Site: www.tinybutstrong.com
Authors: skrol29@freesurf.fr, Pirjo
Date: 2005-05-06

TinyButStrong

version 2.02

Template Engine for Pro and Beginners
for PHP version 4.0.6 or higher

Table of Contents:

Introduction:

TinyButStrong (TBS) is a PHP class useful to develop an application in a clean way, separating PHP scripts and HTML files. With TBS, HTML pages are generated dynamically by merging a template with data. It is called a Template Engine.

The name TBS comes from the fact that this tool contains only 8 functions and yet, it is very powerful. It allows you to merge HTML page templates with your PHP variables or your MySQL, PostgreSQL, or SQLite.

TBS has been engineered so that you can develop your HTML page templates with ease using any visual HTML editors (like Dreamweaver or FrontPage). But if you are used to designing your HTML pages with a text editor, it is nice as well. TBS also enables you to create JavaScript dynamically.

As the name of it tells, TBS is easy to use, strong and fast. It is completely °~° freeware °~°.

Basic principles:

On the HTML side:
You design a page which does not necessarily contain any PHP scripts, nor any programming. In this page you place TBS tags in the places where you want to display the dynamic data. This page is called a 'template'.
There are two types of tags: the 'fields' which are used to display dynamic data items, and the 'blocks' which are used to define an area, mostly in order to display records from a data source.

On the PHP side:
You use an object TBS variable to manage the merge of your HTML Template with the data. At the end, TBS shows the result of the merge.

Installation:

1. Copy the file tbs_class.php in a directory of your Web site.
2. At the beginning of your PHP program, add the lines:

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

Remark: if the TBS file tbs_class.php is in a different directory than your application, then you have to precise the directory in front of the TBS file name.

Explanations and technical details:
TinyButStrong is a library written in PHP, it's a component to be referenced in your own PHP programs. In technical terms, TinyButStrong is a PHP 'class' ; the name of this class is clsTinyButStrong.
The variable $TBS that you add at the beginning of your PHP program enables you to execute the merge of your template from your PHP application. In technical terms, the variable $TBS is an 'instance' of the clsTinyButStrong class.

Mini examples:

Example 1:

Html Template

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

Php Program

```
<?

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

$message = 'Hello' ;
$TBS->Show() ;

?>
```

Result

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

Example 2:

Html Template

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

Php Program

```
<?

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

Result

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

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

$TBS->Show() ;

?>
```

PHP side:

The merging of a template is done in a PHP program using an object variable declared as a clsTinyButStrong class.
Example of statement: `$TBS = new clsTinyButStrong ;`
This object allows you to load a template, to handle the merging of it with data, and then to show the result.

Example of PHP code:

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

Here is the list of the TinyButStrong object's properties and methods:

method LoadTemplate():

Loads a template for the merging process.
The complete contents of the file is stored in the Source property of the TBS object.

Syntax:

```
$TBS->LoadTemplate(string File{, string HtmlCharSet})
```

Argument	Description
File	Local or absolute path of the file to load.
HtmlCharSet	Optional. Indicates the character encoding (charset) to use for Html conversion of the data when they will be merged. It should be the same as the charset of the template. The default value is '' (empty string) which is equivalent to 'ISO-8859-1' (Latin 1). If your template uses a special charset, then indicate the Html value for this charset. In a Html page, the charset is placed at the beginning of the file, in the attribute 'content' of a <Meta> tag. The charsets supported by TBS are the charsets supported by the PHP function htmlentities(). For example: 'BIG5' (Chinese) or 'EUCJP' (Japanese).

No Html conversion:
If you use value False as the parameter HtmlCharSet, data will to not be converted when merged to the model.

User function:
If your charset is not yet supported by PHP, you can indicate a user function that will perform the Html conversion. For this, use the parameter HtmlCharSet with the syntax '=myfunction'.
The user function must take a string argument and return the converted string.

Adding the file at the end of the current template:
You can use the keyword '+' instead of the the charset to have the file added to the end of the current template. Charset parameter stay the same as for the first template.

method MergeBlock():

Merges one or several TBS blocks with records coming from a data source.
Returns the number of the last displayed record (the first is number 1).

TinyButStrong supports several data source types in native:
Php data: an array, a string, a number.

Databases: MySQL ; PostgreSQL ; SQLite.
You can also add a new one: 'adding a data source type'.

There is a display 'By Page' mode, described below.

Syntax: `int $TBS->MergeBlock(string BlockName, mixed Source{, string Query}{, int PageSize, int PageNum}{, int RecCount})`

Argument	Description
BlockName	Indicates the name of the TBS block to merge. You can merge several blocks with the same data by indicating their names separated by commas.
Source	Indicates the data source to merge. The table below shows the possible values according to the data source type.
Query	Optional. Indicates the SQL statement which returns the records to merge. The table below shows the possible values according to the data source type.
PageSize	Optional. This argument must be defined if you want to activate the By Page mode. Indicates the number of records on one page.
PageNum	Optional. This argument must be defined if you want to activate the By Page mode. Indicates the number of the page to display. The first page is number 1. The special value -1 will display the last page of the record set.
RecCount	Optional. This argument is useful only with the By Page mode. It allows to adjust the calculation of the number of records returned by the MergeBlock() method.

RecCount	Value returned by MergeBlock()
0 :	It's the default value. The method returns the number of the last record displayed in the required page.
-1 :	The method reads all the records up to the end and returns the total number of records. However, only records of the required page will be displayed.
>0 :	The method returns the value of RecCount. However, it will return the number of the last record in the required page if it's higher than RecCount.

Use this parameter in order to calculate and save the total number of records.
For example:

```
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);
```

Link between the block and the records:

The MergeBlock() method searches in your template for the specified TBS block name. Then, the block is repeated as many times as there are records in the data source.
To display the data of a record, you have to use a linked TBS Field. A TBS Field is linked when the name of it is composed of the block's name followed by a dot and a column's or a key's name in the record set. A linked field must be inside the block.

Example:
Block's name: block1
Columns returned by the query: field1,field2,field3
Linked TBS Fields: [block1.field1], [block1.field2], [block1.field3]

If no block's definition is found in the template, then the MergeBlock() method will merge the first record with all linked fields found in the template.

You can also define more advanced blocks. For more information, refer to chapter TBS Blocks.

Merging several blocks with the same data:

You can merge several blocks with the same data by indicating their names separated by commas in the BlockName parameter. In this case, the query is opened only one time, and records are buffered to feed blocks.
Example:

```
$TBS->MergeBlock('block1,block2,block3','mysql','SELECT * FROM MyTable');
```

Counting the records:

To display the number of the record, use a TBS Field linked to the virtual column '#'.
If you put this field outside the block, it will display the total number of records.
Example: [block1.#]

The virtual column '$' will display the key of the current record if the data source is a Php array.
Example: [block1.$]

Resource and Request arguments according to the data source type:

Data Source Type	Source	Query
Text (*)	The keyword 'text'	A text
Number (*)	The keyword 'num'	A number or a special array (see below)
Clear (*)	The keyword 'clear'	-
Conditional (*)	The keyword 'cond'	-
PHP Array (*)	A Php array	-
	The keyword 'array'	A Php Array
	The keyword 'array'	A string that represents an array contained or nested in a PHP global variable (see below)
MySQL	A MySql connection identifier or the keyword 'mysql'	An SQL statement
	A MySql result identifier	-
PostgreSQL	A PostgreSql connection identifier	An SQL statement
	A PostgreSql result identifier	-
SQLite	An SQLite connection identifier	An SQLite statement
	An SQLite result identifier	-
custom	A keyword, an object or a resource identifier not mentioned in this table. See the chapter 'adding a data source type'.	An SQL statement or something else.

(*) See explanations in the chapter below.

Php data sources:

Text
The argument Source has to be equal to 'text'.
The whole block is replaced by the text (it must be a string) given as the Query argument. No linked Fields are processed except '#' which returns 1, or 0 if Query is an empty string.

Number
The argument Source has to be equal to 'num'.
The argument Query can be either a number or an array.

arg Query	Returned Record Set
Number:	This number has to be positive or equal to zero. The returned Record Set consists of a column 'val' where the value goes from 1 to this number .
Array:	This array has to contain a key 'min' and a key 'max' and eventually a key 'step'. The returned Record Set consists of a column 'val' which goes from the 'min' value to the 'max' value. Example: array('min'->101,'max'->150) will display 50 blocks numbered from 101 to 150.

Clear
The argument Source has to be the keyword 'clear'.
All blocks and sections are deleted. It is the same thing as merging with an empty array.

Conditional
The argument Source has to be the keyword'cond'.
The block is merged like it was a conditional blocks onload and onshow. The block is not merged with data, and so it must have no linked TBS field. Each block section needs a parameter when or a

parameter default. See conditional blocks for more details.

Array

The argument Source has to be a PHP Array or the keyword 'array'. If you use the keyword 'array', then the argument Query has to be a Php Array or a string that represents an array contained or nested in a global variable.

String syntax: 'globvar[item1][item2]...'

'globvar' is the name of a global variable $globvar which must be an array.
'item1' and 'item2' are the keys of an item or a subitem of $globvar.
Example:

```
$TBS->MergeBlock('block1','array','days[mon]');
```

This will merge 'block1' with the value $day['mon'] assuming it is an array.

It is possible to represent variable's name without items.
Example:

```
$TBS->MergeBlock('block1','array','days');
```

There are two advantages in using a string to represent the array:
-> Items will be read directly in the Array (assigned by reference) instead of reading a copy of the items. This can improve the performance.
-> You can use dynamic queries.

Displaying the key of current record:
You can use the virtual column '$' which will display the key of the current record. This can be useful especially for dynamic queries and sub-blocks.
Example: [block1.$]

Structure of supported arrays:
Items of the specified Array can be of two kinds: simple values with associated keys (case 1), or array values for whom items are themselves simple values with associated keys (case 2).

Case 1:
Example:
```
['key1']=>value1
['key2']=>value2
...
```
The returned Record Set consists of a column 'key' containing the name of the key, and a column 'val' containing the value of the key.

Case 2:
Example:
```
[0] => (['column1']=>value1-0 ; ['column2']=>value2-0 ; ...)
[1] => (['column1']=>value1-1 ; ['column2']=>value2-1 ; ...)
[2] => (['column1']=>value1-2 ; ['column2']=>value2-2 ; ...)
...
```
The returned Record Set consists of the columns 'column1', 'column2',... with their associated values.

By Page mode:

The By Page mode is activated when you place in the PageSize argument a value different from zero. The display of the data will then be limited to the page specified with PageNum. If PageNum has the value -1, then the last page will be displayed.

Important remark:

Although easy and practical, the By Page mode is not optimized for a large number of records. If the display is too slow, or if your database is heavily sought, then it is advised to use the native functions of your Database System (if it has limited queries features).
For example: with MySQL you can use the LIMIT clause.

Explanations: considering the variety of the SQL syntaxes, TinyButStrong is not able to modify a query so that it returns a limited Record Set. For example, it is not able to add the LIMIT clause into a MySQL query.
That's why TinyButStrong has to call the original query, and then read the records one by one ignoring all those who are before the required page. This makes the display more slow when the page number to be reached is high. When the page is reached, TinyButStrong releases the query without going to the end of the Record Set.

method Show():

Terminates the merge.

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

The Show method will perform the following:
- Merge Var fields,
- Merge [onshow] fields,
- Display the result (can be cancelled by Render property),
- End the script (can be cancelled by Render property).

The Render property allows to adjust the behaviour of the Show() method.
Parameter `Render` also allows to adjust the behaviour of the Show() method but only for one call.

method CacheAction():

Call an action of the TinyButStrong's Cache System. The Cache System enables you to manage manually or automatically the backup of the merge result into a temporary file called "cache" file. CacheAction() returns *true* or *false* depending to the success or failure of the action.

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

Argument	Description
CacheId	Unic id of the cache file. This must be a string, and it will be used in the name of the file.
Action/MaxAge	Determine the action to do. It must a be a TinyButStrong's predefined constant or a positive value. See table below for more details on available actions. The default value is 3600 wich correspond to an automatic backup with a max age of 1 hour.
Dir	Optional. The path of the directory where the cache file is saved. By default, it is the same directory as the script.

Manage a cache file:
The Cache System enables you to create, load, update or delete a cache file identified by `CacheId`. There is also an Automatic Mode that let the System manage those actions depending to a max-age of the cache file.

Cache on Show:
If you start the "Cache on Show", then the result of the merge will be automatically saved in the cache file at the first use of the Show() method. The recording takes place after the display of the result.
Please note that by default the Show() method causes the end of the script. If you want to continue some treatments after the "Cache on Show", then you have to set the Render property in order to avoid the end of the script.

Here is the list of possible actions for the `Action/MaxAge` argument, it can be a TinyButStrong preset constant or a positive numerical value.

Action	Description
x >=0 (positive number)	Automatic Mode with max-age: - If the cache file exists and has been created less that x seconds ago, then the file is loaded and the Show() method is executed. If you haven't set the Render property then the script ends after displaying the result, otherwise CacheAction() returns *true*. - If the cache file doesn't exist or if it has been created more than x seconds ago, then "Cache on Show" is started (see above). The script continues normally and CacheAction() returns *false*.
TBS_CACHENOW	Save the current result of the merge in the cache file corresponding to CacheId. CacheAction() returns *false*.
TBS_CACHELOAD	Load the cache file corresponding to CacheId. CacheAction() returns *true* if the cache file has been found and loaded, otherwise it returns *false*.
TBS_DELETE	Delete the cache file corresponding to CacheId, if it exists. You can delete all the cache files of the directory using '*' for CacheId. CacheAction() returns *false*.

TBS_CACHEONSHOW	Start "Cache on show" for the cache file corresponding to CacheId CacheAction() returns *false*.
TBS_CANCEL	Cancel "Cache on Show" whatever is CacheId. CacheAction() returns *false*.
TBS_CACHEGETAGE	Return the age of the cache file in seconds. Return *false* if the cache file doesn't exist.
TBS_CACHEGETNAME	Return the name of the cache file corresponding to the given CacheId. A name is returned even if the cache file doesn't exist yet.
TBS_CACHEISONSHOW	Return *true* if "Cache on show" is activated, otherwise, return *false*.

method GetBlockSource():

Returns the source of the TBS Block.
Only the definition of the first section of block will be returned, unless the Sections argument is set to *True*.
If no block is found, the method returns *False*.

Syntax: `string $TBS->GetBlockSource(string BlockName {, boolean Sections})`

Argument	Description
BlockName	The name of the block to search for.
Sections	Optional. The default value is *False*. If this parameter is set *True* the method returns an array that contains the definitions for all the sections of the named block. The first section is returned into the item [1] of the array.

This method enables you to get the source of a block in order to manually handle the merging.
After that, if you need to replace the block with text, you can use the MergeBlock() method with the 'text' parameter.

method MergeField():

Replaces one or several TBS Fields with a fixed value or by calling a user function.
Each TBS fields having the specified base name will be merged.

Syntax: `$TBS->MergeField(string BaseName, mixed X {, boolean FunctionMode})`

Argument	Description
BaseName	Base name of the TBS Fields. For example 'account'.
X	The value to display or a string that represent the name of a user function.
FunctionMode	Indicates that the value to display is calculated by a user function. The default value is false. If this argument is set to true, then X must be a text string giving the name of the user function. This function must exist and have the syntax described below.

Merging with a value:

X can be numeric, string, an array or an object. For an array or an object, names of TBS Fields must have suffixes like Var Fields.

Example:

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

In this example, the fields [account.id] and [account.name] will be merged.

Merging with a user function:

TBS calls this function for each field found in the template.
This function must have the following syntax:

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

When the function is called, its argument $Subname has for value the suffix of the field's name (example: for a field named 'ml.title', $Subname will have the value 'title'). And the optional argument $PrmLst contains an associative array with the field's parameters. The function must return the value to be merged.

Example:

```
$TBS->MergeField('ml','m_multilanguage',true);
...
function m_multilanguage($Subname) {
  global $lang_id;
  $rs = mysql_query("SELECT text_$lang_id AS txt FROM t_language WHERE key='$Subname");
  $rec = mysql_fetch_array($rs);
  return $rec['txt'] ;
}
```

In this example, a field such as [ml.title] will be merged with the value returned by m_multilanguage('title').

method MergeNavigationBar():

Displays a navigation bar based on specific TBS block and TBS fields.
For more details on how to build a navigation bar, please read 'Display a navigation bar'.

Syntax: `$TBS->MergeNavigationBar(string NavName, mix Options, int PageNum [, int RecCount, int PageSize])`

Argument	Description
NavName	The name of the navigation bar.
Options	Enables you to force some options of the navigation bar. Those options can also be defined using block parameters in the template. But if you put them at the MergeNavigationBar() too, they will be forced. This parameter can be blank ('', 0 or null), a numeric value or an array. If it's a numeric value, it indicates the number of pages displayed. If it's an array, it can contain the following items: Key — Value 'navsize' — Number of pages displayed in the navigation bar. (default = 10). 'navpos' — Position of the navigation bar compared to the active page number. Use one of the following keywords: - 'step' (by default) to have the bar progressing by step. - 'centred' to center the bar on the active page number. 'navdel' — Name of a TBS block to delete when there is only one page or no page to display. This TBS block must surroud the navigation bar. If there are several pages to display then only TBS definition tags of this bloc are deleted. 'pagemin' — Number of the first page (default = 1).
PageNum	Number of the active page. The first page is number 1. To indicate the last page, use the value -1.
RecCount	Optional. The default value is -1. Indicates the total number of records, if known. If this number is unknown, you have to put the value -1. This argument is used only to calculate the number of the last page of the navigation bar.
PageSize	Optional. The default value is 1. Indicates the number of records per page. It has to be used together with RecCount. It is used only to calculate the number of the last page of the navigation bar.

Example:

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

method MergeSpecial():

Replaces the special blocks and fields of the specified type.Syntax: `$TBS->MergeSpecial(string Type)`

The argument `Type` has to be one of the following values:

Value	Description
'var'	Replaces all Var fields.

'onload'	Replaces all onload fields.
'onshow'	Replaces all onshow fields.

Remark:
By default, the Show() method replaces all the special fields and blocks just before showing the merge result. That's why it is rare to use MergeSpecial() in a program.

property Render:

Indicates how the merging ends.
The value must be a combination of the following constants.
The default value is (TBS_OUTPUT + TBS_EXIT).

Syntax: `int $TBS->Render`

The Render property changes the behaviour the methods Show() and CacheAction().

Constant	Description
TBS_NOTHING	Indicates that none of the actions below are proceeded at the end of the merge.
TBS_OUTPUT	Indicates that the result of the merge must be displayed. TBS uses the Php command Echo.
TBS_EXIT	Indicates that we have to quit the script just after the end of the merge.

property Source:

Get or set the HTML source on which the merge process is applied.
After the call to the LoadTemplate() method, this property contains the HTML source of the template.
This property enables you to read or modify the result of the merge, in your code.

Syntax: `string $TBS->Source`

property TplVars:

Contains the array of template variables corresponding to current template.

Syntax: `array $TBS->TplVars`

You can define template variables using one or several onload automatic fields with parameter tplvars.
All other parameters that follow parameter tplvars are added to the TplVars property when the LoadTemplate() method is called.

Example:
[onload;tplvars;template_version='1.12.27';template_date='2004-10-26']
This TBS tag will create two items equivalent to the PHP code:

```
$TBS->TplVars['template_version'] = '1.12.27';
$TBS->TplVars['template_date'] = '2004-10-26';
```

Remarks:
- Parameter tplvars works only with onload automatic fields.
- You can use parameter tplvars several times in the same template.

Adding a data source type:

You can add another data source type not yet supported in native by TinyButStrong.
For that, you have to code three functions (or methods) with specific statements, and names corresponding to the type to add.
Do not add the functions in the TBS source file, code them in your application or in an external Php script.
You can find additional data source types at the TinyButStrong web site.

You have the choice between coding user functions and coding methods of a class.
If you choose to code user functions, they have to have name which use a TBS identifier specific to the data source you use (see below).
If you choose to code methods of a class, those methods must be named tbsdb_open, tbsdb_fetch and tbsdb_close.

Example:

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

TBS identifier (for user functions only):

The $Source argument that you pass to the MergeBlock() method has a specific TBS identifier that you must use for the function naming.
If $Source is an object, then the TBS identifier is the name of Php class.
If $Source is a resource, then the TBS identifier is the resource type.
If $Source is a string, then the TBS identifier is this string.
The type of the $Source argument must not yet be supported in native by TinyButStrong, otherwise the functions will be ignored.
The TBS identifier may be arranged by TBS to make it fit for a function name.

Example:
If $Source is a Sybase connection (resource type = 'sybase-db link'), then the TBS identifier is 'sybase_db'.

Statements of the functions or methods:

The three functions to add in your application must have the following syntax:
If you are coding user functions, then replace the keyword 'customdb' with the TBS identifier of your data source type. If you are coding methods , replace the name of the functions with tbsdb_open, tbsdb_fetch and tbsdb_close.

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

This function must open the required query and return a Record Set identifier.
In case of error, the function should return the value *False*, and can display a message.

Argument	Description
$Source	Is the same argument given to the MergeBlock() method.
$Query	Is the same argument given to the MergeBlock() method.

Example:

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

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

This function has to return an associative array corresponding to the current record, with columns' names and values. The function has to return the value *False* when there is no record left.

Argument	Description
$Rs	The Record Set identifier returned by the tbsdb_customdb_open() function.
$RecNum	Optional. The number of the expected record. First is number 1.

Example:

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

If your data source needs to know the number of the expected record, you can add the argument $RecNum to your function's statement. But in other cases, this argument is optional because all records are called in order anyway.

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

This function has to close or free the Record Set identifier.
It doesn't have to return a value.

Argument	Description
$Rs	The Record Set identifier returned by the tbsdb_customdb_open() function.

Example:

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

Object Oriented Programming:

If you are a OOP developer, you may prefer TBS to work with object methods instead of global functions, and with object properties instead of global variables.

To do this, you first have to make the ObjectRef property referring directly or indirectly to an existing object.
Example:

```
$TBS->ObjectRef = &$MyObject; // Use '&' to define the property by reference.
```

or:

```
$TBS->ObjectRef['key1'] = &$MyObject; // Indirect reference
```

Then, you just have to prefix a function name by character '~' in all TBS features to indicate a ObjectRef method instead of a global function. Methods coded into the class must have the same syntax as the corresponding global function needed for the feature.
Like normal Var Fields, OOP syntax supports sub entities (array items, properties, methods) separated by dots.

Var Fields can refer to methods of the ObjectRef object, but also to its properties.

Features that support the '~' prefix are listed below:

Features	Example
Var Fields	[var.~prop1] ... [var.~.key1.prop1] ... [var.~meth1] ... [var.~prop2.subprop]
Parameter onsection	[blk1.column1;block=tr;onsection=~meth_onsec]
Parameter onformat	[blk1.column2;onformat=~meth_onfrm]
MergeField() method	$TBS->MergeField('fldname','~meth_MrgFld',true);
Custom Data Functions (*)	$TBS->MergeBlock('blk1','~mydb','SELECT * FROM t_table');
Custom Html conversion function	$TBS->LoadTemplate('mytemplate.htm','=~meth_htmlconv');

(*) Three methods, instead of three functions, have to be defined for the given keyword.
For example with the keyword '~mydb', the methods must be named mydb_open(), mydb_fetch() and mydb_close().
Example (fits for ezSQL):

```
class mydb Extends db {
  function mydb_open(&$source,&$query) {
    $this->get_results($query) ;
    return $this ;
  }
  function mydb_fetch(&$db,$num) {
    $x = $this->get_row(null,ARRAY_A,$num-1) ;
    if (is_array($x)) {
      return $x ;
    } else {
      return false ;
    }
  }
  function mydb_close(&$db) {
    // not needed
  }
}
```

HTML side:

You design your template by placing **TBS tags** in the places where data items should appear.

There are two types of TBS tags: *Fields* and *Blocks*.

A TBS Field is a TBS tag which has to be replaced by a single data item. It is possible to specify a display format and also other parameters. The syntax for TBS Fields is described below.

A TBS Block is an area which has to be repeated. It is defined using one or two TBS fields.
Most often, it is the row of an HTML table. The syntax for TBS Blocks is described below.

TBS Fields:

A TBS Field is a TBS tag which has to be replaced by a single data item.

It has a name which enables you to identify it and parameters can be supplied in order to change the display behaviour.

Syntax: **HTML ...** [FieldName;params] **... HTML**

Element	Description
FieldName	The name of the Field. Warning: names that begin with var. , onload and onshow are reserved. They are respectively used for Var fields, and Automatic fields.
params	Optional. One or more parameters from the list below and separated with ';'. Some parameters can be set to a value using the equal sign '='. Example: frm=0.00 If the value contains spaces or semicolons, you can use single quotes. Example: frm='0 000.00'. It is possible to embed TBS fields. It means you can write this: [var.v1; if [var.v2]=1]. But: - for Var fields, you have to make sure that v2 will be merged before v1. - for block fields, you have to make sure that column v2 is before column v1.

Field's parameters:

Parameter	Description
htmlconv=val	Enables you to force or prevent the conversion of the data item to Html text. The value val can be one of the following keywords: yes: (default value) Force the conversion to Html including new lines. no: Prevent the conversion to Html. Useful to modify Javascript code or to modify the Html source. nobr: Force the conversion to Html but new lines (useful for <pre> tags for example). wsp: Preserve white spaces (useful for spaces at the beginning of lines). esc: No Html conversion and double the single quote characters ('). js: Convert the data item to a string that can be inserted between JavaScript text delimiters. look: Convert the data item to Html only if no Html entities are found inside the data item. You can specify several values using seperator '+'. Example : htmlconv=yes+js
. (dot)	If the data item is empty, then an unbreakable space is displayed. Useful for cells in tables.
ifempty=val	If the data item is empty, then it is replaced with the specified value.
magnet=tag	Assign a magnet Html tag to the TBS field. A magnet tag is kept as is when the field has a value, and is deleted when the field is null or empty string. Example: (click here) Result for `$link='www.tbs.com'`: (click here) Result for `$link=''`: () By default, the magnet Html tag should be a pair of opening-closing tags (like <a>) which first tag is placed before the TBS fields. But this can be changed using parameter mtype (see below). Remark: the parameters if then else are processed before parameter magnet.
mtype=val	To be used with parameter magnet. Define the magnet type. Value: m*m — Magnet behavior when field is null or empty string: That's the default value. Delete the pair of tags that surrounds the TBS field. Everything that is between them is deleted also. The field can be put inside one of the tags. Example: (click here) Result for `$link='www.tbs.com'`: (click here) Result for `$link=''`: ()

m+m — Delete the pair of tags that surrounds the TBS field, but keeping everything else that is between the tags.
Example:
([blk.name])
Result for `$email='me@tbs.com'`: (MyName)
Result for `$email=''`: (MyName)

m* — Delete the single tag that is before the field, and everything that is between the tag and the field.
Example 1: <img href="[var.link;magnet=img;mtype=m*]">
Example 2:
 [var.address;magnet=br]

*m — Delete the single tag that is after the field, and everything that is between the tag and the field.
Example: [var.address;magnet=br;mtype=*m]

selected — This parameter enables you to select an item for a List, Radio buttons or Checkboxes placed into a Html form. You have to ensure that items are created (merged) before the merge.

Html List:
Use the parameter selected without setting a value to it. The TBS Field has to be placed within the list of values. When the TBS field is merged it is deleted, but the item which has the same value as the field will be selected. If the value is not found, a new item is added.
Example:

Boston Washington New York [town_id;selected]	which will be after the merge:	Boston **Washington** New York

Radio buttons and Checkboxes:
Use the parameter selected with setting a value to it which is the name of the Radio buttons or Checkboxes to process. The TBS Field has to be placed within the form. When the TBS field is merged it is deleted, but the item which has the same value as the field will be selected.
Example:

○ Boston [town_id;selected=r_test] ○ Washington ○ New York	which will be after the merge:	○ Boston ◉ Washington ○ New York

In this example, the Radio button captioned 'Washington ' has been selected because the name of the Radio button tag is ''r_test' and its value is 2, and the TBS tag named 'town_id' has been merged with the value 2.

Multi-selection:
For Lists, Radio buttons or Checkboxes, you can make a multi-selection by giving a Php array as the value of the TBS field.

Bounds:
By default the bounds for searching items to select are html tags <select> for List , and <form> for Radio buttons and Checkboxes. But you can change them using parameter selbounds (see below).

selbounds=tag — To be used with parameter selected. It enables you to change the search zone for items to select by indicating a Html tag type. By default, this value is select for a List, and form for Radio buttons and Checkboxes.
Example: [town_id;selected=r_test;selbounds=div]
In this example, items to select will be searched between and tags that surround the TBS field.

comm — This parameter enables you to widen the bounds of the TBS Field up to the bounds of the commentary Html tag which surround it.
<!-- [myfield;comm] this is an example--> is strictly identical to [myfield]
This is particularly useful for the template designing when you are using a Visual HTML Editor (such as Dreamweaver or FrontPage).

noerr — Avoid some of the TBS Error messages. When a message can be cancelled, it is

mentioned in the message.

file=filename

Replace the field with the contents of the file. Filename can be a string or an expression built with Var fields that returns the file path.
How to use this parameter is detailed in the chapter Subtemplates.

script=filename

Execute the Php script just before replacing the locator.
Filename can be a string or an expression built with Var fields that returns the file path.

* Take care that in your script **variables are not global but local**. This is because the script is called from a TBS method. In order to define or reach global variables in your script, you have to use the Php instruction global or the array `$GLOBAL`.
* TBS gives to you predefined local variables that can be used in your script:
  - $CurrVal refers to the current value of the field. It can be modified.
  - $CurrPrm refers to the array of field's parameters.
  - $this refers to the current TBS instance. (See parameter subtpl for good usage)
* Parameter script is sensible to the if parameter. If there is a parameter if in the field, then the script is executed only if the condition is verified.

See chapter 'Subtemplates' for more details about how to use this parameter in subtemplate mode.

subtpl

To be used with the parameter script or parameter onformat.
Activate the subtemplate mode during the script or function execution.
See chapter 'Subtemplates' for more details.

once

To be used with the parameter script.
Cancel the script execution if it has previously been called.

getob

This parameter is deprecated because it can be replaced with parameter subtpl.
To be used with the parameter script. Indicates that the text displayed using the echo() command in the Php script replaces the value of the TBS Field.

if expr1=expr2

Display the data item only if the condition is verified, otherwise display nothing unless parameter then or else are used.
Supported operators are:

= or ==	equal
!=	not equal
+-	greater than
+=-	greater than or equal to
-+	less than
-=+	less than or equal to

Both expr1 and expr2 must be string or numerical expressions. You can use the keyword [val] inside the expressions to represent the data item. The expressions may contain TBS fields, but you have to make sure that they are merged before the containing field.
See parameters then and else for some examples.

then val1

If the parameter if is defined and its condition is verified, then the data item is replaced with val1.
Example:
[var.image;if [val]='';then 'image0.gif']

else val2

If the parameter if is defined and its condition is not verified, then the data item is replaced with val2.
Example:
[var.error_id;if [val]=0;then 'no error';else 'error found']

onformat=fct_name

Indicates the name of a user Php function that will be executed before the merge of the field. The function fct_name must have the following syntax:

```
function fct_name($FieldName,&$CurrVal,{&$CurrPrm,{&$TBS}}) { ... }
```

Parameter	Description
$FieldName	Gives the name of the current field (read only).
$CurrVal	Refers to the value of the current field (read/write *; don't forget the & character in the statement*).
$CurrPrm	Optional. Refers to the array of parameters for the current field (*Don't forget the & character in the statement*).
$TBS	Optional. Refers to the current TBS instance. (*Don't forget the & character in the statement*). Use this parameter with lot of care. It is provided for the

subtemplate mode.
See chapter 'Subtemplates' for more details about how to use this parameter in subtemplate mode.

protect=val

Enables you to protect or unprotect the data item to be merged by replacing the characters '[' with their corresponding Html code '['. The value val can be one of the following keywords:
yes: (default value) data item is protected.
no: data item is not protected.
By default, all data merged with a template is protected except if it's a file inclusion. It is strongly recommended to protect data when it comes from free enter like on a forum for example.

max=val

Indicates the maximum number of characters to display. Beyond this limit, the data item is cut and an ellipsis (...) is added at the bottom.

frm=format

Specify a format to display a data item of type date/time or numeric. For a numeric item, it is possible to use a conditional format which changes depending on the sign of the value.

Date-time format:

It is a VisualBasic like format. The following keywords are recognized:

d, dd, ddd, dddd:	number of the day, number of the day in two digits, short name of the day, full name of the day. Use parameter locale to display locale names.
xx	displays st, nd, rd or th depending to the number of the day.
m, mm, mmm, mmmm:	number of the month, number of the month in two digits, short name of the month, full name of the month. Use parameter locale to display locale names.
yy, yyyy:	year in two digits, full year.
hh, nn, ss:	hour, minutes, seconds in two digits.

Other characters are kept.
It is possible to protect the strings inside by putting them between single or double quotes.

Examples:
[fld;frm=mm/dd/yyyy] will display 12/21/2002
[fld;frm='yyyy-mm-dd hh:nn:ss'] will display 2002-12-21 15:45:03

Numeric format:

To define the decimal part, use an expression like '0x0...' where 'x' is the decimal separator , and '0...' is a continuation of zeros corresponding to the number of decimals.
If there is no decimal, use the format '0.' (with a dot).

To define a thousand separator, use an expression like '0z000x...' where 'z' is the thousand separator. If there is no decimal, use the format '0z000.' (with a dot).

If the format contains the character '%', then the value to display will be multiplied by 100. The character '%' is displayed too.

The numerical format may contain other strings. But only the expression with one or more zeroes placed to the right will be taken as a format, other characters will be kept.

Examples:

Value	Field	Display
2456.1426	[fld;frm='0.000']	2456.143
	[fld;frm='$ 0,000.00']	$ 2,456.14
	[fld;frm='$ 0,000.']	2,456
0.2537	[fld;frm='0.00 %']	25.37%
	[fld;frm='coef 0.00']	coef 0.25

Conditional formats:

	You have the possibility to define up to 4 conditional formats when the value is respectively positive, negative, zero or null (or empty string). Conditional formats must be separated by a '|' character. Each conditional format is optional. Examples: Value / Field / Display 2456.1426 [chp;frm='+0.00|-(0.00)|*|empty'] +2456.14 -156.333 [chp;frm='+0.00|-(0.00)|*|empty'] -(156.33) 0 [chp;frm='+0.00|-(0.00)|*|empty'] * null [chp;frm='+0.00|-(0.00)|*|empty'] empty -8.75 [chp;frm='+0.00|-(0.00)'] -(8.75)
locale	To be used with the parameter frm. Indicates that the format specified with frm must display locale day and month's names. Locale informations can be set using the PHP function setlocale().
tplvars	Enables you to define variables in the template that you can retrieve in the Php programm using TplVars property. Works only with onload automatic fields.

Var fields:

A Var field is a TBS Field which displays a Php variable.
The name of it must be composed by the keyword 'var.' followed by the name of the Php variable.
The parameters for standard TBS Fields are available for Var fields.

For example [var.php_version] will be replaced by "4.2.3".

The user variables and the predefined variables can be merged but they must be global variables. Resource variables are ignored.

It is possible to merge an Array variable by indicating the item with a dot.
For example: [var.myarray.item]

It is possible to merge an Object variable by indicating a property (or a method which doesn't need any arguments) with a dot.
For example: [var.myobject.property]

Embedded Var Fields
Embedded Var Fields placed into parameters file, script, if, when (and also then and else since TinyButStrong version 2.02) are always processed. In other cases, you have to be sure that the embedding Field is merged before the embedded Field, otherwise the embedded Var Field is ignored.

When are Var fields merged?
Var fields are merged in the Show() method, this means just before displaying the merge result. But you can force the merge at any time with the MergeSpecial() method.

Security: how to limit Var fields usage in templates?

You can limit the Var fields usage by defining an Allowed Variable Prefix when you create the TinyButStrong object.

Example :

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

In this example, only PHP global variables prefixed by 'x1_' are allowed in the template. Other Var fields will produce an explicit error message when merging.
[var.x1_title] will be merged if the global variable $x1_title exists.
[var.x2_title] will produce an explicit error message.

NB: the first parameter '' de clsTinyButStrong() in the example above is used to define TBS tag delimiters. But this is not described in this manual.

Special Var fields:

A Special Var field is a TBS Field which displays data provided by the TinyButStrong system.
The name of a Special Var field has to begin with 'var..', followed by a keyword in the list below.
The parameters for standard TBS Fields are available for Special Var fields.

Example: Date of the day : [var..now;frm='mm-dd-yyyy']

Name	Description
var..now	Date and hour of the server.
var..version	The version of TinyButStrong.
var..script_name	The name of the PHP file currently executing.
var..template_name	The name of the last loaded template file. It is the name given to the LoadTemplate() method.
var..template_date	The creation date of the last loaded template file.
var..template_path	The directory of the last loaded template file. It is the directory given to the LoadTemplate() method.
var..tplvars.*	The value of an item set in the TplVars property. ('*' must be the key of an existing item in the array)

When are Special Var fields merged?
Special Var fields are merged with normal Var fields. That is in the Show() method, this means just before the display of the merge result. But you can force the merge at any time with the MergeSpecial() method.

TBS Blocks:

A TBS block enables you to display data from a record source.
The merging between the block and the data is done using the MergeBlock() method.

During the merge, the TBS block is repeated as many times as there are records; and the associated TBS fields are replaced by the value of the columns.
A TBS field associated to Block1 and displaying the value of the column ColumnA must be named Block1.ColumnA
Example: [Block1.ColumnA;frm='mm-dd-yyyy']

Two blocks with the same name will be regarded as two sections of the same block (see sections of blocks).

Block syntaxes:

There are three possible syntaxes to define a TBS block:

Explicit Syntax:
Two TBS tags are used. One for the beginning of the block and another for the end of the block.
Example:
HTML...[BlockName;block=begin;params]**...HTML...**[BlockName;block=end]**...HTML**
The TBS tags for the block definition will be deleted during the merging.

Relative Syntax:
The block is defined by a pair of opening-closing Html tags. Only one TBS tag is required.
Example:
HTML...<tag_name...>...[BlockName;block=tag_name;params]**...</tag_name...>...HTML**
The TBS tag for the block definition must be placed between the pair of Html tags.
This TBS tag will be deleted during the merging.

Simplified Syntax:
An associated TBS field is used to define the block in a relative way (see the relative syntax above).
Example:
HTML...<tag_name...>...
[BlockName.ColumnName;block=tag_name;params]**...</tag_name...>...HTML**
The TBS tag for the block definition must be placed between the pair of Html tags.
But it is not necessarily the first TBS field in the block.

Element	Description
BlockName	The name of the TBS block.
block=begin	Indicates the beginning of the block.
block=end	Indicates the end of the block.

block= tag_name	Indicates a block which is between the opening Html tag <tag_name...> and the closing Html tag </tag_name...> that are surrounding the TBS tag. Both the opening and closing Html tags are part of the block. - row can be used as an alias in order to indicate the row of a table. block=row is the same as block=tr. - opt can be used as an alias in order to indicate the item of an HTML list. block=opt is the same as block=option.
params	Optional. One or several parameters from the list below. Separated with ';'.

Which syntax to use?

The 'absolute' syntax is rarely used with Visual Editors because TBS tags have often to be placed between two Html tags. On the other hand, it is convenient for textual editors.

The 'relative' syntax enables you to indicate a block using only one TBS tag. Furthermore, there is no need to hide the TBS tag because it will be deleted during the displaying. This syntax is quite practical.

The 'simplified' syntax is really simple. It enables you to define a TBS block and a TBS Field with only one TBS tag. This syntax is the most current and the most practical.

Tip:
You can use the 'relative' or the 'absolute' syntax with custom tags using the Html standard.
Example:
<custom_tag>Hello [blk1.column1;block=custom_tag], how are you?</custom_tag>

Block's parameters:

Parameter	Description
extend=n extend=tag1,tag2,...	Extend the block definition upon more tags. Can be used only with the relative syntax or the simplified syntax. This enables you, for example, to define a block on two rows of a table. Syntax 1: using a number n (positive or negative) Extend the block definition upon the n next pairs of tags that follow. Tag's names are the same as parameter block. If n is negative, then the block is extended upon the previous pairs of tags. Syntax 2: using a list of tag names Extend the block definition upon the pairs of tags that follow. Tag names are those given by the list .
encaps=num	Indicates encapsulation level of the TBS tags compared to the HTML tags specified with the parameter block. The default value is 1. Example: [block1.field1;block=tr;encaps=2] | [block1.field2] In the example above, the blue row will be duplicated during the merging because there is 'encaps=2'. If 'encaps=1' is set or if the parameter is left off, then it will be the pink row that is duplicated in the merging.
comm	This parameter enables you to widen the bounds of the TBS tag up to the bounds of the commentary tag (HTML) surrounding it. <!-- [block1;block=tr;comm] this is an example--> is strictly identical to [block1;block=tr] This parameter is particularly useful for designing the template when using a visual HTML editor (such as Dreamweaver or FrontPage).
nodata	Indicates a section that is displayed only if there is no data to merge. Example: [block1.field1;block=tr] | [block1.field2] [block1;block=tr;nodata]There is no data.

For more information about sections, see the chapter 'Sections of blocks'.

headergrp=colname	Indicates a header section that is displayed each time the value of column colname changes. colname must be a valid column name returned by the data source. You can define several headergrp sections with different columns. Placement's order of headergrp sections in the block can modify the result. For more information about sections, see the chapter 'Sections of blocks'.
footergrp=colname	Indicates a footer section that is displayed each time the value of column colname changes. See headergrp.
splittergrp=colname	Indicates a splitter section that is displayed each time the value of column colname changes. See headergrp.
parentgrp=colname	Indicates a parent section that is displayed each time the value of column colname changes. Unlike other sections, a parentgrp section allows normal sections inside itself. It's a way to define both a header and a footer in one section.
serial	Indicates that the block is a main block which contains serial secondary blocks. For more information, see the chapter 'serial display (in columns)'.
p1=val1	Indicates the use of a dynamic query. All the occurrences of the string '%p1%' found in the query given to the MergeBlock() method are replaced by the value val1. For more information, see the chapter 'dynamic queries / sub-blocks'.
onsection=fct_name	Indicates the name of a user PHP function that will be executed during the block merging. The function is called each time a record is displayed. The function fct_name must have the following syntax: `function fct_name($BlockName,&$CurrRec,&$DetailSrc,$RecNum) { ... }`

Parameter	Description
$BlockName	Returns the name of the block calling the function (read only).
$CurrRec	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.
$DetailSrc	Returns the source of the current section (read/write *; don't forget the & in the function header*). If you set this variable to '', it cancels the displaying of this record.
$RecNum	Returns the number of the current record (read only, first record is number 1).

when expr1=expr2	Make the section conditional and define its condition. A conditional section is displayed only if its condition is verified. Supported operators are: = or == equal != not equal +- greater than +=- greater than or equal to -+ less than -=+ less than or equal to Both expr1 and expr2 must be string or numerical expressions. The expressions may contain Var fields for an automatic block, or linked fields for a merged block.
default	Indicates a section of block that must be displayed only if no conditional section of the same block has been displayed.
several	Indicates that several conditional sections of the block can be displayed if several conditions are true. By default, conditional sections are exclusive.

Sections of block:

Different blocks having the same name will be regarded as sections of the same block.
Sections can be used to:
- alternate the display (normal sections),
- display something if there is no data (NoData section),
- display a header each time the value of a column changes (grouping sections).

Normal sections:

When you define several normal sections, they will be used alternatively for each record.

Example:

[b1.caption;block=tr]
[b1.caption;block=tr]

In this example, the block named 'b1' contains two normal sections. Records will be displayed alternatively with a green background and with a blue background.

NoData section:

The NoData section is a section displayed only if the data source has no records. There can be only one NoData section in a block. The NoData section is defined by adding the parameter nodata.

Example:

[b1.caption;block=tr]
There is nothing. [b1;block=tr;nodata]

Grouping sections:

Grouping sections are displayed every time a column's value in the record-set changes. You can define header, footer, splitter or parent sections using parameters headergrp, footergrp, splittergrp, and parentgrp. See block's parameters for more details.

Example:

Year: [b1.year;block=tr;headergrp=year]	
[b1.caption;block=tr]	[b1.amount]

Conditional sections:

Conditional sections are displayed only if their condition is verified. The condition for display is defined using parameter when. As soon as a section has this parameter, it becomes conditional. See Conditional display for more details.

Example:

[b1.name;block=tr]
[b1.address;block=tr;when [b1.add_ok]==1]

Serial display (in columns):

The serial display enables you to display several records inside a block. For this, you have to use a main block and secondary blocks.

Example:

Rec 1	Rec 2	Rec 3	Rec 4
Rec 5	Rec 6	Rec 7	Rec 8
Rec 9	...	...	...

In this example, main blocks are the blue lines of the table, the secondary blocks are the pink cells.

Syntax:
The main block and its secondary blocks are merged using only one call to the MergeBock() method. The main block must be defined using the parameter serial. The secondary blocks must be nested into the main block. The secondary block's names must be the name of the main block followed by "_" and a number

indicating display order.

Example:

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

The corresponding PHP is:

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

Empty secondary block:
You can specify a special secondary block that will be used to replace unused secondary blocks (without records). This "Empty" secondary block must have the index 0. It can either be placed inside the main block with the normal secondary block, or alone inside another serial block. The "empty" secondary block is optional.

Example:

[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] **No records found.**			

Remark:
The serial display also works with sections of block and dynamic queries.

Dynamic queries / sub-blocks:

Principles of the dynamic queries:

It is possible to use the MergeBlock() method with a dynamic query.
In your template, you have to define a block by adding the parameters p1, p2, p3,... with their values.
The query given to the MergeBlock() method has to contain marks such as %p1%, %p2%, %p3%, ... in order to welcome the values of the parameters p1, p2, p3,... .

Each section of the block to be merged that contains a parameter p1 will be computed as a separate block for which the dynamic query is re-executed. The sections of the block that have no parameter p1 are combined with the previous section with a parameter p1.

Example:

[blk.town;block=tr;p1='france']	[blk.country]

[blk.town;block=tr;p1='us']	[blk.country]

Corresponding PHP code:

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

Result of the merge:

Paris	france
Toulouse	france

Washington	us
Boston	us

Use with sub-blocks:

Dynamic queries enable you to easily build a system of a main-block with sub-blocks. Here is how you can do it:
- Create a main block, and then a sub-block inside the main block.
- Link them by adding to the sub-block a parameter p1 whose value is a field from the main block.

- At the PHP side, merge the main block first, and then the sub-block.

Example:

Country: [main.country;block=table] [sub.town;block=tr;p1= [main.cntr_id]]

Corresponding PHP code:

```
$TBS->MergeBlock('main',$cnx_id,'SELECT country,cntr_id FROM t_country')
$TBS->MergeBlock('sub',$cnx_id,'SELECT town FROM t_town WHERE (cntr_id=%p1%)')
```

Result of the merge:

Country: France Paris Toulouse
Country: Germany Berlin Munich
Country: Spain Madrid Barcelona

Remarks:
- The parameter htmlconv=esc enables you to pass protected string values to the query.
- The dynamic queries also work with sections of block and serial display.

Display a navigation bar:

TinyButStrong is able to display a navigation bar using the MergeNavigationBar() method.
It is quite similar to merging a block using MergeBlock() except that there are page numbers instead of data, and you can use specific fields to display extra info, and options to arrange the navigation bar.

Blocks and fields:

Use a normal TBS block to display the page numbers.
This block will be merged with a virtual data source having as much records as pages to display, and with the following columns:

Name	Description
page	Returns the number of a common page, reachable from the navigation bar.
curr	Returns the number of the active page.
first	Returns the number of the first page (1 by default).
prev	Returns the number of the previous page.
next	Returns the number of the next page.
last	Returns the number of the last page if it's known, otherwise returns -1.

page is the only value that changes and its linked field must be placed inside the block. Others columns have always the same value and can be placed inside the block as well as outside the block.
Those fields support the parameter endpoint. It will replace the value of the field with an empty string ('') when the active page is equal to first page or last page. This enables you to manage display exceptions with parameter magnet for example.

Example:

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

In this example, the link will be deleted when the active page is the first page.

The block can contain a special section to display the active page differently.
This section is defined using parameter currpage on the block definition.

Example:

Template:

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

Php code used:

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

Result of the merge:

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

Remark: this example doesn't display links.

Options

The block definition can contain parameters that are specific to the navigation bar.
Those options can also be defined as a parameter of the MergeNavigationBar() method.

Parameter	Description
navsize=num	Number of pages displayed in the navigation bar. (default = 10).
navpos=keyword	Position of the navigation bar compared to the active page number. Use one of the following keywords: - 'step' (by default) to have the bar progressing by step. - 'centred' to center the bar on the active page number.
navdel=blockname	Name of a TBS block to delete when there is only one page or no page to display. This TBS block must surroud the navigation bar. If there are several pages to display then only TBS definition tags of this bloc are deleted.
pagemin=num	Number of the first page (default = 1).

Automatic fields and blocks:

onload and onshow are reserved names for TBS fields and blocks that are automatically merged when the template is loaded by the LoadTemplate() method and when the result is shown by the Show() method.

Automatic fields are merged with an empty value. They accept all TBS field's parameters.
They are useful for subtemplate and template variables.

Example:

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

Automatic blocks are not merged with data. They can have only conditional sections.

Examples:

```
[onload;block=tr;when [var.status]==1] Status 1
[onload;block=tr;when [var.status]==2] Status 2
```

See conditional sections for more details.

Subtemplates:

There are two ways to instert subtemplates in your main template.

Primary insertion using parameter file:

This is the best way to simply insert a part contained in another file, like usually done for headers and footers.

The value given to parameter file must be the name of a file existing on the server. You can use an expression with Var Fields and the [val] keyword which represent the value of the field.

Examples:

```
[onload;file=header.htm]
[onload;file=[var.file_header]]
[var.sub1;file=[val]]
```

Contents of the file is inserted at the place of the field, without no Html conversion and no TBS protection. [onload] tags contained in the file are not processed at the insertion. [onshow] tags and Var fields will be merged on the Show() method because they became part of the main template.

The subtemplate can contain any TBS fields, including Var fields and blocks to be merged. If you intend to merge data with a block defined into a subtemplate, then it's suggested to use parameter file in an [onload] field in order to ensure that the subtemplate is inserted before you call MergeBlock().

Contents of the subtemplate can be a full HTML page, because TinyButStrong will search for <body> tags and retain only Html part between those two tags if they're found. This enables you to work with WYSIWYG subtemplates. If your main concern is high speed merging, you can avoid this feature by explicitly defining parameter htmlconv=no in the TBS field.

Parameter file is processed before other field's parameters, and the contents of the file will make the current value of the field. Take this in account if you want to use other parameters in the TBS field.

Insertion driven with Php code using parameter subtpl:

Parameter subtpl is useful to manage subtemplate insertion with Php code. Parameter subtpl is active only when used with a parameter script or onformat. It turns the current TBS instance in Subtemplate mode during the script or function execution and can act on a new template without deteriorating the main template.

The Subtemplate mode presents the following characteristics:

* Php outputs are displayed at the field's place instead of being immediately sent to the client. For example, using the Php command echo() will insert a text in the main template instead of be directly output it. Usin Show() method will also insert the result of the sub-merge into the main template.

* A reference to the TBS instance is provided by local variable $this or $TBS, whether you use parameter sc or onformat. This variable be used for new submerges without deteriorating the main template. The Show method won't stop any script execution during the Subtemplate mode like it does by default in normal mo

When the script or the function ends, the TBS instance returns in normal mode with the main TBS template.

Example with parameter script:

HTML: [var.file;script=specialbox.php;subtpl]

PHP script:
```
<?php
  echo('* Here include a subtemplate *');
  $this->LoadTemplate($CurrVal);
  $this->MergeBlock('blk1',$GLOBALS['conn_id'],'SELECT * FROM table1');
  $this->Show();
?>
```

Remarks: $CurrVal is a local variable provided by TBS when using parameter script ; this variable is a referenc value of the field currently merged. In the example above, $CurrVal has the value of the global varia You can replace it, for example, by the name of the subtemplate to load (for example: 'mysubtpl.htn parameter script for more information.

Example with parameter onformat:

HTML: [var.user_mode;onformat=f_user_info;subtpl]

PHP user function:
```
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.');
  }
}
```

Remarks: $CurrVal is a variable declared as an argument of the function. It's TBS that is in charge to call this f making $CurrVal referring to the value of the fields currently merged. In this example above, $CurrV equal to the global variable $user_mode. In the same way, variable $CurrPrm is a reference to the a parameters of the field currently merged, and $TBS is a reference to the TinyButStrong instance curr used. See parameter onformat for more information.

Conditional display overview:

TinyButStrong offers several tools for conditional display for both fields and blocks.

Conditional fields

For any TBS fields you can use parameters for conditional display, recalled below.

Parameter	Description
. (dot)	Display an Html unbreakable space if the field value is empty.
ifempty=value2	Display value2 if the field value is empty.
magnet=tag	Delete a tag or a pair of tags if the field value is empty.
if condition then value1 else value2	Display value1 or value2 depending on whether the condition is verified or not.
frm=format1|format2|format3|format4	Changes the numeric format or date/time format depending on whether the value is positive, negative, zero or empty.

Example:
[var.error_id;if [val]=0;then 'no error';else 'error found']

Conditional sections

You can use conditional sections any TBS block. A conditional section is a normal section which has a parameter when defining a condition, or parameter default. At the block's merging, each when condition of conditional sections is evaluated until one is verified. As soon as one when condition is verified, its conditional section is kept and other conditional sections are deleted. If no when condition is verified, then the default section is displayed if it exists.
By default conditional sections are exclusive inside a block. It means only one conditional section of a block can be displayed. But this can be changed using parameter several. See below for more details.

Conditional section within normal blocks:

Normal blocks are those that you merge with data using the MergeBlock() method. Normal blocks can have conditional sections. Conditions are evaluated for each record of the data source, and they can be expressions containing linked fields or Var fields.

Example:

Name: [b1.Name;block=tr]	normal section
Address: [b1.add_line1;block=tr;when [b1.address]=1] [b1.add_line2] [b1.add_zip] - [b1.add_town]	conditional section
No address.[b1;block=tr;default]	conditional section

Conditional section within automatic blocks:

Automatic blocks are not merged with data ; that's why they cannot have normal sections and linked fields. Automatic blocks can have only conditional sections. Conditions are evaluated only once, and they be expressions containing Var fields.

Example:

```
[onload_ligth;block=tr;when [var.light]=1] Light is ON.
[onload_ligth;block=tr;when [var.light]=0] Light is OFF.
[onload_ligth;block=tr;default] Light is ?
```

This block will be automatically merged when the template is loaded.

Non-exclusive conditional sections:

If you want a block to have non-exclusive conditional sections, you can use parameter several on the first conditional section. With this parameter, all conditions are evaluated and each true condition makes its section to be displayed.

Example:

```
[onload_err;block=tr;when [var.email]='';several] Your email is empty.
[onload_err;block=tr;when [var.name]=0] Your name is empty.
[onload_err;block=tr;default] All is ok.
```

Summary:

TBS Field's parameters:

Parameter	Summary
htmlconv	Html conversion Mode for the field's value.
. (dot)	If the value is empty, then display an unbreakable space.
ifempty	If the value is empty, then display another value.
magnet	If the value is empty, then delete surrounding tags.
mtype	Use with magnet.
if	If the condition is verified, then change the value.
then	Use with if.
else	Use with if.
onformat	Executes a Php user function to modify the field merging.
max	Limits the number of characters.
frm	Apply a date-time or a numeric format.
locale	Use with frm. Display locale day and month's names.
protect	Protection mode for characters '['.
selected	Selects items in an Html list.
selbounds	Use with selected. Change the default bounds for searching items.
comm	Extends the field's bounds up to the Commentary tag that surround it.
noerr	Avoid some TBS error messages.
file	Includes the contents of the file.
script	Executes the Php script.
subtpl	Use with script or onformat. Turns the TBS instance into subtemplate mode.
once	Use with script. Prevent the script from several executions.
getob	Deprecated. Use with script. Retrieves texts passed to `echo` and puts them to the field's place

TBS Block's parameters:

Parameter	Summary
block	Defines the block's bounds.
extend	Extends the block's bounds upon several successive Html tags.
encaps	Extends the block's bounds upon several encapsulated Html tags.
comm	Extends the block's bounds up to the Commentary tag that surround it.
nodata	Indicates the section that is displayed when there is no data in the data source.
headergrp	Indicates a header section that is displayed when the value of a column changes.
footergrp	Indicates a footer section that is displayed when the value of a column changes.
splittergrp	Indicates a splitter section that is displayed when the value of a column changes.
parentgrp	Indicates a parent section that is displayed when the value of a column changes.
serial	Indicates a section that contains a series of several records.
p1	Sends a parameter to the dynamic query for the data source.

onsection	Executes a Php user function to modify the section merging.
tplvars	Use with onload fields only. Define template variables.
when	Use with onload or onshow. Displays the section when the condition is verified.
default	Use with onload or onshow. Displays the section when no section is displayed.
several	Use with when. Indicate that several blocks of the group can be displayed.

Fields and parameters for Navigation bar:

Fields	Summary
nav.page	Displays the number of a page.
nav.curr	Displays the number of the current page.
nav.first	Displays the number of the first page (allways 1).
nav.prev	Displays the number of the previous page.
nav.next	Displays the number of the next page.
nav.last	Displays the number of the last page (-1 if unknown).

Parameter	Summary
currpage	Indicates a section that is displayed only for the current page.
endpoint	Returns an empty string if the current page is the first page or the last page.
navpos	Indicates how the navigation bar is positioned compared to the current page number.
navsize	Indicates the number of page to display.
pagemin	Indicates the number of the first page.

Names of Special Fields and Blocks:

Name	Summary
val	The keyword [val] can be used in field's parameters to represent the field's value.
var.*	Displays a Php variable.
var..*	Displays information about the TinyButStrong System.
#	Virtual column name for a block. It displays the record's number.
$	Virtual column name for a block. It displays the record's key if the data source is a Php Array.
onload	Automatic field or block, merged when the template is loaded.
onshow	Automatic field or block, merged when the template is shown.