelp' => 'aide sur titre',
        'labelHelpPos' => 'right',
        'clong' => 'col-5',
        'testMatches' => array('REQUIRED'),
        'value' => $this->_tab_donnees['titre'],
    ));
    $this->createField('text', 'annee', array(
        'newLine' => true,
        'dbfield' => 'annee',
        'label' => 'Année de production',
        'llong' => 'col-2',
        'lalign' => 'right',
        'clong' => 'col-1',
        'testMatches' => array('REQUIRED', 'CHECK_INTEGER_4'),
        'value' => $this->_tab_donnees['annee'],
    ));
    $this->createField('radio', 'nb', array(
        'newLine' => true,
        'groupName' => 'visuel',
        'dbfield' => 'cnb',
        'dpos' => 'first',
        'titre' => 'Visuel',
        'tlong' => 'col-2',
        'talign' => 'right',
        'titreHelp' => 'Préciser la chromatographie du film',
        'label' => 'Noir & blanc',
        'lpos' => 'after',
        'clong' => 'col-3',
        'value' => 'nb',
        'checked' => ($this->_tab_donnees['visuel'] == 'nb')
    ));
    $this->createField('radio', 'couleur', array(
        'groupName' => 'visuel',
        'dbfield' => 'cnb',
        'dpos' => 'inter',
        'label' => 'Couleur',
        'lpos' => 'after',
        'checked' => ($this->_tab_donnees['visuel'] == 'couleur'),
        'value' => 'couleur'
    ));
    $this->createField('radio', 'nb_couleur', array(
        'groupName' => 'visuel',
```

Classe UniversalForm

```
'dbfield' => 'cnb',
'dpos' => 'last',
'label' => 'Noir & Blanc et Couleur',
'lpos' => 'after',
'checked' => ($this->_tab_donnees['visuel'] == 'les deux'),
'value' => 'les deux'
));

//construction bouton Submit
$this->createField('bouton', 'submit', array(
    'newLine' => true,
    'dbfield' => 'btvalide',
    'inputType' => 'submit',
    'decalage' => 'col-2',
    'label' => 'Ok',
    'llong' => 'col-12',
    'lclass' => 'btn btn-primary',
    'clong' => 'col-2',
    'value' => 'Ok'
));

//construction bouton Reset
$this->createField('bouton', 'annuler', array(
    'inputType' => 'reset',
    'label' => 'Annuler',
    'llong' => 'col-12',
    'lclass' => 'btn btn-secondary',
    'clong' => 'col-2',
    'value' => 'Annuler'
));

//construction bouton Button
$javascript = 'onclick="alert(\'Ceci est une action javascript\');"';
$this->createField('bouton', 'bouton', array(
    'inputType' => 'button',
    'label' => 'Test',
    'lclass' => 'btn btn-danger',
    'llong' => 'col-12',
    'clong' => 'col-2',
    'javascript' => $javascript,
    'value' => 'Test'
));
}
```

Le résultat graphique est le suivant :

Formulaire

Titre du film*

Année de production*

Visuel ☐ Noir & blanc ☒ Couleur
☐ Noir & Blanc et Couleur

(*) Champ requis (1) Lecture seule

Checkbox

Ajoutons deux séries de cases à cocher : une case à cocher intitulée « vu » qu'il faut cocher lorsque le film a été vu. Un groupe de trois cases à cocher « DVD » (cochée par défaut), « Blu-ray » et « DivX » avec le label commun « Possède ». L'utilisateur cochera les cases en fonction des supports qu'il possède pour ce titre. Enfin, nous allons effectuer un test : La case à cocher « Vu » ne pourra pas être cochée si la date de production est supérieure à l'année en cours. *Let's go !*

Voici le code correspondant :

```
protected function initDonnees() {
    $this->_tab_donnees['titre'] = '';
```

```

$this->_tab_donnees['annee'] = '';
$this->_tab_donnees['visuel'] = 'couleur';
$this->_tab_donnees['vu'] = false;
$this->_tab_donnees['dvd'] = true;
$this->_tab_donnees['bluray'] = false;
$this->_tab_donnees['divx'] = false;
}

protected function construireChamps() {
...

//Construction de la checkbox 'Vu'
$this->createField('checkbox', 'film_vu', array(
    'newLine' => true,
    'dbfield' => 'vu',
    'dpos' => 'alone',
    'titre' => 'Vu',
    'tlong' => 'col-2',
    'talign' => 'right',
    'titreHelp' => 'Cocher si le film a déjà été vu',
    'label' => '',
    'lpos' => 'after',
    'clong' => 'col-1',
    'checked' => ($this->_tab_donnees['vu'] == true)
));

//construction du groupe de checkbox 'possede'
$this->createField('checkbox', 'dvd', array(
    'groupName' => 'possede',
    'dbfield' => 'dvd',
    'dpos' => 'first',
    'titre' => 'Possède',
    'tlong' => 'col-1',
    'label' => 'DVD',
    'lpos' => 'before',
    'clong' => 'col-3',
    'border' => true,
    'checked' => ($this->_tab_donnees['dvd'] == true)
));
$this->createField('checkbox', 'bluray', array(
    'groupName' => 'possede',
    'dbfield' => 'bluray',
    'dpos' => 'inter',
    'label' => 'Blu-ray',
    'lpos' => 'before',
    'checked' => ($this->_tab_donnees['bluray'] == true)
));
$this->createField('checkbox', 'divx', array(
    'groupName' => 'possede',
    'dbfield' => 'divx',
    'dpos' => 'last',
    'label' => 'DivX',
    'lpos' => 'before',
    'checked' => ($this->_tab_donnees['divx'] == true)
));
...

```

Le résultat graphique est le suivant :

Formulaire

Titre du film*

Année de production*

Visuel

☐ Noir & blanc
☒ Couleur
☐ Noir & Blanc et Couleur

Vu
☐
Possède

DVD ☒
Blu-ray ☐
DivX ☐

(*) Champ requis (1) Lecture seule

Comme le test demandé est un test de comparaison entre deux champs, il est obligatoire de l'ajouter dans la méthode `testssupplementairesPosterieurs`.

```
protected function testsSupplementairesPosterieurs() {
    if (($this->field('titre')->value() == 'La Guerre des étoiles') &&
        ($this->field('annee')->value() != '1977')){
        $this->field('annee')->setErreur(true);
        $this->field('annee')->setLiberreux('La Guerre des étoiles ne peut pas être sorti à une autre année que 1977');
        return true;
    }
    if (($this->field('film_vu')->checked()) && ($this->field('annee')->value() > date('Y'))){
        $this->field('film_vu')->setErreur(true);
        $this->field('film_vu')->setLiberreux('Vous ne pouvez pas avoir vu ce film');
        return true;
    }
    return false; //pas d'erreur
}
```

Notons que nous avons utilisé ici la méthode `checked()` de l'objet `Checkbox` pour tester l'état de la boîte à cocher.

Select

Ajoutons maintenant une boîte déroulante qui permette de sélectionner le genre du film (science-fiction, drame, comédie, etc.). La complexité du select tient dans son affichage...

Voici le code correspondant :

```
//-----
// Fonction de callback
// code à insérer dans l'application
//-----
function fillSelect($value)
{
    $html = '';
    ($value == '') ? $default = ' selected' : $default = '';
    $html.= '<option value=""'.$default.'>Choisissez un genre</option>';
    ($value == 'action') ? $default = ' selected' : $default = '';
    $html.= '<option value="action"'.$default.'>Action</option>';
    ($value == 'comedie') ? $default = ' selected' : $default = '';
    $html.= '<option value="comedie"'.$default.'>Comédie</option>';
    ($value == 'horreur') ? $default = ' selected' : $default = '';
    $html.= '<option value="horreur"'.$default.'>Horreur</option>';
    ($value == 'romance') ? $default = ' selected' : $default = '';
    $html.= '<option value="romance"'.$default.'>Romance</option>';
    return $html;
}

//-----
// modification à apporter à Form_videotheque.class.php
//-----

protected function initDonnees() {
    $this->_tab_donnees['titre'] = '';
    $this->_tab_donnees['annee'] = '';
    $this->_tab_donnees['visuel'] = 'couleur';
    $this->_tab_donnees['vu'] = false;
    $this->_tab_donnees['dvd'] = true;
    $this->_tab_donnees['bluray'] = false;
    $this->_tab_donnees['divx'] = false;
    $this->_tab_donnees['genre'] = 'horreur';
}

protected function construitChamps() {
    ...
    $this->createField('select', 'genre_film', array(
        'newLine' => true,
        'dbfield' => 'genre',
        'label' => 'Genre',
        'llong' => 'col-2',
        'lalign' => 'right',
        'complement' => 'fillSelect',
        'clong' => 'col-3',
        'value' => $this->_tab_donnees['genre']
    ));
    ...
}
```

Classe UniversalForm

Explications : Tout d'abord, il faut écrire la fonction callback qui va se charger d'écrire le code HTML de construction de la liste déroulante (en réalité, uniquement les options). Ici, elle s'appelle **fillselect** avec le paramètre **\$value** qui va permettre de sélectionner l'item par défaut, c'est-à-dire celui affiché et visible. Par exemple, l'affichage de « Horreur » renverra comme valeur « horreur » (sans majuscule). On aurait pu choisir un entier ou n'importe quelle autre valeur qui correspondrait à ce terme. La méthode renvoie le code HTML ci-dessous, avec l'option sélectionnée (Horreur).

```
<option value="">Choisissez un genre</option>
<option value="action">Action</option>
<option value="comedie">Comédie</option>
<option value="horreur" selected="">Horreur</option>
<option value="romance">Romance</option>
```

Il faut ensuite initialiser le champ à afficher. Pour rappel, l'initialisation des champs se fait dans la méthode **initDonnees**. Nous avons choisi d'afficher « Horreur » par défaut.

Le résultat graphique est le suivant :

Formulaire

Titre du film*

Année de production*

Visuel ☐ Noir & blanc ☒ Couleur
☐ Noir & Blanc et Couleur

Vu ☐ Possède ☒ DVD ☐ Blu-ray ☐ DivX

Genre

(*) Champ requis (!) Lecture seule

Pour aller plus loin : voici un exemple de fonction générique qui permet de construire le code HTML de remplissage d'une liste déroulante (à partir d'une base de données) et qui fonctionne également pour une liste non déroulante à choix multiple dont **\$default** serait un tableau de plusieurs valeurs de la sélection en cours.

```
function buildThemesListe($default)
{
    $requete = 'SELECT id_theme, theme_fr theme FROM sfm_themes';
    $themes = executeQuery($requete, $nombre, _SQL_MODE_);
    $texte = '';
    foreach($themes as $theme) {
        if (is_array($default)) {
            if (in_array($theme['id_theme'], $default)) $selected = ' selected'; else $selected = '';
        }
        elseif ($default == $theme['id_theme']) {
            $selected = ' selected'; else $selected = '';
            $texte.= '<option value="'. $theme['id_theme'] ."'>".$theme['theme'];
            $texte.= '</option>';
        }
    }
    return $texte;
}
```

9. Affiner la construction du formulaire

Le formulaire étant construit en séquence champ par champ dans la méthode `construitChamps`, il n'est pas possible de lier la construction de certains champs en fonction du contenu d'autres. Dans l'état actuel des choses, voici un exemple de ce que l'on ne peut pas faire : dans notre petite application, si le genre « Comédie » était sélectionné, on aimerait que les champs « titre » et « année » apparaissent sur fond de couleur (couleur Bootstrap `bg-info` par exemple). Ceci ne peut effectivement pas être défini à partir de la méthode `construitChamps`.

Pour pallier au problème, il faut pouvoir accéder à une méthode qui vienne modifier les champs à posteriori de leur construction. Cette méthode virtuelle existe et il faut la surcharger. C'est la méthode `doThings`. Elle doit être appelée juste après la méthode `construitChamps`.

```
protected function doThings() {
    if ($this->field('genre_film')->value() == 'comedie') {
        $this->field('titre')->setCclass(trim($this->field('titre')->cclass().' bg-info'));
        $this->field('annee')->setCclass(trim($this->field('annee')->cclass().' bg-info'));
    }
}

//initialisation des données et construction des champs initialisés
public function init() {
    $this->initDonnees();           //initialisation des données
    $this->construitChamps();        //construction à vide... (cad avec données d'initiation)
    $this->doThings();
}
```

Explications : Si la valeur du champ `genre` est « comedie » (la valeur pas le libellé !), alors on lit la classe CSS actuelle appliquée aux champs `titre` et `annee` et on lui ajoute la classe `bg-info`. La lecture de la classe se fait pas l'appel de la méthode `cclass`. L'écriture de la classe se fait par la méthode `setCclass`.

Nous remarquons pourtant que cela ne semble pas marcher. C'est normal ! Le champ `genre_film` a été initialisé avec le genre « horreur ». Arrivant juste après la construction, le genre étant différent de « comedie », les champs `titre` et `annee` ne sont pas coloriés en bleu clair (`bg-info`). Par contre on pourrait penser qu'en sélectionnant « Comédie », les champs se colorent... il n'en est pourtant rien non plus ! Là aussi c'est normal ! En sélectionnant un autre genre depuis la liste le formulaire n'envoie aucune donnée (pas de POST), donc on ne réaffiche pas le formulaire pour prendre en compte l'action. La seule façon de s'en sortir est d'utiliser JavaScript.

JavaScript ? C'est dur ça ! Pas de panique, tout est prévu ! Il suffit d'entrer le code JavaScript suivant dans la propriété `javascript` du `select`.

```
$javascript = 'onchange="';
$javascript.= 'if (this.options[this.selectedIndex].value == \'comedie\') {';
$javascript.= '    document.getElementById(\'idTitre\').classList.add(\'bg-info\');';
$javascript.= '    document.getElementById(\'idAnnee\').classList.add(\'bg-info\');';
$javascript.= '}';
$javascript.= 'else {';
$javascript.= '    document.getElementById(\'idTitre\').classList.remove(\'bg-info\');';
$javascript.= '    document.getElementById(\'idAnnee\').classList.remove(\'bg-info\');';
$javascript.= '}';
$javascript.= '";
$this->createField('select', 'genre_film', array(
    'newLine' => true,
    'dbfield' => 'genre',
    'label' => 'Genre',
    'llong' => 'col-2',
    'lalign' => 'right',
    'complement' => 'fillSelect',
    'clong' => 'col-3',
    'value' => $this->_tab_donnees['genre'],
```


Classe UniversalForm

```
'javascript' => $javascript  
));
```

Voici le résultat...

Formulaire

Titre du film*

Année de production*

Visuel ☐ Noir & blanc ☒ Couleur
☐ Noir & Blanc et Couleur

Vu ☐ Possède ☐ DVD ☒ Blu-ray ☐ DivX ☐

Genre

(*) Champ requis (1) Lecture seule

10. Bibliothèque de fonctions JavaScript

Une bibliothèque de plusieurs fonctions JavaScript a été prédéveloppée pour nos formulaires. Pour l'installer ajoutez à votre code dans le header la ligne suivante :

```
echo '<script src="js/php.js"></script>';  
echo '<script src="js/universalform.min.js"></script>';
```

ATTENTION : ces fonctions javascript utilisent le fichier `php.js` qui contient des fonctions standard du php réécrites pour javascript. Penser à donc bien ajouter aussi cette bibliothèque avant.

capLock(event, field, lg)

Lorsque la touche majuscule est enfoncée, affiche un avertissement sous le champ text pour lequel il est destiné. Sert par exemple à alerter l'utilisateur lors de la saisie d'un mot de passe.

event : évènements Javascript

field : champ de saisie concerné

lg : langue d'affichage de l'information (**fr** / **en**)

Exemple d'appel pour un champ « mode de passe »

```
// mot de passe  
$this->createField('text', 'old_password', array(  
    'newline' => false,  
    'dbfield' => 'old_password',  
    'inputType' => 'password',  
    'design' => 'online',  
    'label' => getLib('PASSWORD_ANCIEN'),  
    'labelHelp' => getLib('X_CHARACTERES_MINIMUM', 5),  
    'clong' => 'col-12 col-sm-6 mb-3 mr-1',  
    'maxlength' => 100,  
    'spellcheck' => false,  
    'placeholder' => mb_strtolower(getLib('PASSWORD_ANCIEN')),  
    'testMatches' => '',  
    'value' => $this->_tab_donnees['old_password'],  
    'javascript' => 'onkeypress="capLock(event, \'idOld_password\', \'fr\')"'  
));
```

uf_setInvisible(field)

Rend le champ **field** invisible

uf_setVisible(field)

Rend le champ **field** visible

uf_enable(field)

Active le champ **field** passé en paramètre

uf_disable(field)

Désactive le champ **field** passé en paramètre

uf_enableOnChecked(field, declencheur)

Active ou désactive le champ **field** **SELON** l'état « checked » du champ **declencheur**. Par exemple utilisé pour activer ou désactiver un champ selon l'état d'une case à cocher.

Si la case à cocher est cochée, le champ **field** est activé.

Si la case à cocher est décochée, le champ **field** est désactivé.

uf_disableOnChecked(field, declencheur)

Classe UniversalForm

Désactive ou active le champ `field` **SELON** l'état « checked » du champ `déclencheur`. Par exemple utilisé pour activer ou désactiver un champ selon l'état d'une case à cocher.

Si la case à cocher est cochée, le champ `field` est désactivé.

Si la case à cocher est décochée, le champ `field` est activé.

`uf_justEnableOnUnchecked(field, déclencheur)`

Active le champ `field` **SI** l'état « checked » du champ `déclencheur` **N'EST PAS** validé. Ne fait rien dans le cas contraire.

`uf_justCheckOnUnchecked(field, déclencheur)`

Coche la case à cocher `field` **SI** l'état « checked » du champ `déclencheur` **N'EST PAS** validé. Ne fait rien dans le cas contraire.

`uf_check(field)`

Coche la case à cocher `field`.

`uf_uncheck(field)`

Décoche la case à cocher `field`.

`uf_setValue(field, valeur)`

Affiche la valeur `valeur` dans le champ `field`.

`uf_changeClass(field, classe)`

Modifie la classe CSS du champ `field` par `classe`.

`uf_showOnChecked(field, déclencheur)`

Rend visible ou invisible le champ `field` **SELON** l'état « checked » du champ `déclencheur`. Par exemple utilisé pour rendre visible ou invisible un champ selon l'état d'une case à cocher.

Si la case à cocher est cochée, le champ `field` est rendu visible.

Si la case à cocher est décochée, le champ `field` est rendu invisible.

Exemple d'utilisation que l'on peut en faire : On souhaite désactiver la liste déroulante des genres si l'on n'a pas vu le film. Il suffit de synchroniser l'état de la liste déroulante `genre_film` avec l'état de la case à cocher « vu ». Voici le code du champ `film_vu`.

```
//Construction de la checkbox 'Vu'
$javascript = 'onclick=';
$javascript.= 'uf_disableOnChecked(\'genre_film\', \'film_vu\')';
$javascript.= ' ';
$this->createField('checkbox', 'film_vu', array(
    'newLine' => true,
    'dbfield' => 'vu',
    'dpos' => 'alone',
    'titre' => 'Vu',
    'tlong' => 'col-2',
    'talign' => 'right',
    'titreHelp' => 'Cocher si le film a déjà été vu',
    'label' => '',
    'lpos' => 'after',
    'clong' => 'col-1',
    'checked' => ($this->_tab_donnees['vu'] == true),
    'javascript' => $javascript
));
```

11. Modifier les données avant réception

Comme nous l'avons vu, la récupération de la saisie du formulaire est demandée par appel à la méthode `getData` qui renvoie un tableau constitué des champs que le développeur a choisi de renvoyer (par présence de la propriété `dbfield` dans la définition des champs choisis).

Cependant, il peut être intéressant d'accéder aux données juste avant leur envoi, soit pour corriger une donnée, soit pour ajouter d'autres informations "à la volée" à la liste des champs et valeurs retournés. Pour ce faire, il faut faire appel à la méthode protégée `doUltimateThings`.

Cette méthode est systématiquement appelée juste avant de renvoyer les données à l'utilisateur. Si l'on souhaite s'en servir, il faut alors la surcharger. Voici un exemple :

```
protected function doUltimateThings(&$donnees) {  
    //création d'un champ supplémentaire  
    $donnees['preferences'] = 24;  
    //modification d'un champ existant  
    if ($donnees['titre'] = 'Star Wars') $donnees['genre'] = 'Science-fiction';  
}
```

Le tableau de lecture du formulaire renvoyé sera le suivant :

```
$donnees = (tableau)  
Array  
(  
    [titre] => Star Wars  
    [annee] => 1977  
    [cnb] => couleur  
    [vu] => 0  
    [dvd] => 1  
    [bluray] => 1  
    [divx] => 0  
    [genre] => Science-fiction  
    [btvalide] => Ok  
    [preferences] => 24  
)
```

On le voit, à la validation du formulaire, le tableau de données renvoie donc le champ `preferences` avec la valeur 24 qui n'a pourtant pas été définie lors de la fabrication du formulaire. De plus, `genre` renvoie « Science-fiction » qui n'existe pourtant pas dans le sélecteur.

IMPORTANT : Il est important de noter que le paramètre d'entrée de cette méthode protégée doit être passé par adresse et non par valeur (ajout du `&` commercial devant la variable).

12. Envoyer un message à l'application

Le formulaire est capable d'envoyer un message à l'application pour les faire communiquer. Ce peut être n'importe quelle information qui pourra être réutilisée par l'application (texte, valeur, etc.) Par exemple, on peut passer un message d'erreur généré par le formulaire à l'application afin que celle-ci le répercute dans un fichier de log.

Deux méthodes permettent la gestion de ces messages, un getter et un setter.

setMessage(\$message) permet de notifier le message.

getMessage() permet de récupérer le message envoyé par le formulaire.

```
//exemple d'envoi d'un message depuis le formulaire
$this->setMessage('Erreur sur le champ nombre détecté');

//exemple de récupération du message par l'application $frm étant l'objet formulaire
$messageFormulaire = $frm->getMessage();
```

13. Code complet

Voici le code complet de notre tutoriel pas à pas. Pour rappel, par souci de simplification, cet exemple fait appel à la bibliothèque commune **common.inc.php** de **UniversalWeb** qui permet de s'affranchir de quelques définitions préalables mais reste un bon exemple d'utilisation *standalone* de la classe **UniversalForm** pour vos projets. Si vous devez utiliser la structure **UniversalWeb** préférez l'utilisation du script « prêt-à-utiliser » **squelette_formulaire** proposé dans l'outil.

```
<?php
require_once('libs/common.inc.php');

//création du formulaire
$monFormulaire = new Form_videothèque(UniversalForm::AJOUTER, 1);

//création de la page HTML
echo '<!doctype html>';
echo '<html lang="fr">';
echo '<head>';
    echo '<meta charset="utf-8" />';
    //prise en compte pour l'affichage responsive
    echo '<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">';
    echo '<meta http-equiv="x-ua-compatible" content="ie=edge">';
    echo '<link rel="stylesheet" href="bootstrap-4.3.1-dist/css/bootstrap.min.css">'; //CSS Bootstrap
    echo '<title>Vidéothèque</title>';
echo '</head>';

echo '<body>';
    echo '<div class="container-fluid">';
        echo '<div class="row mt-3">';
            echo '<div class="col-6 ml-auto mr-auto">';

                //panel de l'application
                echo '<div class="row p-3">';
                    echo '<div class="col-12 bg-light">';
                        echo '<p class="h1">Saisie vidéothèque</p>';
                        echo '<p class="lead">Version UniversalForm : ' . UniversalForm::VERSION . '</p>';
                    echo '</div>';
                echo '</div>';

                //code propre à la page
                echo '<div class="row mt-3">';
                    echo '<div class="col-12">';

                        //création du formulaire
                        $monFormulaire = new Form_videothèque(UniversalForm::AJOUTER, 1);
```

Classe UniversalForm

```
$action = $monFormulaire->getAction();

if ($action == 'ajouter') {
    $monFormulaire->init();
    echo $monFormulaire->afficher();
}
elseif ($action == 'valid_ajouter') {
    if (!$monFormulaire->tester()) {
        echo $monFormulaire->afficher();
    }
    else {
        $lesDonnees = $monFormulaire->getData();
        echo '<pre>';
        print_r($lesDonnees);
        echo '</pre>';
    }
}

echo '</div>';
echo '</div>';

echo '</div>'; //fin colonne centrale
echo '</div>';
echo '</div>';

//Chargement du code Javascript nécessaire au
//fonctionnement de bootstrap
echo '<script src="js/jquery-3.3.1.min.js"></script>';
echo '<script>';
echo '$(document).ready(function () {';
echo '$("[data-toggle=\'tooltip\"]').tooltip();'
//autres fonctions jQuery à charger ...
echo '});';
echo '</script>';

echo '<script src="bootstrap-4.3.1-dist/js/bootstrap.bundle.min.js"></script>';
echo '<script src="js/php.js"></script>';
echo '<script src="js/universalform.min.js"></script>';

echo '</body>';
echo '</html>';
```

saisie_videotheque.php

```
<?php

//-----
// Fonction de callback
// code à insérer dans l'application
//-----
function fillSelect($value)
{
    $html = '';
    ($value == '') ? $default = ' selected' : $default = '';
    $html.= '<option value=""'.$default.'>Choisissez un genre</option>';
    ($value == 'action') ? $default = ' selected' : $default = '';
    $html.= '<option value="action"'.$default.'>Action</option>';
    ($value == 'comédie') ? $default = ' selected' : $default = '';
    $html.= '<option value="comédie"'.$default.'>Comédie</option>';
    ($value == 'horreur') ? $default = ' selected' : $default = '';
    $html.= '<option value="horreur"'.$default.'>Horreur</option>';
    ($value == 'romance') ? $default = ' selected' : $default = '';
    $html.= '<option value="romance"'.$default.'>Romance</option>';
    return $html;
}

class Form_videotheque extends UniversalForm {

    private $_tab_donnees = array(); //tableau des données associées au formulaire

    //=====
    // Méthodes protégées
    //=====
    // Initialisation des données de travail
    //-----

    protected function initDonnees() {
        //initialisations supplémentaires
        $this->_tab_donnees['titre'] = '';
        $this->_tab_donnees['annee'] = '';
        $this->_tab_donnees['visuel'] = 'couleur';
        $this->_tab_donnees['vu'] = false;
        $this->_tab_donnees['dvd'] = true;
        $this->_tab_donnees['bluray'] = false;
        $this->_tab_donnees['divx'] = false;
        $this->_tab_donnees['genre'] = 'horreur';
    }
}
```

```
//-----  
// construction des champs du formulaire  
//-----  
  
protected function construireChamps() {  
    parent::construireChamps();  
  
    $this->createField('text', 'titre', array(  
        'newLine' => true,  
        'dbfield' => 'titre',  
        'label' => 'Titre du film',  
        'llong' => 'col-2',  
        'lalign' => 'right',  
        'labelHelp' => 'aide sur titre',  
        'labelHelpPos' => 'right',  
        'clong' => 'col-5',  
        'testMatches' => array('REQUIRED'),  
        'value' => $this->_tab_donnees['titre'],  
    ));  
    $this->createField('text', 'annee', array(  
        'newLine' => true,  
        'dbfield' => 'annee',  
        'label' => 'Année de production',  
        'llong' => 'col-2',  
        'lalign' => 'right',  
        'clong' => 'col-2',  
        'testMatches' => array('REQUIRED', 'CHECK_INTEGER_4'),  
        'value' => $this->_tab_donnees['annee'],  
    ));  
    $this->createField('radio', 'nb', array(  
        'newLine' => true,  
        'groupName' => 'visuel',  
        'dbfield' => 'cnb',  
        'dpos' => 'first',  
        'titre' => 'Visuel',  
        'tlong' => 'col-2',  
        'talign' => 'right',  
        'titreHelp' => 'Préciser la chromatographie du film',  
        'label' => 'Noir & blanc',  
        'lpos' => 'after',  
        'clong' => 'col-3',  
        'value' => 'nb',  
        'checked' => ($this->_tab_donnees['visuel'] == 'nb')  
    ));  
    $this->createField('radio', 'couleur', array(  
        'groupName' => 'visuel',  
        'dbfield' => 'cnb',  
        'dpos' => 'inter',  
        'label' => 'Couleur',  
        'lpos' => 'after',  
        'checked' => ($this->_tab_donnees['visuel'] == 'couleur'),  
        'value' => 'couleur'  
    ));  
    $this->createField('radio', 'nb_couleur', array(  
        'groupName' => 'visuel',  
        'dbfield' => 'cnb',  
        'dpos' => 'last',  
        'label' => 'Noir & Blanc et Couleur',  
        'lpos' => 'after',  
        'checked' => ($this->_tab_donnees['visuel'] == 'les deux'),  
        'value' => 'les deux'  
    ));  
  
    //Construction de la checkbox 'Vu'  
    $javascript = 'onclick="';  
    $javascript.= 'uf_disableOnChecked(\'genre_film\', \'film_vu\')';  
    $javascript.= '";'  
    $this->createField('checkbox', 'film_vu', array(  
        'newLine' => true,  
        'dbfield' => 'vu',  
        'dpos' => 'alone',  
        'titre' => 'Vu',  
        'tlong' => 'col-2',  
        'talign' => 'right',  
        'titreHelp' => 'Cocher si le film a déjà été vu',  
        'label' => '',  
        'lpos' => 'after',  
        'clong' => 'col-1',  
        'checked' => ($this->_tab_donnees['vu'] == true),  
        'javascript' => $javascript  
    ));  
  
    //construction du groupe de checkbox 'possede'  
    $this->createField('checkbox', 'dvd', array(  
        'groupName' => 'possede',  
        'dbfield' => 'dvd',
```

```

        'dpos' => 'first',
        'titre' => 'Possède',
        'tlong' => 'col-1',
        'label' => 'DVD',
        'lpos' => 'before',
        'clong' => 'col-3',
        'border' => true,
        'checked' => ($this->_tab_donnees['dvd'] == true)
    ));
    $this->createField('checkbox', 'bluray', array(
        'groupName' => 'possede',
        'dbfield' => 'bluray',
        'dpos' => 'inter',
        'label' => 'Blu-ray',
        'lpos' => 'before',
        'checked' => ($this->_tab_donnees['bluray'] == true)
    ));
    $this->createField('checkbox', 'divx', array(
        'groupName' => 'possede',
        'dbfield' => 'divx',
        'dpos' => 'last',
        'label' => 'DivX',
        'lpos' => 'before',
        'checked' => ($this->_tab_donnees['divx'] == true)
    ));

    //construction liste déroulante
    $javascript = 'onchange=';
    $javascript.= 'if (this.options[this.selectedIndex].value == \'comédie\') {';
    $javascript.= '    document.getElementById(\'idTitre\').classList.add(\'bg-info\');';
    $javascript.= '    document.getElementById(\'idAnnee\').classList.add(\'bg-info\');';
    $javascript.= '};';
    $javascript.= 'else {';
    $javascript.= '    document.getElementById(\'idTitre\').classList.remove(\'bg-info\');';
    $javascript.= '    document.getElementById(\'idAnnee\').classList.remove(\'bg-info\');';
    $javascript.= '};';
    $javascript.= '";';
    $this->createField('select', 'genre_film', array(
        'newLine' => true,
        'dbfield' => 'genre',
        'label' => 'Genre',
        'llong' => 'col-2',
        'lalign' => 'right',
        'complement' => 'fillSelect',
        'clong' => 'col-3',
        'value' => $this->_tab_donnees['genre'],
        'javascript' => $javascript
    ));

    //construction bouton Submit
    $this->createField('bouton', 'submit', array(
        'newLine' => true,
        'dbfield' => 'btvalide',
        'inputType' => 'submit',
        'decalage' => 'col-2',
        'label' => 'Ok',
        'llong' => 'col-12',
        'lclass' => 'btn btn-primary',
        'clong' => 'col-2',
        'value' => 'Ok'
    ));

    //construction bouton Reset
    $this->createField('bouton', 'annuler', array(
        'inputType' => 'reset',
        'label' => 'Annuler',
        'llong' => 'col-12',
        'lclass' => 'btn btn-secondary',
        'clong' => 'col-2',
        'value' => 'Annuler'
    ));

    //construction bouton Button
    $javascript = 'onclick="alert(\'Ceci est une action javascript\');";';
    $this->createField('bouton', 'bouton', array(
        'inputType' => 'button',
        'label' => 'Test',
        'lclass' => 'btn btn-danger',
        'llong' => 'col-12',
        'clong' => 'col-2',
        'javascript' => $javascript,
        'value' => 'Test'
    ));
}

//=====
// Méthodes publiques
//=====

```


Classe UniversalForm

```
//initialisation des données et construction des champs initialisés
public function init() {
    $this->initDonnees();           //initialisation des données
    $this->construitChamps();        //construction à vide... (cad avec données d'initiation)
}

protected function testsSupplementaires($champ) {
    if ($champ->idField() == 'annee') {
        if (($champ->value() < 1920) || ($champ->value() > 2030)) {
            $champ->setErreur(true);
            $champ->setLiberreur('L\'année doit être comprise entre 1910 et 2030');
            return true;
        }
    }
    return false; //pas d'erreur
}

protected function testsSupplementairesPosterieurs() {
    if (($this->field('titre')->value() == 'la guerre des étoiles') && ($this->field('annee')->value() != '1977')) {
        $this->field('annee')->setErreur(true);
        $this->field('annee')->setLiberreur('La Guerre des étoiles ne peut pas être sorti à une autre année que 1977');
        return true;
    }
    if (($this->field('film_vu')->checked()) && ($this->field('annee')->value() > date('Y'))){
        $this->field('film_vu')->setErreur(true);
        $this->field('film_vu')->setLiberreur('Vous ne pouvez pas avoir vu ce film');
        return true;
    }
    return false; //pas d'erreur
}

protected function doThings() {
    if ($this->field('genre_film')->value() == 'comedie') {
        $this->field('titre')->setCclass(trim($this->field('titre')->cclass().' bg-info'));
        $this->field('annee')->setCclass(trim($this->field('annee')->cclass().' bg-info'));
    }
}

protected function doUltimateThings(&$donnees) {
    //création d'un champ supplémentaire
    $donnees['preferences'] = 24;
    //modification d'un champ existant
    if ($donnees['titre'] = 'Star Wars') $donnees['genre'] = 'Science-fiction';
}

//-----
// affichage du formulaire
//-----
public function afficher() {
    parent::afficher();           //permet d'ajouter des tests de construction du formulaire
    //ATTENTION : les champs disabled ne renvoient aucun POST !!!
    //Donc impossible de récupérer les données depuis une suppression
    $enable = (!($this->getOperation() == self::CONSULTER) || ($this->getOperation() == self::SUPPRIMER));

    $chaine = '';
    $chaine.= '<form class="uf" action="'.$_SERVER['REQUEST_URI'].'" method="post" enctype="multipart/form-data">';
    $chaine.= '<fieldset class="border p-3">';
    $chaine.= '<h1>Formulaire</h1>';
    $chaine.= $this->draw($enable);
    $chaine.= '<p class="small">(*) Champ requis (1) Lecture seule</p>';
    $chaine.= '</fieldset>';
    $chaine.= '</form>';
    return $chaine;
}
}
```

Form_videotheque.php

14. Getters et Setters

Comme pour tout objet, il existe des « getters » et des « setters », c'est-à-dire des méthodes qui permettent d'accéder aux propriétés des objets, soit en les lisant (getters) soit en écrivant dedans (setters). Voici ici un récapitulatif de ces méthodes publique si besoin disponibles. Beaucoup ne seront pas d'utilité... attention à ce que vous écrivez dedans !

Getters

Getter	Informations S'applique à
aclass()	Classe CSS personnalisée de l'extension d'un champ search .
addon()	Renvoie l'état du type de champ search (true : le champ search possède une extension, false sinon).
apos()	Position de l'extension d'un champ search par rapport à sa zone de saisie (before / after).
border()	Renvoie le type de bordure du champ. true , le champ possède une bordure standard, false pas de bordure, sinon le code CSS pour la bordure personnalisée.
cclass()	Classe CSS personnalisée de la zone de champ (ou du champ).
clong()	Renvoie la longueur de la zone de champ (en colonnes Bootstrap).
complement()	Complément d'information pour le champ. Utilisé par les champs de type text et select .
dbfield()	Nom du champ dans le tableau de retour validé par le formulaire.
field('id')	Renvoie l' objet champ dont l'identificateur est passé en paramètre.
decalage()	Renvoie le décalage (en colonnes Bootstrap) du champ pour son affichage.
design()	Renvoie le design du champ (inline / online)
dpos()	Renvoie la position d'un champ de type radio ou checkbox dans son groupe (first , inter , last , alone).
enable()	Booléen : true , le champ est actif, false sinon.
erreur()	Booléen : true , il y a une erreur sur le champ, false , pas d'erreur.

Classe UniversalForm

fieldType()	Renvoie le type de champ (chaîne de caractère parmi <code>area</code> , <code>bouton</code> , <code>checkbox</code> , <code>comment</code> , <code>div</code> , <code>divfin</code> , <code>filtretext</code> , <code>filtreselect</code> , <code>hidden</code> , <code>image</code> , <code>radio</code> , <code>search</code> , <code>select</code> , <code>separateur</code> et <code>text</code>).
flexLine()	Renvoie le type de transformation <code>flex</code> des champs de la ligne.
id()	Id JavaScript du champ.
idbchamp()	Id JavaScript du « bloc de champ » du champ.
idField()	Renvoie le nom (identifiant) du champ.
idParentForm()	Renvoie l'id du formulaire parent d'un élément de formulaire. Cet id est l'identifiant unique donné au formulaire lors de sa création par instanciation de la classe <code>UniversalForm</code> , en particulier dans son deuxième paramètre. <i>new(operation, id)</i> .
idzchamp()	Id JavaScript de la « zone de champ » du champ.
idztitre()	Id JavaScript de la « zone de titre » du champ.
inputType()	Type de champ input (text). Choix parmi : <code>button</code> , <code>checkbox</code> , <code>color</code> , <code>date</code> , <code>datetime</code> , <code>datetime-local</code> , <code>email</code> , <code>file</code> , <code>hidden</code> , <code>image</code> , <code>month</code> , <code>number</code> , <code>password</code> , <code>radio</code> , <code>range</code> , <code>reset</code> , <code>search</code> , <code>submit</code> , <code>tel</code> , <code>text</code> , <code>time</code> , <code>url</code> , <code>week</code> . Par défaut <code>text</code> .
invisible()	Booléen : <code>true</code> , le champ est invisible, <code>false</code> sinon.
javascript()	Code JavaScript associé au champ.
label()	Label du champ.
labelHelp()	Libelle d'aide qui s'affiche au survol du label du champ.
labelHelpPos()	Position du libelle d'aide qui s'affiche au survol du label du champ par rapport au champ (<code>auto</code> , <code>left</code> , <code>top</code> , <code>right</code> , <code>bottom</code>). La valeur <code>auto</code> laisse l'application décider de la meilleure position en fonction des autres éléments alentour.
lalign()	Renvoie l'alignement du label (<code>left</code> , <code>center</code> , <code>right</code> , <code>justify</code>) dans sa zone de titre.
labelPlus()	Libellé droite d'un libellé scindé
labelPlusHelp()	Libelle d'aide qui s'affiche au survol du label de droite scindé.
labelPlusHelpPos()	Position du libelle d'aide qui s'affiche au survol du label de droite du libellé scindé (<code>auto</code> , <code>left</code> , <code>top</code> , <code>right</code> , <code>bottom</code>). La valeur <code>auto</code> par laisse l'application décider de la meilleure position en fonction des autres éléments alentour.

Classe UniversalForm

lclass()	Classe CSS personnalisée de la zone de titre (ou du label).
liberreur()	Libellé d'erreur.
liberreurHelp()	Libellé d'aide affiché sur le champ erreur (pour donner plus d'information sur l'erreur).
llong()	Longueur de la zone de titre en colonnes Bootstrap.
lpos()	Position du label du champ par rapport au champ (before / after)
maxlength()	Taille maximum du champ de type text .
newLine()	Dit si le champ commence son dessin sur une nouvelle ligne. Booléen.
placeholder()	Text en surimpression posée sur le champ avant sa saisie.
postIsTab	Booléen : true , le POST du champ est un tableau, false sinon
postName()	Nom de la variable associée au champ pour le POST du formulaire. Généré automatiquement par la classe. A priori vous ne deviez pas en avoir besoin. Il est même conseillé de ne pas y toucher. Il faut simplement savoir que le nom est construit à partir d'un préfixe propre au type de champ, suivi du nom du champ et de l'id du formulaire. Ex : memCommentaires_1 , txtAnnee_1 , etc.
readonly()	Booléen : True , le champ est en lecture seule, false sinon.
showErreur()	Booléen. Indique si les erreurs qui vont se produire sur le champ vont être affichées.
spellcheck()	Booléen. Renvoie l'état d'activation du correcteur orthographique du navigateur pour le champ (les éventuelles fautes d'orthographe son soulignées en rouge). Si true , signifie que cette option dépend du paramétrage du navigateur .
talign()	Renvoie l'alignement du titre (left , center , right , justify) dans sa zone de titre.
tclass()	Classe CSS personnalisée du titre du champ.
testMatches()	Tableau des tests unitaires programmés pour le champ
titre()	Titre (ou légende) du champ.
titreHelp()	Libellé d'aide affiché au survol du titre du champ.
titreHelpPos	Position du libelle d'aide qui s'affiche au survol du titre par rapport à au titre (auto , left , top , right , bottom). La valeur auto par laisse l'application décider de la meilleure position en fonction des autres éléments alentour.
tlong()	Longueur du titre en colonnes Bootstrap.

Classe UniversalForm

value()	Valeur du champ.
----------------	------------------

Setters

En rouge, il est préférable de ne pas y toucher. Servent à la conception de la classe.

En bleu, l'utilisateur peut y accéder mais la meilleure façon d'y accéder est via les propriétés de construction du champ.

En vert, l'utilisateur peut y avoir accès à volonté hors de la phase de construction du champ via la méthode `construitChamps`.

Getter	Informations S'applique à
seIdParentForm()	Positionne l'ID unique du formulaire. La méthode est automatiquement appelée lors de l'instanciation de la classe UniversalForm <i>new(operation, id)</i>
setFieldType()	Positionne le type de champ. Ne pas toucher, réservé aux concepteurs.
setIdField()	Positionne le nom (identifiant) du champ. Automatique. Ne pas toucher, réservé aux concepteurs. Méthode privée.
setDbfield()	Positionne le nom du champ dans le tableau de retour validé par le formulaire.
setInputType()	Positionne le type de champ input (text). Choix parmi : <code>button</code> , <code>checkbox</code> , <code>color</code> , <code>date</code> , <code>datetime</code> , <code>datetime-local</code> , <code>email</code> , <code>file</code> , <code>hidden</code> , <code>image</code> , <code>month</code> , <code>number</code> , <code>password</code> , <code>radio</code> , <code>range</code> , <code>reset</code> , <code>search</code> , <code>submit</code> , <code>tel</code> , <code>text</code> , <code>time</code> , <code>url</code> , <code>week</code> . Par défaut <code>text</code> .
setAddon()	Positionne la propriété addon d'un champ de type <code>search</code> . Passer un booléen.
setAclass()	Positionne la classe CSS de l'extension (addon) d'un champ de type <code>search</code> .
setApos()	Détermine la position de l'extension d'un champ <code>search</code> par rapport à son champ de saisie (<code>before</code> / <code>after</code>).
setPostIsTab()	Booléen. Positionne l'information qui informe que le champ génère un POST de type tableau (true) (comme les champs de type <code>file</code> ou <code>select</code> multiple)
setPostName()	Positionne le nom de la variable associée au champ pour le POST du formulaire. Généré automatiquement par la classe. A priori vous ne deviez pas en avoir besoin. Il est même conseillé de ne pas y toucher. Il faut simplement savoir que le nom est construit à partir d'un préfixe propre au type de champ, suivi du nom du champ et de l'id unique du formulaire parent. Ex : <code>memCommentaires_1</code> , <code>txtAnnee_1</code> , etc.
_setId()	Positionne l'Id JavaScript du champ. Méthode privée.
setDesign()	Positionne le design du champ (<code>inline</code> / <code>online</code>)

Classe UniversalForm

setDpos()	Positionne la position du champ dans son groupe (first , inter , last , alone)
setDecalage()	Positionne le décalage du champ sur la droite en colonnes Bootstrap.
setClong()	Positionne la longueur (en colonnes Bootstrap) de la zone de champ.
setCclass()	Positionne la classe CSS du champ.
setLlong()	Positionne la longueur (en colonnes Bootstrap) de la zone de titre.
setLclass()	Positionne la classe CSS du label du champ.
setLabel()	Positionne le label du champ.
setLabelHelp()	Positionne le libellé d'aide qui s'affiche au survol du label du champ.
setLabelHelpPos()	Définit la position du libellé d'aide qui s'affiche au survol du label du champ par rapport au champ (auto , left , top , right , bottom). La valeur auto par laisse l'application décider de la meilleure position en fonction des autres éléments alentour.
setLabelPlus()	Scinde le label en deux partie et positionnel le libellé de droite.
setLabelPlusHelp()	Positionne le libellé d'aide qui s'affiche au survol du libellé de droite du champ scindé.
setLabelPlusHelpPos()	Définit la position par rapport au champ (auto , left , top , right , bottom) du libellé d'aide qui s'affiche au survol du libellé de droite du champ scindé. La valeur auto par laisse l'application décider de la meilleure position en fonction des autres éléments alentour.
setLalign()	Positionne l'alignement du label dans la zone de titre (left , center , right , justify)
setValue()	Positionne la valeur du champ.
setMaxlength()	Positionne la taille maximum du champ de type text .
setJavascript()	Positionnel le code JavaScript associé au champ.
setTitre()	Positionne le titre du champ.
setTitreHelp()	Positionne le libellé d'aire affiché au survol du titre du champ.
setTitreHelpPos()	Définit la position du libellé d'aide qui s'affiche au survol du titre par rapport au titre (auto , left , top , right , bottom). La valeur auto par laisse l'application décider de la meilleure position en fonction des autres éléments alentour.

Classe UniversalForm

setTclass()	Positionne la classe CSS du titre du champ.
setTlong()	Positionne la longueur (en colonnes Bootstrap) du titre.
setLpos()	Détermine la position du label par rapport au champ (<i>before</i> / <i>after</i>)
setErreur()	Positionne le booléen d'erreur sur le champ : <i>true</i> (erreur) / <i>false</i> (pas d'erreur).
setShowErreur()	Positionne le booléen d'affichage ou de masquage des erreurs survenues sur un champ : <i>true</i> (affiche les erreurs) / <i>false</i> (masque les erreurs).
setComplement()	Positionne des informations complémentaires pour le champ.
setSpellcheck()	Positionne l'état spellcheck du champ pour activer la correction automatique (selon paramétrage du navigateur qui a le dernier mot).
setPlaceholder()	Positionne le texte de surimpression d'un champ de saisie.
setNewLine()	Positionne le dessin du champ sur une nouvelle ligne du formulaire.
setFlexLine()	Positionne l'option de positionnement des champs à l'intérieur de la ligne. (flexbox)
setLiberreur()	Positionne le libellé d'erreur affiché sur le champ.
setLiberreurHelp()	Positionne le libellé d'aide affiché sur le champ erreur (pour donner plus d'information sur l'erreur).
setBorder()	Positionne la bordure du champ : <i>true</i> , le champ possède une bordure standard, <i>false</i> pas de bordure, sinon le code CSS de la bordure personnalisée.
setInvisible()	Positionne la visibilité du champ : <i>true</i> , le champ est invisible, <i>false</i> sinon.
setTalign()	Positionne l'alignement du titre dans la zone de titre (<i>left</i> , <i>center</i> , <i>right</i> , <i>justify</i>)
setTestMatches()	Positionne les tests unitaires sur le champ. Préférer de ne pas y toucher directement, hormis par surcharge.
setReadOnly()	Positionne le booléen de lecture seule sur le champ : <i>true</i> / <i>false</i> .
setEnable()	Positionne l'activité du champ : <i>True</i> , le champ est actif, <i>false</i> sinon.
_setIdbchamp()	Positionne l'id JavaScript du « bloc de champ » du champ. Méthode privée.
_setIdztitre()	Positionne l'id JavaScript de la « zone de titre » du champ. Méthode privée.
_setIdzchamp()	Positionne l'id JavaScript de la « zone de champ » du champ. Méthode privée.

Classe UniversalForm

setOperation()

Modifie l'opération en cours dans la gestion du formulaire. Cela peut parfois s'avérer intéressant pour modifier le comportement d'un formulaire.

15. Récapitulatif des méthodes disponibles

Voici un récapitulatif des méthodes qui sont accessibles pour créer vos formulaires :

Méthode	Description
createField()	Crée un nouveau champ pour le formulaire. Usage : <code>\$this->createField(fieldType, idField, infos());</code> Paramètres : fieldType : type de champ idField : identifiant propre au champ infos : tableau de propriétés du champ
afficher()	Affichage du formulaire. Usage : <code>afficher();</code>
doThings()	Méthode (à surcharger) appelée par la méthode <code>tester</code> . L'appel se fait après le POST mais juste avant le test des champs du formulaire. Usage : <code>doThings();</code>
doUltimateThings()	Méthode (à surcharger) appelée juste avant de renvoyer les données à l'utilisateur par l'appel de <code>getData</code> . Usage : <code>doUltimateThings(&\$donnees);</code>
testsSupplementaires()	Méthode (à surcharger) qui gère les tests supplémentaires de la saisie. Usage : <code>testsSupplementaires(\$champ);</code>
testsSupplementairesPosterieurs()	Méthode (à surcharger) qui gère les tests supplémentaires de la saisie après vérification unitaire des champs.
initDonnees()	Méthode (à surcharger) qui initialise les données du formulaire. Usage : <code>initDonnees();</code>

Classe UniversalForm

construitChamps()	<p>Méthode (à surcharger) a qui est confiée la création du formulaire avec ses champs.</p> <p>Usage : <code>construitChamps();</code></p>
getMessage()	<p>Récupère un message.</p> <p>Usage : <code>getMessage();</code></p>
fields()	<p>Renvoie un pointeur sur le tableau collection des champs du formulaire.</p> <p>Usage : <code>\$champs = \$this->fields();</code></p>
field()	<p>Renvoie l'objet champ identifié par son identifiant.</p> <p>Usage : <code>\$objet = \$this->field(identifiant);</code></p>
getFieldObjectByGroupName()	<p>Renvoie le premier objet dont la propriété <code>groupName</code> vaut <code>identifiant</code>.</p> <p>Usage : <code>\$objet = \$this->getFieldObjectByGroupName(identifiant);</code></p>
getAction()	<p>Renvoie l'action à mener (<code>consulter</code> / <code>valid_consulter</code> / <code>ajouter</code> / <code>valid_ajouter</code> / etc.)</p> <p>Usage : <code>\$action = getAction();</code></p>
getOperation()	<p>Renvoie l'opération demandée (<code>consulter</code> / <code>ajouter</code> / <code>modifier</code> / etc)</p> <p>Usage : <code>\$operation = getOperation();</code></p>
getData()	<p>Renvoie les résultats de la saisie du formulaire après le POST.</p> <p>Usage : <code>\$donnees = getData();</code></p>
getIdTravail()	<p>Renvoie l'id de travail (si existe) du tuple de données en cours.</p> <p>Usage : <code>\$id = getIdTravail();</code></p>
setIdTravail()	<p>Positionne l'id de travail</p> <p>Usage : <code>setIdTravail(\$valeur);</code></p> <p>Si doit être utilisé, doit être impérativement appelé avant la méthode <code>construitChamps</code>.</p>

Classe UniversalForm

setOperation()	Positionne l'opération à mener. Usage : <code>setOperation(\$valeur);</code>
setMessage()	Positionne un message à destination de l'application. Usage : <code>setMessage(\$message);</code>
draw()	Dessine le formulaire construit pas <code>construitChamps</code> . Le paramètre <code>enable</code> (booléen) positionne le formulaire en <code>enable</code> (saisie possible) ou <code>disable</code> (saisie impossible). Usage : <code>draw(\$enable);</code>
enableOnChecked()	Active un champ <code>élément</code> selon l'état d'un <code>trigger</code> checkable (comme une <code>checkbox</code>) Usage : <code>enableOnChecked(\$element, \$trigger)</code>
disableOnChecked()	Désactive un champ <code>élément</code> selon l'état d'un <code>trigger</code> checkable (comme une <code>checkbox</code>) Usage : <code>disableOnChecked(\$element, \$trigger)</code>
setInvisibleOnChecked()	Cache ou affiche un champ <code>élément</code> selon l'état d'un <code>trigger</code> checkable (comme une <code>checkbox</code>) Usage : <code>setInvisibleOnChecked(\$element, \$trigger)</code>
tester()	Lance les tests du formulaire. Renvoie un booléen selon le résultat obtenus Usage : <code>\$res = \$frm->tester()</code>

16. Gestion complète d'un formulaire

Nous venons de voir le fonctionnement de base des formulaires gérés par les classes UniversalForm. Vous est proposé ci-dessous le code complet, dit « squelette », issu de **UniversalWeb**, qui peut être utilisé pour chaque formulaire envisagé. En plus de la saisie (ajout de données), ce gestionnaire de formulaire vous montre comment gérer la modification ou la suppression de données d'un formulaire (car jusque-là nous n'avons vu QUE la saisie de nouvelles données).

```
<?php
//-----
// SQUELETTE FORMULAIRE UNIVERSALWEB
//-----
// ééâç : pour sauvegarde du fichier en utf-8
//-----
require_once('libs/common.inc.php');

//creation de l'objet table Application
$table = new table();

//----- H T M L -----
$titrePage = _APP_TITLE_;
$scriptSup = '';
$fjQuery = '';
echo writeHTMLHeader($titrePage, '', '');

echo '<body>';
echo '<div class="container-fluid">';
//-----
// HEADER
//-----
include_once(_BRIQUE_HEADER_);

//-----
// CORPS
//-----
echo '<section>';
echo '<article>';
echo '<div class="container">';

echo '<div class="row">';
echo '<div class="col-12">';

    $frm = new Form_xxxxx($operation, 1);
    $action = $frm->getAction();
    switch($action)    {

        //-----
        // Ajouter
        //-----
        case 'ajouter': {
            $frm->init();
            echo $frm->afficher();
            break;
        }
        //-----
        // Valide ajouter
        //-----
        case 'valid_ajouter': {
            if (!$frm->tester()) {
                echo $frm->afficher();
            }
            else {
                //ok ajouter
                $donnees = $frm->getData();
                //DEBUG_('donnees', $donnees);
                $res = $table->add($donnees);
                if ($res === false) {
                    riseErrorMessage('erreur SQL');
                }
                elseif ($res == 0) {
                    riseWarningMessage('Aucun ajout');
                }
                else {
                    riseMessage('Donnée ajoutée');
                }
                //branchement vers la page d'appel
                goPageBack();
            }
        }
    }
}
```

```
}
break;
}
//-----
// consulter / modifier / supprimer
//-----
case 'consulter':
case 'modifier':
case 'supprimer': {
    //recuperation de l'id de l'item à charger
    (isset($_GET['id'])) ? $id = mysqlDataProtect($_GET['id']) : $id = 0;
    $res = $table->get($id, $tuple);
    if ($res !== false) {
        $frm->charger($id, $tuple);
        echo $frm->afficher();
    }
    break;
}
//-----
// Valide consulter
//-----
case 'valid_consulter': {
    //branchement vers la page d'appel
    goPageBack();
    break;
}
//-----
// Valide modifier
//-----
case 'valid_modifier': {
    if (!$frm->tester()) {
        echo $frm->afficher();
    }
    else {
        $donnees = $frm->getData();
        //DEBUG_('donnees', $donnees);
        //modification effective des données : la clé unique du tuple à modifier*
        //est disponible dans $frm->getIdTravail()
        $res = $table->update($frm->getIdTravail(), $donnees);
        if ($res === false) {
            raiseErrorMessage('erreur sql');
        }
        elseif ($res == 0) {
            raiseWarningMessage('Aucune modification');
        }
        else {
            raiseMessage('Donnée modifiée');
        }
        //branchement vers la page d'appel
        goPageBack();
    }
    break;
}
//-----
// Valide supprimer
//-----
case 'valid_supprimer': {
    //suppression effective des donnees
    $res = $table->delete($frm->getIdTravail());
    if ($res === false) {
        raiseErrorMessage('Erreur SQL');
    }
    elseif ($res == 0) {
        raiseWarningMessage('Aucune suppression');
    }
    else {
        raiseMessage('Donnée supprimée');
    }
    //branchement vers la page d'appel
    goPageBack();
    break;
}
//-----
// Commandes non reconnues
//-----
default: {
    raiseErrorMessage('Erreur commande&hellip;');
    goPageBack();
    break;
}
}

echo '</div>';
echo '</div>';
echo '</div>';

echo '</article>';
```

```
echo '</section>';

//-----
// FOOTER
//-----
include_once(_BRIQUE_FOOTER_);

echo '</div>';
echo '</body>';
echo '</html>';
```

Pour les fonctions « consulter », « modifier » et « supprimer », on notera l'appel à la méthode **charger**. Cette méthode publique qu'il faut implémenter dans la classe de création du formulaire doit *charger* les données à consulter, modifier ou supprimer. D'une manière générale il s'agit d'un tuple de base de données (MySQL par exemple). On n'oubliera d'ailleurs pas de passer en paramètre de la méthode l'identifiant unique de ce tuple. En effet il sera nécessaire plus tard, lors de la validation du formulaire.

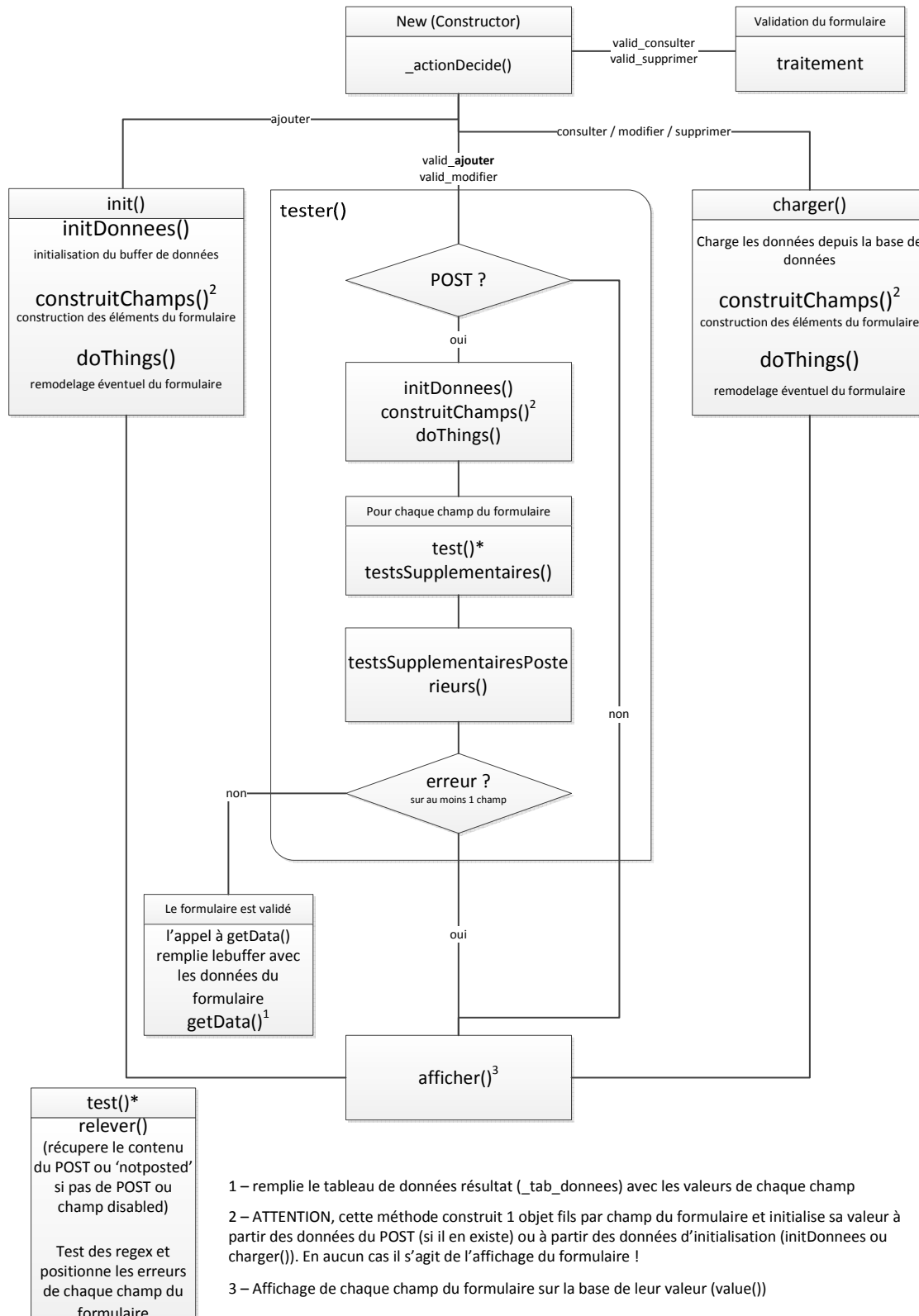
Exemple de méthode **charger**.

```
//charger les données à partir d'un enregistrement (mysql par exemple)
public function charger($id) {
    //sauvegarde de l'identifiant unique du tuple pour pouvoir y revenir
    $this->setIdTravail($id);
    //pour l'exemple, le tuple est issu d'une variable de session (et non d'une base de données)
    $donnees = $_SESSION['donnees'][$id];
    //remplissage du tampon de données du formulaire
    $this->_tab_donnees['titre'] = $donnees['titre'];
    $this->_tab_donnees['annee'] = $donnees['annee'];
    $this->_tab_donnees['visuel'] = $donnees['cnb'];
    //construction des champs en séquence
    $this->construitChamps();
    //Amélioration des champs construits
    $this->doThings();
    return true; //ou false si erreur de chargement données
}
```

A vous de jouer !

17. Annexes

Flux de la gestion des formulaires UniversalForm



Classe UniversalForm

Tableau récapitulatif des propriétés de chaque type de champ du formulaire

	text	radio	checkbox	switch	select	area	bouton	image	search	filtretext	filtreselect	separateur	comment	div	divfin	hidden
newLine	X	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-
flexLine	X	X	X		X	X	X	X	X	X	X	X	X	-	-	-
dbfield	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	X
groupName	-	X	X	-	-	-	-	-	-	-	-	-	-	-	-	-
inputType	X	-	-	-	-	-	X	-	X	X	-	-	-	-	-	-
addon	-	-	-	-	-	-	-	-	X	X	-	-	-	-	-	-
apos	-	-	-	-	-	-	-	-	X	X	-	-	-	-	-	-
aclass	-	-		-					X	X	-					
design	X	X	X	-	X	X	-	X	-	-	X	-	X	-	-	-
multiple	X*	-	-	-	X	-	-	-	-	-	X	-	-	-	-	-
custom	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-	-
accept	X*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
size	-	-	-	-	X	-	-	-	-	-	X	-	-	-	-	-
dpos	-	X	X	-	-	-	-	-	-	-	-	-	-	-	-	-
titre	-	X	X	X	-	-	-	-	X	X	-	-	-	-	-	-
tlong	-	X	X	X	-	-	-	-	X	X	-	-	-	-	-	-
talign	-	X	X	X	-	-	-	-	X	X	-	-	-	-	-	-
tclass	-	X	X	X	-	-	-	-	X	X	-	-	-	-	-	-
titreHelp	-	X	X	X	-	-	-	-	X	X	-	-	-	-	-	-
titreHelpPos	-	X	X	X	-	-	-	-	X	X	-	-	-	-	-	-
decalage	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-
label	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-
labelPlus	X	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-
llong	X	-	-	-	X	X	X	X	X	-	-	-	X	-	-	-
lclass	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-
lalign	X	-	-	X	X	X	-	X	-	X	X	-	X	-	-	-
labelHelp	X	X	X	-	X	X	X	X	X	X	X	X	X	-	-	-
labelPlusHelp	X	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-
labelHelpPos	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-
labelPlusHelpPos	X	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-
lpos	X	X	X	-	X	X	-	X	X	-	X	-	X	-	-	-
clong	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-
rows	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-
cclass	X	X	X	-	X	X	X	X	X	X	X	X	X	X	-	-
border	-	X	X	-	-	-	X	X	-	-	-	X	X	-	-	-
maxlength	X	-	-	-	-	X	-	-	X	X	-	-	-	-	-	-
spellcheck	X	-	-	-	-	X	-	-	X	X	-	-	-	-	-	-
placeholder	X	-	-	-	-	X	-	-	X	X	-	-	-	-	-	-
testMatches	X	-	-	-	X	X	-	-	-	-	X	-	-	-	-	-
value	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	X
valueInverse	-	-	X	X	-	-	-	-	-	-	-	-	-	-	-	-

Classe UniversalForm

	text	radio	checkbox	switch	select	area	bouton	image	search	filtretext	filtreselect	separateur	comment	div	divfin	hidden
checked	-	X	X	X	-	-	-	-	-	-	-	-	-	-	-	-
complement	X*	-	-	-	X	-	-	-	X	X	X	-	-	-	-	-
erreur	X	X	X	X	X	X	X	X	X	X	X	-	X	-	-	-
liberreur	X	X	X	X	X	X	X	X	X	X	X	-	X	-	-	-
liberreurHelp	X	X	X	X	X	X	X	X	X	X	X	-	X	-	-	-
showErreur	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-
javascript	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-
enable	X	X	X	X	X	X	X	-	X	X	X	-	-	-	-	-
readonly	X	X	X	X	X	X	-	-	X	X	X	-	-	-	-	-
Invisible	X	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-

* pour les champs "text" dont le "inputType" est "file"