

Template Engine pour Pro et débutants sous PHP version 4.0.6 ou supérieure

Rubrique	Description
<ul style="list-style-type: none"> ● Présentation <ul style="list-style-type: none"> Principes de base Installation Mini exemples ● Coté PHP <ul style="list-style-type: none"> • Pour commencer <ul style="list-style-type: none"> méthode LoadTemplate() méthode MergeBlock() méthode Show() • Avancé <ul style="list-style-type: none"> méthode CacheAction() méthode GetBlockSource() méthode MergeField() méthode MergeNavigationBar() méthode MergeSpecial() propriété Render propriété Source propriété TplVars ajout d'un type de sources de données Programmation Orientée Objet (POO) ● Coté HTML <ul style="list-style-type: none"> • les champs TBS <ul style="list-style-type: none"> définition et syntaxe paramètres les champs Var les champs Var Spéciaux • les blocs TBS <ul style="list-style-type: none"> définition et syntaxes paramètres sections de bloc affichage en série (en colonnes) requêtes dynamiques / sous-blocs affichage d'une barre de navigation • Autre <ul style="list-style-type: none"> champs et blocs automatiques sous-modèles vue d'ensemble de l'affichage conditionnel ● Résumés <ul style="list-style-type: none"> paramètres des champs TBS paramètres des blocs TBS champs et paramètres de barre de navigation noms de champs et blocs spéciaux 	<p>charge le contenu d'un modèle à partir d'un fichier</p> <p>fusionne une partie du modèle avec une source de données</p> <p>traitements automatiques et affichage du résultat</p> <p>active le système mise en cache des résultats de fusion</p> <p>retourne le source de la définition d'un bloc</p> <p>fusionne un champ particulier avec une valeur</p> <p>fusionne une barre de navigation</p> <p>fusionne les blocs conditionnels, variables Php et autres...</p> <p>pour modifier les options de fin de fusion</p> <p>retourne le contenu courant du résultat de la fusion</p> <p>retourne les variables du modèle</p> <p>pour que TBS reconnaisse un nouveau type de base de données</p> <p>pour que TBS vous facilite la POO.</p>

Présentation :

TinyButStrong (TBS) est une classe PHP utile pour développer une application en séparant proprement vos scripts PHP de vos pages HTML. Avec TBS, les pages HTML sont générées dynamiquement en fusionnant un modèle avec des données. C'est ce qu'on appelle un moteur de modèle (Template Engine).

TBS tient son nom du fait qu'il ne présente que 8 fonctions mais qu'il permet de faire le maximum. Il est **... très très fort ...** pour fusionner des modèles de pages HTML avec vos variables PHP ou vos requêtes MySQL, PostgreSQL, ou SQLite.

TBS a été conçu pour que vous puissiez développer avec facilité vos modèles depuis n'importe quel éditeur HTML visuel (comme Dreamweaver ou FrontPage), mais si vous avez l'habitude d'utiliser un éditeur textuel il est tout aussi pratique. TBS permet aussi de créer du JavaScript dynamiquement.

Comme son nom l'indique, TBS est simple à utiliser, puissant et rapide. Il est complètement **°~° freeware °~°**.

Principes de base :

Du coté HTML :

Vous concevez une page qui n'a pas besoin de contenir de script PHP, ni de programmation. Dans cette page vous placez des balises TBS aux endroits où doivent s'afficher les données dynamiques. Cette page est appelée un 'modèle'. Il existe deux types de balises : les '[champs](#)' qui servent à afficher une donnée dynamiquement, et les '[blocs](#)' qui servent à définir une zone, le plus souvent pour afficher les enregistrements d'une source de données.

Du coté PHP :

Vous utilisez une variable objet TBS pour piloter la fusion de votre modèle HTML avec des données. à la fin, TBS affiche le résultat de la fusion.

Installation :

1. Copiez le fichier **tbs_class.php** dans un répertoire de votre site Web.
2. Au début de votre programme PHP, ajoutez les lignes :

```
include_once('tbs_class.php');  
$TBS = new clsTinyButStrong ;
```

Remarque : si le fichier tbs_class.php se trouve dans un répertoire différent de celui de votre programme, vous devrez préciser le répertoire devant le nom du fichier.

Explications et détails techniques :

TinyButStrong est un librairie écrite en PHP, c'est un composant à référencer dans vos propres programmes PHP. En terme technique, TinyButStrong est une 'classe' d'objet PHP ; le nom de cette classe est clsTinyButStrong. La variable \$TBS que ajoutez en début de programme sert à piloter la fusion de votre modèle depuis votre application PHP. En terme technique, la variable \$TBS est une 'instance' de la classe clsTinyButStrong.

Mini exemples :

Exemple 1 :

Modèle Html

```
<html>  
<body>  
[var.message]  
</body>  
</html>
```

Programme Php

```
<?  
  
include_once('tbs_class.php');  
$TBS = new clsTinyButStrong ;  
$TBS->LoadTemplate('template.htm')  
;  
  
$message = 'Hello' ;  
$TBS->Show() ;  
  
?>
```

Résultat

```
<html>  
<body>  
Hello  
</body>  
</html>
```

Exemple 2 :

Modèle Html

```
<table>
<tr><td>[blk.val;block=tr]</td></tr>
</table>
```

Programme Php

```
<?
include_once('tbs_class.php');
$TBS = new clsTinyButStrong ;
$TBS->LoadTemplate('template.htm')
;

$liste = array('X','Y','Z') ;
$TBS->MergeBlock('blk',$liste) ;
$TBS->Show() ;

?>
```

Résultat

```
<table>
<tr><td>X</td></tr>
<tr><td>Y</td></tr>
<tr><td>Z</td></tr>
</table>
```

Coté PHP :

Le pilotage de la fusion d'un modèle se fait dans un programme PHP en utilisant une variable objet déclaré à partir de la classe `clsTinyButStrong`.

Exemple de déclaration : `$TBS = new clsTinyButStrong ;`

Cet objet vous permet de charger un modèle, piloter sa fusion avec des données, puis afficher le résultat.

Exemple de code PHP :

```
include_once('tbs_class.php');
$TBS = new clsTinyButStrong ;
$TBS->LoadTemplate('template.htm') ;
$TBS->MergeBlock('ctry','mysql','SELECT * FROM t_coutry') ;
$TBS->Show() ;
```

Voici la liste des propriétés et méthodes d'un objet TinyButStrong :

Méthode LoadTemplate() :

Charge un modèle en vue de son traitement pour la fusion.

Le contenu complet du fichier est enregistré dans la propriété [Source](#) de l'objet TinyButStrong.

Syntaxe : `$TBS->LoadTemplate(string Fichier{, string HtmlCharSet})`

Argument	Description
Fichier	Chemin local ou absolu du fichier modèle à charger.
HtmlCharSet	Facultatif. Indique l'encodage des caractères (charset) à utiliser pour la conversion Html des données lorsqu'elles seront fusionnées. La valeur par défaut est " (chaîne vide) qui équivaut à 'ISO-8859-1' (Latin 1). Si votre modèle utilise un charset spécial, indiquez la valeur Html de ce charset. Dans une page Html, le charset se trouve en tête du fichier, dans l'attribut 'content' d'une balise <Meta>. Les charset supportés par TBS sont ceux supportés par la fonction htmlentities() de Php. Par exemple : 'BIG5' (chinois) ou 'EUCJP' (japonais).

Pas de conversion Html :

Si vous utilisez la valeur False pour l'argument **HtmlCharSet**, alors les données ne seront pas converties lors de la fusion avec le modèle.

Fonction utilisateur :

Si votre charset n'est pas supporté par PHP, vous pouvez désigner une fonction utilisateur qui réalise la conversion Html. Pour cela, utilisez l'argument **HtmlCharSet** avec la syntaxe '=myfunction'.

La fonction utilisateur doit prendre une chaîne texte comme argument et retourner une chaîne texte.

Ajouter le fichier à la suite du modèle en cours :

Vous pouvez utiliser le mot-clé '+' à la place du charset pour que le fichier spécifié soit ajouté à la fin du modèle en cours au lieu de l'écraser. Le charset utilisé est alors le même que pour le premier modèle.

Méthode MergeBlock() :

Fusionne un ou plusieurs [blocs TBS](#) avec les données d'une source d'enregistrements. Retourne le numéro du dernier enregistrement affiché (le premier porte le numéro 1).

TinyButStrong supporte plusieurs types de sources de données en natif :

Données Php : un tableau ; une chaîne texte, un nombre.

Base de données : MySQL ; PostgreSQL ; SQLite.

Mais vous pouvez aussi en ajouter de nouveaux : '[ajout d'un type de sources de données](#)'.

Il existe un mode d'affichage 'Par Page' décrit [plus bas](#).

Syntaxe : `int $TBS->MergeBlock(string NomBloc, mixed Source{, string Requête}{, int PageTaille, int PageNum}{, int NbrEnreg})`

Argument	Description
NomBloc	Indique le nom du bloc TBS à fusionner. Vous pouvez fusionner plusieurs blocs avec les mêmes données en indiquant leurs noms séparés par des virgules.
Source	Variable ou mot-clé qui désigne la source de données pour la fusion. Le tableau ci-dessous indique les valeurs possibles selon le type de source de données.
Requête	Facultatif. Indique la requête SQL qui retourne les données à fusionner. Le tableau ci-dessous indique les valeurs possibles selon le type de source de données.
PageTaille	Facultatif. Cet argument doit être défini si vous voulez activer l'affichage par page . Indique le nombre d'enregistrements pour une page.
PageNum	Facultatif. Cet argument doit être défini si vous voulez activer l'affichage par page . Indique le numéro de page à afficher. La première page porte le numéro 1. La valeur spéciale -1 affichera la dernière page du jeu d'enregistrements.
NbrEnreg	Facultatif. Cet argument n'est utile que lors de l'affichage par page . Il permet de modifier le calcul du nombre d'enregistrements retourné par la méthode MergeBlock(). NbrEnreg Valeur retournée par MergeBlock() 0 : C'est la valeur par défaut. La méthode retourne le numéro du dernier enregistrement affiché par la page demandée. -1 : La méthode lie tous les enregistrements jusqu'à la fin et retourne le nombre total d'enregistrements. Toutefois, seuls les enregistrements de la page demandée seront affichés. >0 : La méthode retourne la valeur NbrEnreg . Toutefois, elle retournera le numéro du dernier enregistrement affiché si celui-ci est supérieur à NbrEnreg .

Utilisez ce paramètre pour calculer puis conserver le nombre d'enregistrements total.

Par exemple :

```
if (isset($_POST['nbr_rec'])) {  
    $nbr_rec = $_POST['nbr_rec'] ;  
} else {  
    $nbr_rec = -1 ;  
}  
$nbr_rec = $TBS->MergeBlock('blk1',$cnx_id,'select * from  
t_country',$p_size,$p_num,$nbr_rec);
```

Liaison entre le bloc et les enregistrements :

La méthode MergeBlock() recherche le bloc TBS avec le nom spécifié dans votre modèle. Puis, le bloc TBS est répété autant de fois qu'il y a d'enregistrement dans la source de données.

Pour afficher les données d'un enregistrement, vous devez utiliser un champ TBS lié. Un champ TBS est lié lorsque son nom est composé du nom du bloc, suivi d'un point et du nom d'une colonne ou d'une clé du jeu d'enregistrements. Un champ lié doit se trouver à l'intérieur du bloc.

Exemple :

Nom du bloc : **bloc1**

Colonnes retournées par la requête : **champ1,champ2,champ3**

Champs TBS liés : **[bloc1.champ1], [bloc1.champ2], [bloc1.champ3]**

Si aucune définition de bloc n'est trouvée dans le modèle, La méthode MergeBlock() fusionnera le premier enregistrement avec tous les champs liés trouvés dans le modèle.

Vous pouvez définir des blocs plus évolués. Pour plus d'information, consultez la rubrique [Blocs TBS](#).

Fusionner plusieurs blocs avec les mêmes données :

Vous pouvez fusionner plusieurs blocs avec les mêmes données en indiquant leurs noms séparés par

des virgules dans l'argument **NomBloc**. Dans ce cas, la requête n'est ouverte qu'une seule fois et les enregistrements sont mis dans un cache afin de remplir les blocs.

Exemple:

```
$TBS->MergeBlock('bloc1,bloc2,bloc3','mysql','SELECT * FROM MaTable');
```

Décompte des enregistrements :

Pour afficher le numéro d'enregistrement, utilisez un champ TBS lié à la colonne virtuelle '**#**'.

Si vous placez ce champ en dehors du bloc, il affichera le nombre total d'enregistrements.

Exemple : `[bloc1.#]`

La colonne virtuelle '**\$**' permet d'afficher la clé de l'enregistrement en cours lorsque la source de données est un tableau Php (array).

Exemple: `[bloc1.$]`

Utilisation des arguments Source et Requête selon le type de source de données :

Type de source de données	Source	Requête
Texte (*)	Le mot-clé ' text '	Un texte
Nombre (*)	Le mot-clé ' num '	Un nombre ou un tableau spécial (voir plus bas)
Vide (*)	Le mot-clé ' clear '	-
Conditionnel (*)	Le mot-clé ' cond '	-
Tableau PHP (*)	Un tableau Php	-
	Le mot-clé ' array '	Un tableau Php
	Le mot-clé ' array '	Une chaîne texte qui représente un tableau contenu ou encapsulé dans une variable PHP globale (voir ci-après)
MySQL	Une ressource de connexion MySql ou le mot-clé ' mysql '	Une requête SQL
	Une ressource de résultat MySql	-
PostgreSQL	Une ressource de connexion PostgreSQL	Une requête SQL
	Une ressource de résultat PostgreSQL	-
SQLite	Une ressource de connexion SQLite	Une requête SQLite
	Une ressource de résultat SQLite	-
Personnalisé	Un mot-clé, un objet ou une ressource non listé dans ce tableau. Voir le paragraphe ' ajout d'un type de sources de données '.	Une requête SQL ou autre chose.

(*) Voir explications dans la rubrique ci-après.

Source de données Php :

Texte

L'argument **Source** doit être égale à '**text**'.

Tout le bloc est remplacé par le texte contenu dans la paramètre **Requête**. Les champs liés ne sont pas gérés sauf '**#**' qui retourne 1, ou 0 si **Requête** est une chaîne vide.

Nombre

L'argument **Source** doit être égale à '**num**'.

L'argument **Requête** peut être soit un nombre, soit un tableau.

arg Requête Jeu d'enregistrements retourné

Nombre : Ce nombre doit être supérieur ou égale à zéro. Le jeu d'enregistrement retourné est composé d'une colonne '**val**' dont la valeur va de 1 à ce nombre.

Tableau : Ce tableau doit contenir une clé '**min**' et une clé '**max**' et éventuellement une clé '**step**'. Le jeu d'enregistrement retourné est composé d'une colonne '**val**' qui va de la valeur de '**min**' à la valeur de '**max**'.

Exemple : `array('min'=>101,'max'=>150)` affichera 50 blocs numérotés de 101 à 150.

Vide

L'argument **Source** doit être le mot-clé 'clear'.

Tous les blocs et leurs sections sont supprimés. C'est la même chose que fusionner avec un tableau vide.

Conditionnel

L'argument **Source** doit être le mot-clé 'cond'.

Le bloc est fusionné comme si c'était un [bloc conditionnel](#) **onload** et **onshow**. Le bloc n'est pas fusionné avec des données et donc il ne doit pas avoir de champ TBS lié. Chaque section de bloc doit avoir un paramètre **when** ou un paramètre **default**. Voir [bloc conditionnel](#) pour plus de détails.

Tableau

L'argument **Source** doit être un tableau Php (type Array), ou le mot-clé 'array'. Si le mot-clé 'array' est utilisé, alors l'argument **Requête** doit être un tableau Php ou une chaîne texte qui représente tableau contenu ou encapsulé par une variable Php globale.

Syntaxe de la chaîne : 'globvar[item1][item2]...'

'globvar' est le nom d'un variable global \$globvar qui doit être un tableau.

'item1' et 'item2' sont des clés d'un item ou d'un sous item de \$globvar.

Exemple:

```
$TBS->MergeBlock('bloc1','array','jours[lun]');
```

Cela va fusionner 'block1' avec \$jours['lun'] qui est supposé être un tableau.

Il est possible de représenter un nom de variable sans item.

Exemple:

```
$TBS->MergeBlock('bloc1','array','jours');
```

Il y a deux avantages à utiliser une chaîne pour représenter le tableau :

-> Les items seront lus directement dans le tableau (assigné par référence) au lieu de lire une copie des items. Cela peut améliorer les performances.

-> Vous pouvez utiliser des requêtes dynamiques.

Afficher la clé de l'enregistrement en cours :

Vous pouvez utiliser la colonne virtuelle '\$' qui affiche la clé de l'enregistrement en cours. Cela peut être particulièrement utile pour les [requêtes dynamiques et sous-blocs](#).

Exemple: `[bloc1.$]`

Structure des tableaux supportés :

Les items du tableau spécifié peuvent être de deux types : des valeurs simples avec des clés associées (cas 1), ou des valeurs tableaux dont les items sont eux-mêmes des valeurs simples avec des clés associées (cas 2).

Cas 1 :

Exemple : `['clé1']=>valeur1`

`['clé2']=>valeur2`

...

Le jeu d'enregistrement retourné est composé d'une colonne 'key' contenant le nom de la clé, et d'une colonne 'val' contenant la valeur de la clé.

Cas 2 :

Exemple : `[0] => (['colonne1']=>valeur1-0 ; ['colonne2']=>valeur2-0 ; ...)`

`[1] => (['colonne1']=>valeur1-1 ; ['colonne2']=>valeur2-1 ; ...)`

`[2] => (['colonne1']=>valeur1-2 ; ['colonne2']=>valeur2-2 ; ...)`

...

Le jeu d'enregistrement retourné est composé des colonnes 'colonne1', 'colonne2', ... avec leurs valeurs associées.

Mode 'Par Page' :

Le mode affichage 'Par Page' est activé lorsque vous renseignez l'argument **PageTaille** avec une valeur différente de zéro. L'affichage des données sera alors limité à la page spécifiée par **PageNum**. Si **PageNum** a la valeur -1, c'est la dernière page qui sera affichée.

Remarque importante :

Bien qu'il soit facile et pratique, le mode 'Par Page' n'est pas optimisé pour un grand nombre d'enregistrements. Si votre affichage est trop lent, ou si votre base de données est trop sollicitée, il est conseillé d'utiliser les fonctions natives de votre système de base de données (si celui-ci gère les requêtes limitées).

Par exemple avec MySQL, utilisez la clause LIMIT.

Explications : compte tenu de la diversité des syntaxes SQL, TinyButStrong n'est pas capable de modifier lui-même une requête afin qu'elle retourne un jeu d'enregistrement limité. Par exemple, il n'est pas capable d'ajouter la clause LIMIT à une requête MySQL. C'est pourquoi, TinyButStrong doit appeler la requête originale, puis lire les enregistrements un par un en ignorant tous ceux qui se trouvent avant la page demandée. Cela a pour effet de rendre l'affichage d'autant plus lent que la page à atteindre est élevée. Une fois que la page atteinte, TinyButStrong libère la requête sans aller à la fin du jeu d'enregistrement.

Méthode Show() :

Termine la fusion.

Syntaxe : `$TBS->Show({int Render})`

La méthode Show exécutera les opérations suivantes :

- fusion des champs Var,
- fusion des champs [onshow],
- affiche le résultat (peut être annulé par la propriété Render),
- termine le script (peut être annulé par la propriété Render).

La propriété [Render](#) permet de régler le comportement de la méthode Show().

Le paramètre *Render* permet lui aussi d'ajuster le comportement de la méthode Show() mais seulement pour un appel.

Méthode CacheAction() :

Lance une action du Système de cache de TinyButStrong. Le Système de cache permet de gérer manuellement ou automatiquement la sauvegarde du résultat de la fusion dans un fichier temporaire appelé fichier "cache". La méthode CacheAction() retourne *True* ou *False* selon que l'action est réussie ou non.

Syntaxe : `bool $TBS->CacheAction(string CacheId {, int Action/MaxAge}{, string Dir})`

<u>Argument</u>	<u>Description</u>
<i>CacheId</i>	Identifiant unique du fichier cache. Ce doit être une chaîne texte, elle sera utilisée dans le nom du fichier.
<i>Action/MaxAge</i>	Détermine l'action à effectuer. Ce doit être une constante prédéfinie ou une valeur positive. Voir le tableau ci-après pour plus de détail sur les actions possibles. La valeur par défaut est 3600 , ce qui correspond à une sauvegarde automatique avec un age maxi d'1 heure.
<i>Dir</i>	Facultatif. Désigne le répertoire où est enregistré le fichier cache. Par défaut il s'agit du répertoire du script.

Gérer un fichier cache :

Le Système de cache vous permet créer, charger, mettre à jour ou supprimer un fichier cache identifié par *CacheId*. Il existe aussi un Mode automatique qui laisse le Système gérer ces actions en fonction d'un age maxi du fichier cache.

Cache à l'affichage :

Si vous déclenchez le "cache à l'affichage", le résultat de la fusion sera automatiquement enregistré dans le fichier cache à la première utilisation de la méthode [Show\(\)](#). L'enregistrement a lieu après affichage du résultat.

Notez que par défaut la méthode Show() provoque aussi la fin du script. Si vous souhaitez continuer les traitements après le "cache à l'affichage", vous devez donc paramétrer la propriété [Render](#) de manière à empêcher la fin du script.

Voici la liste des actions possibles pour l'argument *Action/MaxAge*, ce peut être une constante prédéfinie par TinyButStrong ou valeur numérique positive.

<u>Action</u>	<u>Description</u>
<i>x</i> >=0 (numérique positif)	Mode automatique avec age-maxi : <ul style="list-style-type: none">- Si le fichier cache existe et qu'il a été créé il y a moins de <i>x</i> secondes, alors le fichier est chargé, puis la méthode Show() est exécutée automatiquement. Si vous n'avez pas modifié la propriété Render alors le script s'arrête après

	l'affichage du cache, sinon CacheAction() retourne <i>true</i> . - Si le fichier cache n'existe pas ou si il a été crée il y a plus de <i>x</i> secondes, alors le "cache à l'affichage" est déclanchée (voir plus haut). Le script continue normalement et CacheAction() retourne <i>false</i> .
TBS_CACHENOW	Sauvegarde le résultat actuel de la fusion dans le cache correspondant à <i>CacheId</i> . CacheAction() retourne <i>false</i> .
TBS_CACHELOAD	Charge le fichier cache correspondant à <i>CacheId</i> . CacheAction() retourne <i>true</i> si le fichier cache a été trouvé et chargé, sinon il retourne <i>false</i> .
TBS_DELETE	Supprime le fichier cache correspondant à <i>CacheId</i> , si il existe. Vous pouvez supprimer tous les fichiers cache du répertoire en utilisant '*' pour <i>CacheId</i> . CacheAction() retourne <i>false</i> .
TBS_CACHEONSHOW	Active le "cache à l'affichage" pour le cache correspondant à <i>CacheId</i> . CacheAction() retourne <i>false</i> .
TBS_CANCEL	Annule le "cache à l'affichage" quel que soit <i>CacheId</i> . CacheAction() retourne <i>false</i> .
TBS_CACHEGETAGE	Retourne l'age du fichier cache en exprimé secondes. Retourne <i>false</i> si le fichier cache n'existe pas.
TBS_CACHEGETNAME	Retourne le nom du fichier cache correspondant au <i>CacheId</i> renseigné.
TBS_CACHEISONSHOW	Retourne <i>true</i> si le "Cache à l'affichage" est activé, sinon retourne <i>false</i> .

Méthode GetBlockSource() :

Retourne la source d'un bloc TBS.

Seule la définition de la première section du bloc sera retournée à moins que l'argument *Sections* soit à *True*.

Si aucun bloc n'est trouvé, la méthode retourne la valeur *False*.

Syntaxe : *string* \$TBS->GetBlockSource(*string* NomBloc {, *boolean* Sections})

<u>Argument</u>	<u>Description</u>
<i>NomBloc</i>	Nom du bloc à rechercher.
<i>Sections</i>	Facultatif. La valeur par défaut est <i>False</i> . Si ce paramètre est à <i>True</i> , la méthode retourne un tableau contenant les définitions de toutes les sections du bloc nommé. La première section est retournée dans l'élément [1] du tableau.

Cette méthode permet de récupérer la source d'un bloc afin de gérer manuellement sa fusion. Si par la suite vous souhaitez remplacer le bloc par du texte, vous pouvez utiliser ma méthode [MergeBlock\(\)](#) avec le paramètre 'text'.

Méthode MergeField() :

Fusionne un ou plusieurs champs TBS avec une valeur fixe ou en appelant une fonction utilisateur. Tous les champs du modèle ayant le nom de base indiqué seront fusionnés.

Syntaxe : \$TBS->MergeField(*string* NomBase, *mixed* X {, *boolean* ModeFonction})

<u>Argument</u>	<u>Description</u>
<i>NomBase</i>	Le nom de base des champs TBS. Par exemple 'compte'.
<i>X</i>	La valeur à afficher, ou une chaîne représentant le nom d'une fonction utilisateur.
<i>ModeFonction</i>	Indique que la valeur à afficher est calculée par une fonction utilisateur. La valeur par défaut est <i>false</i> . Si cet argument est à <i>true</i> , alors <i>X</i> doit être une chaîne texte donnant le nom de la fonction utilisateur. Cette fonction doit exister et avoir la syntaxe décrite ci-après.

Fusion avec une valeur :

X peut être un numérique, une chaîne, un tableau ou un objet. Pour un tableau ou un objet, les noms de champs TBS doivent avoir des suffixes comme pour les [champs Var](#).

Exemple :

```
$TBS->MergeField('compte', array('id'=>55, 'nom'=>'Bob')) ;
```

Dans cet exemple, les champs [compte.id] et [compte.nom] seront fusionnés.

Fusion avec une fonction utilisateur :

TBS appelle une fonction utilisateur pour chaque champ trouvé dans le modèle.

Cette fonction doit avoir la syntaxe suivante:

```
function fct_utilisateur($Subname [, $PrmLst]) {...}
```

Lors de l'appel à la fonction, son argument **\$Subname** a pour valeur le suffixe du nom du champ (par exemple pour un champ nommé 'ml.titre', **\$Subname** aura pour valeur 'titre'). Et l'argument optionnel **\$PrmLst** contient un tableau associatif avec les paramètres du champ. La fonction doit retourner la valeur à fusionner.

Exemple :

```
$TBS->MergeField('ml', 'm_multilangue', true);
...
function m_multilangue($Subname) {
    global $langue_id;
    $rs = mysql_query("SELECT texte_$langue_id AS txt FROM t_langue WHERE cle='$Subname'");
    $rec = mysql_fetch_array($rs);
    return $rec['txt'] ;
}
```

Dans cet exemple, un champ tel que [ml.titre] sera fusionné avec la valeur retournée par m_multilangue('titre').

Méthode MergeNavigationBar() :

Affiche une barre de navigation à partir de blocs TBS et de champs TBS spécifiques.

Pour plus d'info sur la construction d'une barre de navigation, consultez la rubrique '[Affichage d'une barre de navigation](#)'.

Syntaxe : **\$TBS->MergeNavigationBar(string NomNav, mix Options, int PageNum[, int NbrEnreg, int TaillePage])**

<u>Argument</u>	<u>Description</u>										
NomNav	Nom de la barre de navigation. Remarque : vous pouvez fusionner plusieurs barres de navigation en une seule fois et avec les mêmes options en séparant leurs noms par une virgule.										
Options	Permet de forcer des options de la barres de navigation. Ces options peuvent aussi être définies avec des paramètres de bloc dans le modèle. Mais si vous paramétrez en plus dans la méthode MergeNavigationBar(), ces options seront forcées. Ce paramètre peut être vide (0, " ou null), une valeur numérique ou un tableau. Si c'est une valeur numérique, elle indique le nombre de page à afficher. Si c'est un tableau, il peut contenir les items suivants : <table><tr><th><u>Clé</u></th><th><u>Valeur</u></th></tr><tr><td>'navsize'</td><td>Nombre de page pages affichées dans la barre de navigation. (par défaut = 10).</td></tr><tr><td>'navpos'</td><td>Position de la barre de navigation par rapport au numéro de la page courante. Utilisez un des mots-clés suivants : - 'step' (par défaut) pour avoir une barre de progression pas à pas. - 'centred' pour centrer la barre sur le numéro de page courante.</td></tr><tr><td>'navdel'</td><td>Nom d'un bloc TBS à supprimer lorsqu'il n' y a qu'une seule page ou aucune page à afficher. Ce bloc TBS doit entourer la barre de navigation. Si il y plusieurs pages à afficher alors seulement les balises TBS de définition de ce bloc seront supprimées.</td></tr><tr><td>'pagemin'</td><td>Numéro de la première page (par défaut = 1).</td></tr></table>	<u>Clé</u>	<u>Valeur</u>	'navsize'	Nombre de page pages affichées dans la barre de navigation. (par défaut = 10).	'navpos'	Position de la barre de navigation par rapport au numéro de la page courante. Utilisez un des mots-clés suivants : - 'step' (par défaut) pour avoir une barre de progression pas à pas. - 'centred' pour centrer la barre sur le numéro de page courante.	'navdel'	Nom d'un bloc TBS à supprimer lorsqu'il n' y a qu'une seule page ou aucune page à afficher. Ce bloc TBS doit entourer la barre de navigation. Si il y plusieurs pages à afficher alors seulement les balises TBS de définition de ce bloc seront supprimées.	'pagemin'	Numéro de la première page (par défaut = 1).
<u>Clé</u>	<u>Valeur</u>										
'navsize'	Nombre de page pages affichées dans la barre de navigation. (par défaut = 10).										
'navpos'	Position de la barre de navigation par rapport au numéro de la page courante. Utilisez un des mots-clés suivants : - 'step' (par défaut) pour avoir une barre de progression pas à pas. - 'centred' pour centrer la barre sur le numéro de page courante.										
'navdel'	Nom d'un bloc TBS à supprimer lorsqu'il n' y a qu'une seule page ou aucune page à afficher. Ce bloc TBS doit entourer la barre de navigation. Si il y plusieurs pages à afficher alors seulement les balises TBS de définition de ce bloc seront supprimées.										
'pagemin'	Numéro de la première page (par défaut = 1).										
PageNum	Numéro de la page courante. La première page a le numéro 1. Pour indiquer la dernière page, utilisez la valeur -1.										
NbrEnreg	Facultatif. La valeur par défaut est -1. Indique le nombre d'enregistrements total connus. Si ce nombre est inconnu il faut mettre la valeur -1. Cet argument sert uniquement à calculer le numéro de la dernière page de la										

barre de navigation.

TaillePage Facultatif. La valeur par défaut est **1**.
Indique le nombre d'enregistrement par page. Il s'utilise conjointement avec **NbrEnreg** et sert uniquement à calculer le numéro de la dernière page de la barre de navigation.

Exemple :

```
$TBS->MergeNavigationBar('nav','', $page, $rec_nbr, $page_size);
```

Méthode MergeSpecial() :

Remplace les champs et blocs spéciaux du type spécifié.

Syntaxe : `$TBS->MergeSpecial(string Type)`

L'argument **Type** doit être l'une des valeurs suivantes :

<u>Valeur</u>	<u>Description</u>
'var'	Remplace tous les champs Var .
'onload'	Remplace tous les champs onload .
'onshow'	Remplace tous les champs onshow .

Remarque :

Par défaut, la méthode [Show\(\)](#) remplace tous les champs et blocs spéciaux juste avant l'affichage du résultat de la fusion. C'est pour cela qu'il est rare d'utiliser MergeSpecial() dans un programme.

Propriété Render :

Détermine comment doit se terminer la fusion.
Sa valeur doit être une combinaison des constantes du tableau ci-dessous.
Par défaut, sa valeur est (**TBS_OUTPUT** + **TBS_EXIT**).

Syntaxe : `int $TBS->Render`

La propriété Render influe sur le comportement des méthodes [Show\(\)](#) et [CacheAction\(\)](#).

<u>Constante</u>	<u>Description</u>
TBS_NOTHING	Indique que aucune action ci-dessous n'est effectuée à la fin de la fusion.
TBS_OUTPUT	Indique que le résultat de la fusion doit être affiché. (utilisation de la commande PHP Echo)
TBS_EXIT	Indique qu'on doit quitter le script juste après la fin de la fusion.

Propriété Source :

Définie ou renvoie le code HTML sur lequel on applique les opérations de fusion.
Après l'appel à la méthode [LoadTemplate\(\)](#), cette propriété contient le source HTML du modèle demandé.
Cette propriété permet de lire ou modifier le résultat de la fusion, au fur et à mesure des traitements.

Syntaxe : `string $TBS->Source`

Propriété TplVars :

Contient le tableau des variables de modèle correspondant au modèle en cours.

Syntaxe : `array $TBS->TplVars`

Vous pouvez définir des variables de modèle en utilisant un ou des [champs automatiques onload](#) avec le paramètre **tplvars**.

Tous les autres paramètres qui suivent le paramètre **tplvars** sont ajoutés à la propriété TplVars quand la méthode LoadTemplate() est appelée.

Exemple :

```
[onload;tplvars;template_version='1.12.27';template_date='2004-10-26']
```

Cette balise TBS va créer deux items équivalents au code PHP suivant :

```
$TBS->TplVars['template_version'] = '1.12.27';
```

```
$TBS->TplVars['template_date'] = '2004-10-26';
```

Remarques :

- Le paramètre **tplvars** ne fonctionne que avec les [champs automatiques onload](#).
- Vous pouvez utiliser le paramètre **tplvars** plusieurs fois dans le même modèle.

Ajout d'un type de source de données :

Il est possible d'ajouter un autre type de source de données non encore supporté en natif par TinyButStrong.

Pour cela, vous devez coder trois fonctions (ou méthodes) avec des déclarations spécifiques et des noms correspondant au type à ajouter.

N'ajouter pas ces fonctions dans le fichier source de TBS, codez les dans votre programme ou dans un script Php externe.

Vous pouvez trouver des nouveaux types de sources de données sur le [site](#) Web de TinyButStrong.

Vous avez le choix entre coder des fonctions utilisateur ou coder des méthodes d'une classe.

Si vous codez des fonctions utilisateurs, elles devront avoir des noms qui utilisent un identifiant TBS propre à la source de donnée utilisée (voir ci-après).

Si vous codez des méthodes d'une classe, ces méthodes devront être nommées **tbsdb_open**, **tbsdb_fetch** et **tbsdb_close**.

Exemple :

```
class clsTest {  
    function tbsdb_open(...) {...}  
    function tbsdb_fetch(...) {...}  
    function tbsdb_close(...) {...}  
}
```

Identifiant TBS (pour les fonctions utilisateur uniquement) :

L'argument **\$Source** que vous passez à la méthode [MergeBlock\(\)](#) a un identifiant TBS que vous devez utiliser pour nommer vos fonctions.

Si **\$Source** est un objet, alors l'identifiant TBS est le nom de la classe de cet objet.

Si **\$Source** est une ressource, alors l'identifiant TBS est le type de la ressource.

Si **\$Source** est une chaîne texte, alors l'identifiant TBS est cette même chaîne.

Le type de l'argument **\$Source** ne doit pas être déjà supporté en natif par TinyButStrong, sinon les fonctions seront ignorées.

L'identifiant TBS peut être arrangé par TBS pour convenir à un nom de fonction.

Exemple :

Si **\$Source** est une connexion Sybase (type de ressource = **'sybase-db link'**), alors l'identifiant TBS est **'sysbase_db'**.

Déclarations des fonctions ou méthodes :

Les trois fonctions ou méthodes à ajouter dans votre programme doivent avoir les syntaxes suivantes :

Si vous codez des fonctions utilisateur, remplacez le mot-clé **'customdb'** par l'identifiant TBS identifier de votre type de source de données. Si vous codez des méthodes, remplacez les noms de fonctions par **tbsdb_open**, **tbsdb_fetch** et **tbsdb_close**.

```
function tbsdb_customdb_open(&$Source,&$Requete) {...}
```

Cette fonction doit ouvrir la requête demandée, et retourner un identifiant de jeu d'enregistrements.

En cas d'erreur, la fonction doit retourner la valeur **False**, et peut afficher un message.

Argument	Description
----------	-------------

\$Source	Il s'agit du même argument passé à la méthode MergeBlock().
-----------------	---

\$Requête	Il s'agit du même argument passé à la méthode MergeBlock().
------------------	---

Exemple :

```
function tbsdb_sybase_db_open(&$Source,&$Query) {  
    return sybase_query($Query,$Source) ;  
}
```

```
function tbsdb_customdb_fetch(&$Rs{, $RecNum}) {...}
```

Cette fonction doit retourner un tableau associatif correspondant à l'enregistrement en cours, avec les noms de colonnes et les valeurs. La fonction doit retourner la valeur **False** quand il n'y a plus d'enregistrement.

Argument	Description
\$Rs	L'identifiant de jeu d'enregistrement retourné par la fonction tbsdb_customdb_open() .
\$RecNum	Facultatif. Le numéro de l'enregistrement attendu. Le premier porte le numéro 1.

Exemple :

```
function tbsdb_sybase_db_fetch(&$Rs) {
    return sybase_fetch_assoc($Rs) ;
}
```

Si la source de données a besoin du numéro de l'enregistrement demandé, vous pouvez ajouter l'argument \$RecNum à la déclaration de la fonction. Mais dans les autres cas, cet argument est facultatif parce que tous les enregistrements seront appelés dans l'ordre de toute façon.

```
function tbsdb_customdb_close(&$Rs) {...}
```

Cette fonction doit fermer ou libérer l'identifiant de jeu d'enregistrements. Elle n'a pas besoin de retourner une valeur.

Argument	Description
\$Rs	L'identifiant de jeu d'enregistrement retourné par la fonction tbsdb_customdb_open() .

Exemple :

```
function tbsdb_sybase_db_close(&$Rs) {
    return sybase_free_result($Rs) ;
}
```

Programmation Orientée Objet :

Si vous êtes un développeur POO, vous préféreriez sans doute que TBS travaille avec des méthodes plutôt que des fonctions globales, et des propriétés plutôt que des variables globales.

Pour réaliser cela, vous devez au préalable référencer la propriété ObjectRef directement ou indirectement sur un objet existant. Exemple :

```
$TBS->ObjectRef = &$MonObjet; // Utilisez '&' pour définir la propriété par référence.
ou :
$TBS->ObjectRef['item1'] = &$MonObjet; // Référence indirecte
```

Puis il suffit de préfixer le nom d'une fonction par '~' dans toutes les fonctionnalités TBS pour désigner une méthode de ObjectRef au lieu d'une fonction globale. Les méthodes codées dans la classe doivent avoir la même syntaxe que les fonctions globales correspondantes à la fonctionnalité.

Les champs Var peuvent se référer aux méthodes de l'objet ObjectRef, mais aussi à ses propriétés.

Les fonctionnalités qui supportent le préfixe '~' sont listées ci-dessous :

Fonctionnalités	Exemple
Champs Var	[var.~prop1] ... [var.~item1.prop1] ... [var.~meth1] ... [var.~prop2.sousprop]
Paramètre ondata	[blk1.colonne1;block=tr;onsection=~meth_ondt]
Paramètre onformat	[blk1.colonne2;onformat=~meth_onfrm]
Méthode MergeField()	\$TBS->MergeField('nomchamp','~meth_MrgFld',true);
Fonctions perso de lecture de données (*)	\$TBS->MergeBlock('blk1','~madb','SELECT * FROM t_table');
Fonctions perso de conversion Html	\$TBS->LoadTemplate('monmodele.htm','=~meth_htmlconv');

(*) Trois méthodes, au lieu de trois fonctions, doivent être définies pour le mot-clé donné.

Par exemple pour le mot-clé '~madb', les méthodes doivent être nommées mydb_open(), mydb_fetch() et mydb_close().

Exemple (valable pour ezSQL) :

```
class madb Extends db {
    function madb_open(&$source,&$query) {
        $this->get_results($query) ;
        return $this ;
    }
    function madb_fetch(&$db,$num) {
        $x = $this->get_row(null,ARRAY_A,$num-1) ;
        if (is_array($x)) {
            return $x ;
        } else {
            return false ;
        }
    }
}
```

```
function madb_close(&$db) {
    // pas besoin
}
```

Coté HTML :

Vous concevez votre modèle en plaçant des balises TBS aux endroits où doivent figurer les données.

Il existe deux types de balises TBS : les *champs* et les *blocs*.

Un champ TBS est une balise TBS qui doit être remplacé par une donnée simple. Il est possible de spécifier un format d'affichage ainsi que d'autres paramètres. La syntaxe des champs TBS est décrite [ci-après](#).

Un bloc TBS est une région qui devra être répétée. Il est défini par une ou deux balises TBS. Le plus souvent il s'agit d'une ligne d'un tableau HTML. La syntaxe des blocs TBS est décrite [ci-après](#).

Les champs TBS :

Un champ TBS est une balise TBS qui doit être remplacé par une donnée simple. Il a un nom qui permet de l'identifier et on peut définir des paramètres pour modifier le comportement de l'affichage.

Syntaxe : **HTML ... [NomChamp;params] ... HTML**

<u>élément</u>	<u>Description</u>
NomChamp	Le nom du champ. Attention : les noms de champs commençant par var. , onload et onshow sont réservés pour les champs Var , les champs automatiques .
params	Facultatif. Un ou plusieurs paramètres de la liste ci-dessous, séparés par des ';'. Certains paramètres peuvent être affectés d'une valeur en utilisant le caractère '='. Exemple : frm=0.00 Si la valeur du paramètre contient des espaces, des points-virgules ou des guillemets, alors vous pouvez utiliser les guillemets simples comme délimiteurs. Exemple : frm='0 000.00' Utilisez deux guillemets simples pour définir un caractère guillemet dans une chaîne délimitée. Exemple: ifempty='coucou c'est moi' Il est possible d'imbriquer des champs TBS les uns dans les autres. Cela veut dire que vous pouvez écrire ceci : [var.v1; if [var.v2]=1]. Mais : - pour les champs Var , vous devez vous assurer que v2 sera fusionné avant v1. - pour les champs de blocs , vous devez vous assurer que la colonne v2 est avant la colonne v1.

Paramètres des champs :

<u>Paramètre</u>	<u>Description</u>
htmlconv=val	Permet de forcer ou empêcher la conversion de la donnée en texte Html. La valeur val peut être l'un des mots-clés suivants : <ul style="list-style-type: none"> yes : (valeur par défaut) Force la conversion en Html avec sauts de ligne. no : Empêche la conversion en Html. Utile pour modifier du code Javascript ou modifier le source HTML. nobr : Force la conversion en Html sans les sauts de ligne (utile pour la balise <pre> par exemple). wsp : Préserve les espaces blancs (utile pour les espaces de début de ligne). esc : Pas de conversion Html et double les caractères guillemets simples (''). js : Convertie la donnée en une chaîne qui peut être insérée entre des délimiteur texte JavaScript look : Convertie la donnée en Html si aucune balise Html n'est trouvée dans cette donnée. <p>Vous pouvez spécifier plusieurs mots-clés en utilisant le séparateur '+'. Exemple :</p>

htmlconv=yes+js

. (point)

Si la donnée est vide, on affiche un espace Html insécable. Utile pour les cellules d'un tableau.

ifempty=val

Si la donnée est vide, on la remplace par la valeur indiquée.

magnet=tag

Assigne une balise Html magnétique à un champ TBS. Une balise magnétique est gardée telle quelle quand le champ à une valeur, et elle est supprimée quand le champ est null ou chaîne vide.

Exemple :

([cliquez ici]([var.lien;magnet=a]))

Résultat pour \$lien='www.tbs.com': ([cliquez ici](www.tbs.com))

Résultat pour \$lien='': ()

Par défaut, la balise Html magnétique doit être un couple de balises ouvrante-fermante (comme) dont la première balise est placée avant le champ TBS. Mais cela peut être changé en utilisant le paramètre mtype (voir ci-après).

Remarque : les paramètres if then else sont traités avant le paramètre magnet.

mtype=val

S'utilise avec la paramètre magnet. Définit le comportement de la balise magnétique.

Valeur Comportement magnétique lorsque le champ est nul ou chaîne vide

m*m C'est la valeur par défaut. Supprime le couple de balises qui entoure le champ TBS. Tout ce qui se trouve entre ces balises est aussi supprimé. Il est possible de placer le champ TBS à l'intérieur de l'une des balises.

Exemple :

([cliquez ici]([var.lien;magnet=a]))

Résultat pour \$lien='www.tbs.com': ([cliquez ici](www.tbs.com))

Résultat pour \$lien='': ()

m+m Supprime le couple de balises qui entoure le champ TBS, mais conserve ce qui est entre les balises.

Exemple :

([\[blk.nom\]](mailto:[blk.email;magnet=a;mtype=m+m]))

Résultat pour \$email='moi@tbs.com': ([MonNom](mailto:moi@tbs.com))

Résultat pour \$email='': (MyName)

m* Supprime la balise simple qui se trouve avant le champ TBS, ainsi que tout de qui se trouve entre la balise et le champ.

Exemple 1: 

Exemple 2:
 [var.address;magnet=br]

***m** Supprime la balise simple qui se trouve après le champ TBS, ainsi que tout de qui se trouve entre la balise et le champ.

Exemple : [var.address;magnet=br;mtype=*m]

selected

Ce paramètre sert à sélectionner un item dans une liste, des radio-boutons ou des cases-à-cocher positionnés dans un formulaire Html. Il faut s'assurer au préalable que les items soient déjà créés (fusionnés).

Liste Html :

Utilisez le paramètre select sans lui donner de valeur. Le champ TBS doit obligatoirement être placée parmi la liste des items. Quand le champ TBS est fusionné, il sera supprimé, mais l'item qui a la même valeur que le champ sera sélectionné. Si la valeur n'est pas trouvé, un nouvel item est ajouté.

Exemple :

Paris
Toulouse
Brest

[ville_id;selected] Ce qui donnera après fusion :

Paris
Toulouse
Brest

Radio-boutons et cases-à-cocher :

Utilisez le paramètre select en lui donnant comme valeur le nom des radio-bouton ou cases-à-cocher à traiter. Le champ TBS doit obligatoirement être placé à l'intérieur du formulaire. Quand le champ TBS est fusionné, il sera supprimé mais l'item qui a la même valeur que le champ sera sélectionné.

Exemple :

☐ Paris [ville_id;selected=r_test] Ce qui donnera après fusion : ☒ Paris

☐ Toulouse
☐ Brest

☒ Toulouse
☐ Brest

Dans cet exemple, le radio-bouton intitulé 'Toulouse' a été sélectionné parce que le nom de la balise radio-bouton a le nom 'r_test' et sa valeur 2, et que le champ TBS nommé 'ville_id' a été fusionné avec la valeur 2.

Multi-sélection :

Pour les listes, les radios-boutons ou les cases-à-cocher, vous pouvez faire une multi-sélection donnant à la valeur du champ TBS un tableau Php.

selbounds=tag

S'utilise avec le paramètre **selected**. Cela permet de changer la zone de recherche des items à sélectionner en désignant un type de balise Html. Par défaut, cette valeur vaut **select** pour une liste, et **form** pour des cases-à-cocher ou radio-boutons.

Exemple : [ville_id;**selected=r_test;selbounds=div**]

Dans cet exemple les items seront recherchés entre les balises `<div>` et `</div>` qui encadrent le champ TBS.

comm

Ce paramètre permet d'étendre les limites du champ jusqu'aux limites de la balise commentaire (Html) qui l'entoure.

`<!-- [monchamp;comm] ceci est un exemple-->` est rigoureusement identique à `[monchamp]`

C'est particulièrement pratique pour l'élaboration du modèle avec un éditeur HTML visuel (tel que Dreamweaver ou FrontPage).

noerr

Empêche l'affichage de certains messages d'erreurs TBS. Quand un message peut être annulé, cela est mentionné dans le message.

file=nomfichier

Remplace le champ par le contenu du fichier. **Nomfichier** peut être une chaîne fixe ou une expression composée de **champs Var** qui retourne le chemin du fichier. L'utilisation de ce paramètre est détaillée dans la rubrique [Sous-modèles](#).

script=nomfichier

Exécute le script PHP juste avant le remplacement du champ.

Nomfichier peut être une chaîne fixe ou une expression composée de **champs Var** qui retourne le chemin du fichier.

* Tenez compte du fait que dans votre script **les variables ne sont pas globales mais locales**. Cela est dû à ce que le script est appelé depuis une méthode de TBS. Pour atteindre ou définir une variable globale dans votre script, vous devez utiliser l'instruction Php [global](#) ou le tableau `$GLOBAL`.

* TBS met à disposition des variables locales prédéfinies que vous pouvez utiliser dans votre script :

- **\$CurrVal** se réfère à la valeur du champ en cours. Elle peut être modifiée.

- **\$CurrPrm** se réfère au tableau de paramètre du champ en cours.

- **\$this** se réfère à l'instance en cours de TBS. (Voir le paramètre **subtpl** pour un bon usage)

* Le paramètre **script** est sensible au paramètre **if**. Si le champ possède un paramètre **if**, alors le script n'est exécuté que si la condition est vérifiée.

See chapter '[Subtemplates](#)' for more details about how to use this parameter in subtemplate mode.

subtpl

S'utilise avec le paramètre **script** ou **onformat**.

Active le mode sous-modèle pendant l'exécution du script ou de la fonction.

Voir le chapitre [Sous-modèles](#) pour plus d'information.

once

S'utilise avec le paramètre **script**.

Annule l'exécution du script s'il a déjà été appelé auparavant.

getob

Ce paramètre est obsolète parce qu'il peut être remplacé par le paramètre **subtpl**. S'utilise avec le paramètre **script**. Indique que le texte affiché par les commandes `echo()` du script Php remplacent la valeur du champ TBS.

if exp1=exp2

Affiche la donnée si la condition est vérifiée, sinon n'affiche rien à moins que les paramètres **then** ou **else** soient utilisés.

Les opérateurs supportés sont :

= ou ==	égale
!=	différent
+-	supérieur strictement
+=-	supérieur ou égal
-+	inférieur strictement

- = + inférieur ou égal

exp1 et **exp2** doivent être des expressions numériques ou textes. Vous pouvez utiliser le mot-clé **[val]** dans les expressions pour représenter la valeur du champ. Les expressions peuvent contenir des champs TBS, mais vous devez vous assurer qu'ils soient fusionnés avant le champ contenant. Voir paramètres **then** et **else** pour des exemples.

then val1 Si le paramètre **if** a été défini et que sa condition est vérifiée, alors la donnée sera remplacée par **val1**.

Exemple :

```
[var.image;if [val]="";then 'image0.gif']
```

else val2 Si le paramètre **if** a été défini et que sa condition n'est pas vérifiée, alors la donnée sera remplacée par **val2**.

Exemple :

```
[var.error_id;if [val]=0;then 'pas d''erreur';else 'erreur constatée']
```

onformat=nom_fct Indique le nom d'une fonction PHP utilisateur qui sera exécutée avant la fusion du champ. La fonction **nom_fct** doit avoir la syntaxe suivante :

```
function nom_fct ($NomChamp, &$CurrVal, {&$CurrPrm, {&$TBS}}) { ... }
```

Argument	Description
----------	-------------

\$NomChamp	Retourne le nom du champ qui appelle la fonction (lecture seule).
-------------------	---

\$CurrVal	Retourne la valeur en cours (lecture/écriture ; ne pas oublier le & dans la déclaration de la variable).
------------------	--

\$CurrPrm	Facultatif. Fait référence au tableau des paramètres du champ courant (ne pas oublier le & dans la déclaration de la variable).
------------------	---

\$TBS	Facultatif. Fait référence à l'instance courante de TBS. (ne pas oublier le & dans la déclaration de la variable). Utilisez cet argument avec prudence. Il est fourni pour le mode Sous-modèle.
--------------	---

Voir le paragraphe '[Sous-modèles](#)' pour plus de détails sur l'utilisation de ces arguments en mode sous-modèle.

protect=val Permet de protéger ou non la donnée à fusionner en remplaçant les caractères '[' pour leur équivalent Html '['. La valeur **val** peut être l'un des mots-clés suivants :

yes : (valeur par défaut) la donnée est protégée.

no : la donnée n'est pas protégée.

Par défaut, toutes données fusionnées avec un modèle sont protégées sauf s'il s'agit de l'inclusion d'un autre fichier. Il est fortement recommandé de protéger les valeurs affichées lorsque qu'il s'agit de données saisies librement comme sur un forum par exemple.

max=val Indique le nombre maximum de caractères à afficher. Au delà de cette limite, la donnée est tronquée, et des points de suspensions "..." sont ajoutés à la fin.

frm=format Spécifie un format d'affichage pour une donnée de type date/heure ou numérique. Pour un numérique, il est possible d'utiliser un format conditionnel qui change selon le signe de la valeur.

Format date/heure :

Il s'agit d'un format semblable au format VisualBasic. Les mots-clés suivants sont reconnus :

d, dd, ddd, dddd : numéro du jour, numéro du jour sur deux chiffres, nom du jour court, nom du jour complet. Utilisez le paramètre **locale** pour afficher des noms locaux.

xx affiche **st, nd, rd** ou **th** selon le numéro du jour.

m, mm, mmm, mmmm : numéro du mois, numéros du mois sur deux chiffres, nom du mois court, nom du mois complet. Utilisez le paramètre **locale** pour afficher des noms locaux.

yy, yyyy : année sur deux chiffres, années complètes.

hh, nn, ss : heure, minutes, seconde sur deux chiffres.

Les autres caractères sont conservés.

Il est possible de mettre de protéger des chaînes de texte en les plaçant entre guillemets simples ou double.

Exemples :

```
[chp;frm=dd/mm/yyyy] affichera 21/12/2002
```


[**chp**;frm='yyyy-mm-dd hh:nn:ss'] affichera 2002-12-21 15:45:03

Format numérique :

Pour définir la partie décimale, utilisez une expression du type '0x0...' où 'x' est le séparateur de décimal, et '0...' est une répétition de zéro correspondant au nombre de décimales.

S'il n'y a aucune décimale, utilisez le format '0.' (avec un point).

Pour définir un séparateur de milliers, utilisez une expression du type '0z000x...' où 'z' est le séparateur de milliers. S'il n'y a aucune décimale, utilisez le format '0z000.' (avec un point).

Si le format contient le caractère '%', alors la valeur affichée sera multipliée par 100. Le caractère '%' reste affiché.

Le format numérique peut contenir d'autres chaînes de texte. Mais seule l'expression de zéro placée la plus à droite sera considérée comme un format, les autres caractères seront conservés.

Exemples :

<u>Valeur</u>	<u>Champ</u>	<u>Affichage</u>
2456,1426	[chp ;frm='0,000']	2456,143
	[chp ;frm='\$ 0 000,00']	\$ 2 456,14
	[chp ;frm='\$ 0 000.']	2 456
0,2537	[chp ;frm='0,00 %']	25,37%
	[chp ;frm='coef 0,00']	coef 0,25

Formats conditionnels :

Il est possible de définir jusqu'à 4 formats conditionnels selon que la valeur est respectivement positive, négative, zéro ou nulle (ou chaîne vide). Les formats conditionnels doivent être séparés par un caractère '|'. Chaque format conditionnel est facultatif.

Exemples :

<u>Valeur</u>	<u>Champ</u>	<u>Affichage</u>
2456,1426	[chp ;frm='+0,00 -(0,00) * vide']	+2456,14
-156,333	[chp ;frm='+0,00 -(0,00) * vide']	-(156,33)
0	[chp ;frm='+0,00 -(0,00) * vide']	*
null	[chp ;frm='+0,00 -(0,00) * vide']	vide
-8,75	[chp ;frm='+0,00 -(0,00)']	-(8,75)

locale

S'utilise avec le paramètre **frm**.

Indique que le format spécifié par **frm** doit afficher des noms de jours et de mois locaux.

Les informations de localisation peuvent être modifiées avec la fonction PHP [setlocale\(\)](#).

tplvars

Permet de définir des variables dans le modèle que vous pouvez récupérer dans le programme PHP en utilisant la propriété [TplVars](#). Ce paramètre ne fonctionne que avec des [champs automatiques](#) **onload**.

Les champs Var :

Un champ de variable PHP est un champ qui affiche une variable PHP.

Son nom doit être composé du mot-clé '**var.**' suivi du nom de la variable PHP.

Les paramètres de champs standards sont valables pour les champs de variables PHP.

Par exemple [**var.php_version**] sera remplacé par "4.2.3". Les variables utilisateurs ainsi que les variables prédéfinies peuvent être fusionnées mais elles doivent être globales. Les variables de type **Ressource** sont ignorées.

Il est possible de fusionner une variable **Tableau** en indiquant une clé du tableau à l'aide d'un point.

Par exemple : [**var.montableau.item**]

Il est possible de fusionner une variable **Objet** en indiquant une propriété (ou une méthode qui n'a besoin

d'aucun argument) à l'aide d'un point.
Par exemple : [**var.monobjet.propriete**]

Champs Var imbriqués

Les champs Var imbriqués dans des paramètres **file**, **script**, **if** et **when** (mais aussi **then** et **else** depuis TinyButStrong version 2.02) sont systématiquement traités. Dans les autres cas, il faut s'assurer que le champ englobant soit fusionné avant le champ imbriqué, sinon le champ Var imbriqué est ignoré.

Quand sont fusionnés les champs Var ?

Les champs Var sont fusionnés à l'appel de la méthode [Show\(\)](#), c'est à dire juste avant l'affichage du résultat de la fusion. Mais vous pouvez forcer la fusion à tout moment avec la méthode [MergeSpecial\(\)](#).

Sécurité : comment limiter l'utilisation des champs Var dans les modèles ?

Vous pouvez limiter l'utilisation des champs Var en définissant un préfixe de variable autorisé lors de la création de l'objet TinyButStrong.

Exemple :

```
$TBS = new clsTinyButStrong('', 'x1_');
```

Dans cet exemple, seules les variables Php globales préfixées par 'x1_' sont autorisées dans le modèle. Les autres champs Var produiront un message d'erreur explicite au moment de leur fusion.

[**var.x1_title**] sera fusionné si la variable globale **\$x1_title** existe.

[**var.x2_title**] produira un message d'erreur explicite.

NB: le premier paramètre '' de `clsTinyButStrong()` dans cet exemple est utilisé pour définir les délimiteurs de balises TBS. Mais cela n'est pas décrit dans ce manuel.

Les champs Var Spéciaux :

Un champ Var Spécial est un champ TBS qui affiche des données fournies par le système TinyButStrong. Le nom d'un champ Var Spécial doit être un de la liste ci-dessous.

Les paramètres de champs TBS standards sont valables pour les champs Var Spéciaux.

Exemple : **Date du jour** : [**var..now**;frm='dd/mm/yyyy']

<u>Nom</u>	<u>Description</u>
var..now	Date et heure du serveur.
var..version	La version de TinyButStrong.
var..script_name	Le nom du fichier PHP en cours d'exécution.
var..template_name	Le nom du dernier fichier modèle chargé. Il s'agit du nom tel que indiqué lors de l'appel à la méthode LoadTemplate() .
var..template_date	La date de création du dernier fichier modèle chargé.
var..template_path	Le répertoire du dernier fichier modèle chargé. Il s'agit du répertoire tel que indiqué lors de l'appel à la méthode LoadTemplate() .
var..tplvars.*	La valeur d'un item défini dans la propriété TplVars . ('*' doit être la clé d'un item existant dans le tableau)

Quand sont fusionnés les champs Var Spéciaux ?

Les champs Var Spéciaux sont fusionnés à l'appel de la méthode [Show\(\)](#), c'est à dire juste avant l'affichage du résultat de la fusion. Mais vous pouvez forcer la fusion à tout moment avec la méthode [MergeSpecial\(\)](#).

Les blocs TBS :

Un bloc TBS permet d'afficher les données d'une source d'enregistrements.

La fusion entre un bloc et des données est réalisée grâce à la méthode [MergeBlock\(\)](#).

Lors de la fusion, le bloc TBS est répété autant de fois qu'il y a d'enregistrements ; et les champs TBS associés sont remplacés par les valeurs des colonnes.

Un champ fusion associé au bloc **Bloc1** et qui affiche la valeur de la colonne **ColonneA** doit être nommé **Bloc1.ColonneA**

Exemple : [**Bloc1.ColonneA**;frm='dd-mm-yyyy']

Deux blocs portant le même nom seront considérés comme deux sections du même bloc (voir [sections de bloc](#)).

Syntaxes des blocs :

Il existe trois syntaxes possibles pour définir un bloc TBS :

Syntaxe explicite :

On utilise deux balises TBS. L'une pour le début du bloc, l'autre pour la fin du bloc.

Exemple :

HTML...[NomBloc;block=begin;params]...**HTML...**[NomBloc;block=end]...**HTML**

Les balises TBS de définition du bloc seront supprimées lors de la fusion.

Syntaxe relative :

Le bloc est défini par un couple de balises Html ouvrante-fermante. Il suffit alors d'une seule balise TBS.

Exemple :

HTML...<nom_balise...>...[NomBloc;block=nom_balise;params]...**</nom_balise...>...HTML**

La balise TBS de définition du bloc doit se trouver n'importe où entre le couple de balises Html.

Cette balise TBS sera supprimée lors de la fusion.

Syntaxe simplifiée :

On utilise un champ TBS associé pour définir le bloc de façon relative (voir syntaxe relative ci-dessus).

Exemple :

HTML...<nom_balise...>...[NomBloc.NomColonne;block=nom_balise;params]...**</nom_balise...>.**

Le champ TBS qui contient la définition du bloc doit se trouver entre le couple de balises Html.

Mais ce n'est pas obligatoirement le premier champ TBS du bloc.

<u>élément</u>	<u>Description</u>
NomBloc	Est le nom du bloc.
block=begin	Désigne le début du bloc.
block=end	Désigne la fin du bloc.
block=nom_balise	Désigne un bloc compris entre la balise Html ouvrante <nom_balise...> et la balise fermante </nom_balise...> qui encadrent la balise TBS. Les balises Html ouvrantes et fermantes font partie du bloc. <ul style="list-style-type: none">- row peut être utilisé comme alias pour désigner la ligne d'un tableau. block=row équivaut à block=tr.- opt peut être utilisé comme alias pour désigner un item d'un liste Html. block=opt équivaut à block=option.
params	Facultatif. Un ou plusieurs paramètres de la liste ci-dessous. Séparés par des ';'.

Quelle syntaxe utiliser ?

La syntaxe 'explicite' est rarement utilisée avec des éditeurs visuels parce que les balises TBS doivent souvent être placés entre deux balises Html. Par contre, elle convient assez bien pour des éditeurs textuels.

La syntaxe 'relative' permet de désigner un bloc avec seulement une balise TBS. De plus, on pas besoin de cacher la balise TBS car elle sera supprimée lors de l'affichage. Cette syntaxe est assez pratique.

La syntaxe 'simplifiée' est réellement simple. Elle permet de définir un bloc TBS et un champ TBS avec une seule balise TBS. Cette syntaxe est la plus courante et la plus pratique.

Astuce :

Vous pouvez utiliser la syntaxe 'relative' ou 'simplifiée' avec des balises personnelles ayant la norme Html.

Exemple :

<balise_perso>Bonjour [**blk1.colonne1;block=balise_perso**], comment allez vous ?**</balise_perso>**

Paramètres des blocs :

<u>Paramètre</u>	<u>Description</u>
extend=n extend=tag1,tag2,...	Étend la définition du bloc sur plus de balises. Ce paramètre n'a d'effet qu'avec la syntaxe relative ou la syntaxe simplifiée des blocs. Il permet par exemple de définir un bloc sur deux lignes d'une table.

Syntaxe 1 : avec un nombre **n** (positif ou négatif)

Étend la définition du bloc sur les **n** paires de balises qui suivent.

Les noms de balises sont les mêmes que celles du paramètre **block**.

Si **n** est négatif, alors le bloc est étendu sur les balises qui précèdent.

Syntaxe 2 : avec une liste de noms de balises

Étend la définition du bloc sur les paires de balises qui suivent.

Les balises sont celles données dans la liste.

encaps=num

Indique le niveau d'encapsulation de la balise TBS par rapport aux balises Html spécifiées par le paramètre **block**. Par défaut cette valeur est à **1**.

Exemple :

[bloc1.champ1;block=tr;encaps=2]	[bloc1.champ2]
----------------------------------	----------------

Dans l'exemple ci-dessus, la ligne bleu sera dupliquée lors de la fusion car on a 'encaps=2'.

Si on met 'encaps=1' ou si on retire le paramètre, ce sera la ligne rose qui sera dupliquée lors de la fusion.

comm

Ce paramètre permet d'étendre les limites de la balise TBS jusqu'aux limites de la balise commentaire Html qui l'entoure.

`<!-- [bloc1;block=tr;comm] ceci est un exemple-->` est rigoureusement identique à `[bloc1;block=tr]`

Ce paramètre facilite l'élaboration du modèle avec un éditeur HTML visuel (tel que Dreamweaver ou FrontPage).

nodata

Désigne une section qui ne s'affiche que s'il n'y a aucune donnée à fusionner.

Exemple :

[bloc1.champ1;block=tr]	[bloc1.champ2]
[bloc1;block=tr;nodata]Il n'y a aucune donnée.	

Pour plus d'information sur les sections, voir le paragraphe '[sections de bloc](#)'.

headergrp=colnom

Désigne une section d'entête qui sera affichée chaque fois que la colonne **colnom** change de valeur. **colnom** doit être un nom de colonne valide retourné par la source de données.

Vous pouvez définir plusieurs sections **headergrp** sur des colonnes différentes.

L'ordre de placement des sections **headergrp** dans un bloc peut changer le résultat.

Pour plus d'information sur les sections, voir le paragraphe '[sections de bloc](#)'.

footergrp=colnom

Désigne une section de pied qui sera affichée chaque fois que la colonne **colnom** change de valeur. See **headergrp**.

splittergrp=colnom

Désigne une section de séparation qui sera affichée chaque fois que la colonne **colnom** change de valeur. See **headergrp**.

parentgrp=colnom

Désigne une section parent qui sera affichée chaque fois que la colonne **colnom** change de valeur. Contrairement aux autres sections, une section **parentgrp** peut contenir des sections normales. C'est un moyen de définir un entête et pied de groupe avec une seule section.

serial

Indique que le bloc est un bloc principal qui contient une série de blocs secondaires.

Pour plus d'information, voir le paragraphe '[Affichage en série \(en colonne\)](#)'.

p1=val1

Signal l'utilisation d'une requête dynamique. Toutes les occurrences de la chaîne '%p1%' trouvées dans la requête passée à la méthode MergeBlock() sont remplacées par la valeur **val1**.

Pour plus d'information, voir le paragraphe '[Requêtes dynamiques / sous blocs](#)'.

ondata=nom_fct

Indique le nom d'une fonction Php utilisateur qui sera exécutée pendant la fusion du bloc. Cette fonction est appelée chaque fois qu'un enregistrement est récupéré dans la source de données. Vous pouvez utiliser les arguments d'une telle fonction Php pour modifier les enregistrements avant qu'ils ne soient fusionnés. La fonction doit avoir la syntaxe suivante:

```
function nom_fct($NomBloc, &$Enreg, $NumEnreg) { ... }
```

Argument	Description
----------	-------------

\$NomBloc Returns the name of the block calling the function (read only).

\$Enreg Returns an associative PHP array containing the current record (read/write ; *don't forget the & in the function header*). If you set this variable to **False**, it ends the merging like it was the end of the record set.

\$NumEnreg Returns the number of the current record (read only, first record is number 1).

Exemples :

```
function f_ajout_colonne($NomBloc, &$Enreg, $NumEnreg) {
    $Enreg['taille'] = strlen($Enreg['texte']);
}
```

onsection=nom_fct

Utilisez ce paramètre avec précaution car il ne sera peut être plus supporté dans le futur. Il est conservé dans le but d'assurer la compatibilité, mais vous devriez utiliser à la place le paramètre **ondata** qui est plus rapide.

Le paramètre **onsection** fonctionne de la même manière que **ondata**, sauf que la fonction Php est appelée à chaque fois qu'un section est fusionnée. Ainsi donc, si votre bloc contient des sections d'entête ou des sections conditionnelles, alors la fonction pourrait être appelée plusieurs fois pour un même enregistrement.

Syntaxe de la fonction Php :

```
function nom_fct($NomBloc, &$Enreg, &$DetSrc, $NumEnreg) { ... }
```

L'argument **\$DetSrc** retourne le source de la section courante (en lecture/écriture ; *n'oubliez pas le & dans la déclaration de la fonction*). Si vous passez cette variable à "", cela annule l'affichage de l'enregistrement.

when expr1=expr2

S'utilise uniquement avec les [blocs conditionnels](#). Affiche le bloc que si la condition est vérifiée.

Les opérateurs supportés sont :

= ou ==	égale
!=	différent
+ -	supérieur strictement
+ = -	supérieur ou égal
- +	inférieur strictement
- = +	inférieur ou égal

exp1 et **exp2** doivent être des expressions numériques ou textes. Les expressions peuvent contenir des [champs Var](#).

default

S'utilise uniquement avec les [blocs conditionnels](#). Indique un bloc conditionnel qui ne doit s'afficher que si aucune des autres sections du même bloc (nommé) n'a été affiché.

several

S'utilise uniquement avec les [blocs conditionnels](#). Indique que plusieurs sections du bloc (nommé) peuvent être affichées si plusieurs conditions sont vérifiées. Par défaut, les sections sont exclusives.

Sections de bloc :

Différents blocs portant le même nom seront considérés comme des sections du même bloc.

Les sections peuvent servir pour :

- alterner la présentation (sections normales),
- afficher quelque chose si il n'y a aucun enregistrement (section NoData),
- afficher un entête à chaque changement de la valeur d'une colonne (sections de regroupement).

Sections normales :

Quand vous définissez plusieurs sections normales, elles seront utilisées de façons alternatives à chaque enregistrement.

Exemple :

```
[b1.libelle; block=tr]
```

```
[b1.libelle; block=tr]
```

Dans cet exemple, le bloc nommé 'b1' contient deux sections normales. Les enregistrements seront affichés alternativement avec un fond vert puis un fond bleu.

Section NoData :

Affiche la section si la source de données de contient aucun enregistrement.

La section NoData est définie en ajoutant le paramètre **nodata**.

Exemple :

```
[b1.libelle;block=tr]
Il n'y a rien. [b1;block=tr;nodata]
```

Sections de regroupement :

Les sections de regroupement s'affichent chaque fois que la valeur d'une colonne du jeu d'enregistrement change. Vous pouvez définir des sections d'entêtes, de pieds-de-groupe, de séparation et des sections parent à l'aide des paramètres **headergrp**, **footergrp**, **splittergrp** et **parentgrp**. Voir les [paramètres de bloc](#) pour plus de détail.

Exemple :

```
Année :[b1.annee;block=tr;headergrp=annee]
[b1.libelle] [b1.montant;block=tr]
```

Sections conditionnelles :

Les sections conditionnelles ne s'affichent que lorsque leur condition est vérifiée. La condition d'affichage est définie avec le paramètre **when**. Dès qu'une section a ce paramètre, elle devient conditionnelle. Voir le chapitre sur l'[affichage conditionnel](#) pour plus de détail.

Exemple :

```
[b1.nom;block=tr]
[b1.adresse;block=tr;when [b1.adr_ok]=1]
```

Affichage en série (en colonnes) :

L'affichage en série permet d'afficher plusieurs enregistrements dans un même bloc. Pour cela, on utilise un bloc principal et des blocs secondaires.

Exemple :

Enr 1	Enr 2	Enr 3	Enr 4
Enr 5	Enr 6	Enr 7	Enr 8
Enr 9

Dans cet exemple, les blocs principaux sont les lignes bleues du tableau, les blocs secondaires sont les cases roses.

Syntaxe :

Le bloc principal et ses blocs secondaires sont fusionnés à l'aide d'un seul appel à la méthode MergeBlock(). Le bloc principal doit être défini en utilisant le paramètre **serial**. Les blocs secondaires doivent être inclus dans le bloc principal. Leur nom de bloc doit être celui du bloc principal suivi de "_" puis du numéro d'ordre d'affichage.

Exemple :

```
[bx;block=tr;serial][bx_1.txt;block=td] [bx_2.txt;block=td] [bx_3.txt;block=td] [bx_4.txt;block=td]
```

Le code PHP correspondant est :

```
$TBS->MergeBlock('bx', $cnx_id, 'SELECT txt FROM t_info ORDER BY txt')
```

Bloc secondaire vide :

Vous pouvez désigner un bloc secondaire spécial qui sera utilisé en remplacement des blocs secondaires inexploités (sans enregistrement). Ce bloc secondaire "vide" doit avoir l'indice 0. Il peut être placé dans un bloc principal avec les blocs secondaires normaux, ou alors seul dans autre bloc **serial**. Le bloc secondaire "vide" est facultatif.

Exemple :

[bx;block=tr;serial][bx_1.txt;block=td]	[bx_2.txt;block=td]	[bx_3.txt;block=td]	[bx_4.txt;block=td]
[bx;block=tr;serial][bx_0;block=td] Vide			

Remarque :

L'affichage en série marche aussi avec les [sections de bloc](#) et les [requêtes dynamiques](#).

Requêtes dynamiques / sous-blocs :

Principe des requêtes dynamiques :

Il est possible d'utiliser la méthode MergeBlock() avec une requête dynamique.

Dans votre modèle, vous devez définir un bloc en ajoutant les paramètres **p1**, **p2**, **p3**,... avec leurs valeurs.

La requête passée à la méthode MergeBlock() doit contenir des marqueurs **%p1%**, **%p2%**, **%p3%**, ... pour accueillir les valeurs des paramètres **p1**, **p2**, **p3**,... .

Chaque section du bloc à fusionner contenant un paramètre **p1** sera traité comme un bloc à part pour lequel on ré-exécute la requête dynamique. Les sections du bloc qui n'ont pas de paramètre **p1** sont rattachées à la section avec paramètre **p1** qui précède.

Exemple :

Pays : France

[blk.ville;block=tr;p1='france']	[blk.pays]
----------------------------------	------------

Pays : USA

[blk.ville;block=tr;p1='us']	[blk.pays]
------------------------------	------------

Code PHP correspondant :

```
$TBS->MergeBlock('blk',$cnx_id,"SELECT ville,pays FROM t_geo WHERE (pays='%p1%')")
```

Résultat de la fusion :

Pays : France

Paris	france
Toulouse	france

Pays : USA

Washington	us
Boston	us

Utilisation avec des sous-blocs :

Les requêtes dynamiques vous permettent de réaliser simplement un système de bloc-principal / sous-blocs. Voici comment vous pouvez faire :

- Créez un bloc principal, puis un sous-bloc contenu dans le bloc principal.
- Liez-les en ajoutant au sous-bloc un paramètre **p1** qui prend pour valeur un champ du bloc principal.
- Du côté PHP, fusionnez d'abord le bloc principal, puis le sous-bloc.

Exemple :

Pays : [pri.pays;block=table]
[sub.ville;block=tr;p1=[pri.pays_id]]

Code PHP correspondant :

```
$TBS->MergeBlock('pri',$cnx_id,'SELECT pays,pays_id FROM t_pays')  
$TBS->MergeBlock('sub',$cnx_id,'SELECT ville FROM t_ville WHERE (pays_id=%p1%)')
```

Résultat de la fusion :

Pays : France
Paris
Toulouse
Pays : Allemagne
Berlin
Munique
Pays : Espagne
Madrid
Barcelone

Remarques :

- Le paramètre `htmlconv=esc` permet de passer à la requête des valeurs chaînes protégées.
- Les requêtes dynamiques marchent aussi avec les [sections de bloc](#) et l'[affichage en série](#).

Affichage d'une barre de navigation :

TinyButStrong est capable d'afficher une barre de navigation à partir de blocs spécifiques.
La barre est fusionnée grâce à la méthode [MergeNavigationBar\(\)](#).

Exemple :

|< < [nav.page;block=td] [nav.page;block=td;currpage] > >|

Code Php utilisé :

```
$TBS->MergeNavigationBar('nav',10,17) ;
```

Résultat de la fusion:

|< < 11 12 13 14 15 16 17 18 19 20 > >|

Remarque : cet exemple n'affiche pas de lien.

Voici les éléments que vous pouvez utiliser pour construire votre barre de navigation :

- Utilisez un bloc TBS normal pour afficher les numéros de pages normales.
- Utilisez un autre bloc TBS du même nom mais avec le paramètre `currpage` pour afficher la page courante de façon différente.
- Utilisez les champs suivants où vous voulez (dans un bloc ou hors d'un bloc) :

Nom du champ	Description
<code>page</code>	Retourne le numéro d'une page normale accessible depuis la barre de navigation. Ce champ doit être contenu dans un bloc.
<code>curr</code>	Retourne le numéro de la page courante.
<code>first</code>	Retourne le numéro de la première page (1).
<code>prev</code>	Retourne le numéro de la page précédente.
<code>next</code>	Retourne le numéro de la page suivante.
<code>last</code>	Retourne le numéro de la dernière page si il est connu, sinon, retourne -1.

(Tous les éléments doivent être préfixés par le nom du navigateur suivi d'un point. Comme pour un bloc normal.)

Astuce :

Si vous utilisez le paramètre `endpoint` sur ces champs, alors lorsque la page courante est sur la première ou la dernière page, les champs correspondants retourneront une chaîne vide (") au lieu d'un numéro de page. Cela permet de gérer des exceptions d'affichage avec la propriété [magnet](#) par exemple.

Exemple :

```
<a href="script.php?page=[nav.first;endpoint;magnet=a;mttype=m+m]">Début</a>
```

Dans cet exemple, le lien sera supprimé lorsque la page courante est la première page.

Options

La définition du bloc peut contenir des paramètres spécifiques à la barre de navigation.

Ces options peuvent aussi être définies en tant que paramètres de la méthode [MergeNavigationBar\(\)](#).

Paramètre	Description
<code>navsize=num</code>	Nombre de page pages affichées dans la barre de navigation. (par défaut = 10).
<code>navpos=motclé</code>	Position de la barre de navigation par rapport au numéro de la page courante. Utilisez un des mots-clés suivants : - 'step' (par défaut) pour avoir une barre de progression pas à pas. - 'centred' pour centrer la barre sur le numéro de page courante.
<code>navdel=nombloc</code>	Nom d'un bloc TBS à supprimer lorsqu'il n'y a qu'une seule page ou aucune page à afficher. Ce bloc TBS doit entourer la barre de navigation. Si il y plusieurs pages à afficher alors seulement les balises TBS de définition de ce bloc seront supprimées.
<code>pagemin=num</code>	Numéro de la première page (par défaut = 1).

Champs et blocs automatiques :

`onload` et `onshow` sont des noms réservés pour les champs et de blocs TBS qui sont fusionnés automatiquement quand le modèle est chargé par la méthode `LoadTemplate()` et quand le résultat est affiché par la méthode `Show()`.

Les champs automatiques sont fusionnés avec une valeur vide. Ils acceptent tous les paramètres de champs TBS.

Ils sont pratiques pour les [sous-modèles](#) ou les [variables de modèle](#).

Exemple :

```
[onload;file=entete.htm]
```

Les blocs automatiques ne sont pas fusionnés avec des données. Il ne peuvent avoir que des [sections conditionnelles](#).

Exemples :

```
[onload;block=tr;when [var.statut]==1] Statut 1
```

```
[onload;block=tr;when [var.statut]==2] Statut 2
```

Voir les [sections conditionnelles](#) pour plus d'information.

Sous-modèles :

Il y a deux façons pour insérer des sous-modèles dans votre modèle principal.

Insertion brute avec le paramètre `file` :

C'est la meilleure façon pour insérer une partie se trouvant dans un autre fichier, comme cela est habituellement fait pour les entêtes et pieds-de-page.

La valeur donnée au paramètre `file` doit être le nom d'un fichier existant sur le serveur. Vous pouvez utiliser une expression avec des champs Var et le mot-clé `[val]` qui représente la valeur du champ.

Exemples :

```
[onload;file=entete.htm]
```

```
[onload;file=[var.fichier_entete]]
```

```
[var.sub1;file=[val]]
```

Le contenu du fichier est inséré à la place du champ, sans [conversion Html](#) et sans [protection TBS](#). Les balises `[onload]` contenues dans le fichier ne sont pas traitées au moment de l'insertion. Les balises `[onshow]` et les [champs Var](#) seront fusionnés sur la méthode `Show()` parce qu'il sont devenus partie intégrante du modèle principal.

Le sous-modèle peut contenir des champs TBS, y compris des champs Var et des blocs à fusionner. Si vous avez l'intention de fusionner des données avec un bloc défini dans un sous-modèle, alors il est conseillé d'utiliser le paramètre `file` dans un champ `[onload]` pour s'assurer que le sous-modèle soit inséré avant que vous appeliez `MergeBlock()`.

Le contenu du sous-modèle peut être une page HTML complète car TinyButStong va rechercher les balises `<body>` et ne retenir que la partie Html entre ces deux balises si elles sont trouvées. Cela vous permet de travailler avec des sous-modèles [WYSIWYG](#). Si vous vous préoccupez de l'optimisation fine de la fusion, vous pouvez annuler cette fonctionnalité en définissant explicitement le paramètre `htmlconv=no` dans le champ TBS.

Le paramètre `file` est traité avant les autres paramètres du champ, et le contenu du fichier devient la

valeur courante du champ. Prenez cela en compte si vous voulez utiliser d'autres paramètres dans le champ TBS.

Insertion assistée par du code Php avec le paramètre **subtpl** :

Le paramètre **subtpl** est utile pour gérer l'insertion d'un sous-modèle à l'aide de code Php. La paramètre **subtpl** n'est actif que lorsqu'il est utilisé avec un paramètre **script** ou **onformat**. Il bascule l'instance courante de TBS dans le mode Sous-modèle pendant l'exécution du script ou de la fonction, et peut agir sur un nouveau modèle sans altérer le modèle principal.

Le mode Sous-modèle présente les caractéristiques suivantes :

- * Les sorties Php sont affichées à l'emplacement du champ au lieu d'être envoyés immédiatement au client. Par exemple, l'utilisation de la commande Php `echo()` insérera un texte dans le modèle principal au lieu de le sortir directement. L'utilisation de la méthode `Show()` insérera aussi le résultat de la sous-fusion dans le modèle principal.
- * Une référence à l'instance de TBS est fournie par la variable **\$this** ou **\$TBS**, selon que vous utilisez le paramètre **script** ou **onformat**. Cette variable peut être utilisée pour de nouvelles sous-fusions sans altérer le modèle principal. La méthode `Show()` n'arrêtera pas l'exécution de script durant le mode Sous-modèle, comme cela est fait par défaut dans le mode normal.

Quand le script ou la fonction se termine, l'instance TBS retourne en mode normal et agit sur le modèle principal.

Exemple avec le paramètre **script** :

<u>HTML :</u>	<code>[var.fichier;script=specialbox.php;subtpl]</code>
<u>Script PHP :</u>	<pre><?php echo(' * Ici insertion d'un sous-modèle * '); \$this->LoadTemplate(\$CurrVal); \$this->MergeBlock('blk1',\$GLOBALS['conn_id'],'SELECT * FROM table1'); \$this->Show(); ?></pre>
<u>Remarques :</u>	\$CurrVal est une variable locale fournie par TBS quand on utilise le paramètre script ; cette variable fait référence à la valeur du champ en cours de fusion. Dans l'exemple ci-dessus, \$CurrVal contient la valeur de la variable globale \$file . Vous pouvez la remplacer, par exemple, par le nom du fichier du sous-modèle à charger (par exemple : 'monsousmodele.htm'). Voir le paramètre script pour plus d'information.

Exemple avec le paramètre **onformat** :

<u>HTML :</u>	<code>[var.user_mode;onformat=f_user_info;subtpl]</code>
<u>Fonction PHP :</u>	<pre>function f_user_info(\$FieldName,&\$CurrVal,&\$CurrPrm,&\$TBS) { if (\$CurrVal==1) { // User is logged in \$TBS->LoadTemplate('user_info.htm'); \$TBS->MergeBlock('blk1',\$GLOBALS['conn_id'],'SELECT * FROM table1'); \$TBS->Show(); } else { // User not logged in echo('You are not logged in.');</pre>
<u>Remarques :</u>	\$CurrVal est une variable déclarée en tant qu'argument de la fonction. C'est TBS qui se charge d'appeler cette fonction de sorte que \$CurrVal fasse référence à la valeur du champ en cours de fusion. Dans exemple ci-dessus, \$CurrVal est égal à la valeur global \$user_mode . De même la variable \$CurrPrm fait référence au tableau des paramètres du champ en cours de fusion, et \$TBS fait référence l'instance de TinyButStrong en cours d'utilisation. Voir le paramètre onformat pour plus d'information.

Affichage conditionnel :

TinyButStrong offre plusieurs outils pour gérer l'affichage conditionnel pour les champs et les blocs.

Champs conditionnels

Tous les champs TBS acceptent les paramètres d'affichage conditionnel rappelés ci-dessous.

<u>Paramètre</u>	<u>Description</u>
------------------	--------------------

. (point)	Affiche un espace insécable Html si la valeur du champ est vide.
ifempty=valeur2	Affiche valeur2 si la valeur du champ est vide.
magnet=tag	Supprime une balise ou un couple de balise si la valeur du champ est vide.
if condition then valeur1 else valeur2	Affiche valeur1 ou valeur2 selon que la condition est vérifiée ou non.
frm=format1 format2 format3 format4	Change le format numérique ou date/heure selon que la valeur du champ est positive, négative, zéro ou vide.

Exemple :

```
[var.error_id;if [val]=0;then 'pas d''erreur';else 'erreur constatée']
```

Sections conditionnelles

Vous pouvez utiliser des sections conditionnelles dans n'importe quel bloc TBS. Une section conditionnelle est une section normale qui a un paramètre **when** définissant une condition, ou bien un paramètre **default**. Lors de la fusion du bloc, chaque condition **when** est évaluée jusqu'à ce qu'une soit vérifiée. Dès qu'une condition **when** est vérifiée, sa section conditionnelle est conservée et toutes les autres sont supprimées. Si aucune condition **when** n'est vérifiée, alors la section **default** est conservée si elle existe. Par défaut, les sections conditionnelles sont exclusives pour un même bloc. C'est à dire qu'une seule section conditionnelle peut être affichée par bloc. Mais cela peut être changé en utilisant le paramètre **several**. Voir ci-après pour plus de détails.

Section conditionnelle pour les blocs normaux :

Les blocs normaux sont ceux que vous fusionnez avec des données en utilisant la méthode MergeBlock(). Les blocs normaux peuvent avoir des sections conditionnelles. Les conditions sont évaluées pour chaque enregistrement de la source de données, et elles peuvent être des expressions contenant des champs liés ou des champs Var.

Exemple :

Nom : [b1.Nom;block=tr]	section normale
Adresse : [b1.ad_ligne1;block=tr;when [b1.adresse]=1] [b1.ad_ligne2] [b1.ad_cp] - [b1.add_town]	section conditionnelle
Sans adresse.[b1;block=tr;default]	section conditionnelle

Section conditionnelle pour les blocs automatiques :

Les **blocs automatiques** ne sont pas fusionnés avec des données ; c'est pourquoi ils ne peuvent pas avoir de sections normales ni de champs liés. Les blocs automatiques ne peuvent avoir que des sections conditionnelles. Les conditions sont évaluées une seule fois, et elles peuvent être des expressions contenant des champs Var.

Exemple :

[onload_lumiere;block=tr;when [var.lumiere]=1] Lumière allumée.
[onload_lumiere;block=tr;when [var.lumiere]=0] Lumière éteinte.
[onload_lumiere;block=tr;default] Lumière ?

Ce bloc sera fusionné automatiquement quand le modèle est chargé.

Sections conditionnelles non-exclusives :

Si vous souhaitez qu'un bloc ait des sections conditionnelles non-exclusives, vous pouvez utiliser le paramètre **several** sur la première section conditionnelle. Avec ce paramètre, toutes les conditions sont évaluées et chaque condition vraie fera afficher sa section.

Exemple :

```
[onload_err;block=tr;when [var.email]="";several] Votre email est vide.
```

```
[onload_err;block=tr;when [var.nom]=0] Votre nom est vide.
```

```
[onload_err;block=tr;default] Tout est ok.
```

Résumés :

Paramètres de champ TBS :

<u>Paramètre</u>	<u>Résumé</u>
htmlconv	Mode de conversion Html pour la valeur du champ.
. (point)	Si la valeur du champ est vide, affiche un espace insécable.
ifempty	Si la valeur du champ est vide, affiche une autre valeur.
magnet	Si la valeur du champ est vide, supprime des balises proches.
mtype	S'utilise avec magnet .
if	Si la condition est vérifiée, change la valeur affichée.
then	S'utilise avec if .
else	S'utilise avec if .
onformat	Exécute une fonction Php de l'utilisateur qui modifie la fusion du champ.
max	Limite le nombre de caractères affichés.
frm	Applique un format date-heure ou numérique.
locale	S'utilise avec frm . Affiche des noms de jours et de mois locaux.
protect	Mode de protection sur les caractères '['.
selected	Sélectionne un item dans une liste Html.
selbounds	S'utilise avec selected . Change la zone de recherche des items.
comm	étend les limites du champ à la balise commentaire qui l'entoure.
noerr	Empêche l'affichage de certaines erreurs TBS.
file	Insert le contenu du fichier.
script	Exécute le script Php.
getob	S'utilise avec script . Récupère les textes affichés par <code>echo</code> et les place à l'emplacement du champ.
once	S'utilise avec script . Empêche le script de s'exécuter plusieurs fois.

Paramètres de bloc TBS :

<u>Paramètre</u>	<u>Résumé</u>
block	Défini les limites du bloc.
extend	étend les limites du bloc sur plusieurs balises Html successives.
encaps	étend les limites du bloc sur plusieurs balises Html encapsulées.
comm	étend les limites du bloc à la balise Html commentaire..
nodata	Désigne la section qui s'affiche lorsqu'il n'y a aucun enregistrement dans la source de données.
headergrp	Désigne une section d'entête qui sera affichée lorsque la valeur d'une colonne change.
footergrp	Désigne une section de pied qui sera affichée lorsque la valeur d'une colonne change.
splittergrp	Désigne une section de séparation qui sera affichée lorsque la valeur d'une colonne change.
parentgrp	Désigne une section parent qui sera affichée lorsque la valeur d'une colonne change.
serial	Désigne une section qui contient une série de plusieurs enregistrements.

p1	Envoie un paramètre à la requête dynamique de la source de données.
ondata	Exécute une fonction Php de l'utilisateur pour modifier l'enregistrement lorsqu'il vient juste d'être récupéré de la source de données.
onsection	Exécute une fonction Php de l'utilisateur pour modifier la section en cours de fusion.
tplvars	S'utilise avec des champs onload uniquement. Définit des variables de modèle.
when	S'utilise avec onload ou onshow . Affiche la section quand la condition est vérifiée.
default	S'utilise avec onload ou onshow . Affiche la section quand aucune condition n'est vérifiée.
several	S'utilise avec when . Indique que plusieurs sections du groupe peuvent être affichées.

Champs et paramètres de barre de navigation :

<u>Champ</u>	<u>Résumé</u>
nav.page	Affiche le numéro d'une page.
nav.curr	Affiche le numéro de la page en cours.
nav.first	Affiche le numéro de la première page (toujours 1).
nav.prev	Affiche le numéro de la page précédente.
nav.next	Affiche le numéro de la page suivante.
nav.last	Affiche le numéro de la dernière page (-1 si inconnu).

<u>Paramètre</u>	<u>Résumé</u>
currpage	Indique que la section ne s'affiche que pour la page courante.
endpoint	Retourne une chaîne vide si la page courante est la première page ou la dernière page.
navpos	Indique comment la barre de navigation est positionnée par rapport au numéro de page active.
navsize	Indique le nombre de page à afficher.
pagemin	Indique le numéro de la première page.

Noms de blocs et champs spéciaux :

<u>Nom</u>	<u>Résumé</u>
val	Le mot-clé [val] peut être utilisé dans des paramètres de champs pour représenter la valeur de ce champ.
var.*	Affiche une variable Php.
var..*	Affiche une information du système TinyButStrong.
#	Colonne virtuelle pour un bloc. Affiche le numéro de l'enregistrement en cours.
\$	Colonne virtuelle pour un bloc. Affiche la clé de l'enregistrement en cours si la source de données est un tableau Php.
onload	Champ ou bloc automatique, fusionné lorsque le modèle est chargé.
onshow	Champ ou bloc automatique, fusionné lorsque le modèle est affiché.

~~~~~