

**Forms multipages  
OVERVIEW  
New Release 2013**

- Possibility to have several styles to display error messages according to their severity.
- Erasing immediate error message, when editing the wrong information by the user and redraw possible explanatory message.
- "ModeAutoReconnect" for using the SQL (stored procedures) of Class SQL.
- Demonstration programs to chain multiple stored procedures with different SQL Driver 'Demo SQL with multiples Stored Procedures'
- Function that returns an array results form cf 4-1-1-7 ResultsForms
- Addition of new features for the management of forms. (array data)

**INTEGRATED DEVELOPPEMENT FOR FORMS AND SQL WITH PHP, INCLUDING ADVANCED PHP TOOLS .**

**Available with php 5.3.0 alpha4**

It is the development environment php5 for process forms with more pages. It consists of an integrated set of tools. It allows personalized controls on each fields by using callback functions.

OBJECT:

- Reduce code HTML
- Travel information from screen to screen
- Custom forms fields
- Construct, fit forms
- Allow interface with user control for fields

**Note:**

**It is advised to carefully read the scripts, including scripts examples of creating forms and functions users. They are particularly commented.**

**Keywords:**

'|','!','-','=','?','&','!', 'p\_back=','p\_backf'

**1- CLASS SQL**

**2- ERROR HANDLING**

**3- FORMS**

**3- 1 DEMONSTRATION**

*3-1-1 - Forms Adress Postal*

*3-1-2 - Forms Send Email*

*3-1-3 Forms More Pages*

*3-1-4 DownLoad*

*3-1-5 Encoding -Decoding*

*3-1-6 Test SQL*

**3-2 Syntax Scripts Form**

**4- FORMS CLASS OVERVIEW**

**4-1 CLASS 'OBJ\_DAEMONFORM'**

*4-1-1 Functions*

**4-2 CLASS 'OBJ\_FORM'**

*4-2-1 Functions*

*4-2-2 Fields*

*4-2-2-1 Attributess*

*4-2-2-2 Functions*

**5 - FOLDERS**

## Forms multipages

### 1 - CLASS SQL

#### OVERVIEW

Each hosting web sites has its own drivers SQL. Therefore, an application SQL is not always a portable host to another. In the expectation that PDO become fully operational et become a standart, the PHP Data Object (Obj\_SQLDB) provides the same functions to issue queries and fetch data for drivers PDO, Postgres, MySQLi and MySQL.

**Of course, it is still possible with Obj\_SQLDB to use the functions specific to each of these drivers. For MySQL and PgSQL, you can use specifics function by call function GetLinkDB() for obtain resource connection.**

Obj\_SQLDB requires the new OO features in the core of PHP 5, and so will not run with earlier versions of PHP.

All requests (except MySQL) are executed by use 'prepare' statement and return a statement handle to be used for further operations on this statement.

Two difficulties were encountered with driver mysql:

- Failure to free query for prepared statement that execute a stored procedure returning more results set with PDO or MySQLi or MySQL, except with version php 5.3 and new driver mysqlnd for PDO.

This query is proceeding normally and returns expected results, but if it is followed by another query, the execution of this second request fails because first query is not freed. This is because with mysql a stored procedure can returns multiple results set.

With version php < 5.3

- MySQL
  - Use function mysql\_more\_results() is not a valid callback !
- MySQLi
  - This difficulty is resolved by use mysqli\_more\_results()
- PDO\_MySQL
  - Function: nextRowset message:SQLSTATE[HYC00]: Optional feature not implemented !

With version php 5.3.0 alpha4 and new driver mysqlnd

- MySQL
  - Use function mysql\_more\_results() is not a valid callback (invalid function name) !
- MySQLi
  - Packets out of order !
- PDO\_MySQL
  - Implementation is proceeding normally

Where such an error occurs, user must create a new instance representing a connection to a database. To do this, the user must run the function object "DataBase Reconnect()" (cf 1-2-2 object DataBase->Reconnect) which makes the reconnection with the server and rebuild all prepare statements, without the user having to re-run the functions " object DataBase->Prepare()". Then, the buffer being freed, User must re-execute the instruction that has not worked.

**The user can use the mode "auto\_reconnect" (mode by default) by setting attribute " object DataBase->ModeAutoReconnect". In this case, if such an error occurs, Obj\_SQLDB performs itself a new instance representing a connection to a database and rebuild all prepares statements" and re-run the instruction that has not worked, without any user intervention. The user can be notified of such an incident by setting the mode indicator of attribute " object DataBase->ModeAutoReconnect" .**

**The demonstration of such mode is in script 'Demo SQL with multiples Stored Procedures'**

**Note: 5.3.0 alpha4 is a version being updated by PHP**

## Forms multipages

- **Inability to recover INOUT parameters from a stored procedure with driver pdo\_mysql, mysqli, mysql:**

This difficulty is resolved by performing the following treatments without any user intervention :

- change the request query (replacement parameters OUT and INOUT by variables SQL ((@ Name)
- call query ('set @Name1 = xxx), query ('set @Name2 = xxx)... to assign value to these variables
- call query ('select @Name1, @Name2...')->fetch\_row() or ('select @Name1, @Name2...')->fetch() for recovery new value

- **PDO, Postgres, MySQLi and MySQL run with mode exception.**

All the querPrepares the SQL query and return a string index to be used for further operations on the statement

- Samples in script 'Page\_DemoSQL.php' with 'CallUserFunc\_DemoSQL.php'  
and 'Page\_DemoStoredProcedure.php' with '\_CallUserFunc\_DemoStoredProcedure'

## Forms multipages

### 1-1 Predefined Constants:

These constants are the same that PDO. If PDO is not implemented on your WEB hosting, a virtual class is created with these constants. (<http://fr.php.net/manual/en/pdo.constants.php>)

- **PDO::FETCH\_ASSOC**  
Specifies that the fetch method shall return each row as an array indexed by column name as returned in the corresponding result set. If the result set contains multiple columns with the same name, PDO::FETCH\_ASSOC returns only a single value per column name
- **PDO::FETCH\_NUM**  
Specifies that the fetch method shall return each row as an array indexed by column number as returned in the corresponding result set, starting at column 0.
- **PDO::FETCH\_BOTH**  
Specifies that the fetch method shall return each row as an array indexed by both column name and number as returned in the corresponding result set, starting at column 0.
- **PDO::FETCH\_OBJ**  
returns an anonymous object with property names that correspond to the column names returned in your result set
- **PDO::FETCH\_LAZY**  
combines PDO::FETCH\_BOTH and PDO::FETCH\_OBJ, creating the object variable names as they are accessed
- **PDO::FETCH\_BOUND**  
returns TRUE and assigns the values of the columns in your result set to the PHP variables to which they were bound with the
- **PDO::PARAM\_BOOL** represents a boolean data type
- **PDO::PARAM\_NULL** represents the SQL NULL data type
- **PDO::PARAM\_INT** represents the SQL INTEGER data type
- **PDO::PARAM\_STR** represents the SQL CHAR, VARCHAR, or other string data type
- **PDO::PARAM\_LOB** represents the SQL large object data type
- **PDO::PARAM\_INPUT\_OUTPUT** specifies that the parameter is an INOUT parameter  
for a stored procedure. You must bitwise-OR this value with an explicit PDO::PARAM\_\* data type
- **PDO::PARAM\_PARAM\_INPUT\_OUTPUT** specifies that the parameter is an INOUT parameter
- **PDO::PARAM\_STMT** (integer) Represents un type de jeu de resultat. Not currently supported by any drivers.
- **PDO::FETCH\_ORI\_NEXT** Fetch the next row in the result set. Valid only for scrollable cursors.
- **PDO::FETCH\_ORI\_PRIOR** Fetch the previous row in the result set. Valid only for scrollable cursors.
- **PDO::FETCH\_ORI\_FIRST** Fetch the first row in the result set. Valid only for scrollable cursors.
- **PDO::FETCH\_ORI\_LAST** Fetch the last row in the result set. Valid only for scrollable cursors.
- **PDO::FETCH\_ORI\_ABS** Fetch the requested row by row number from the result set. Valid only for scrollable cursors.
- **PDO::FETCH\_ORI\_REL** Fetch the requested row by relative position from the current position of the cursor in the result set. Valid only for scrollable cursors.
- **PDO::CURSOR\_FWDONLY** Create a PDOStatement object with a forward-only cursor. This is the default cursor choice, as it is the fastest and most common data access pattern in PHP.
- **PDO::CURSOR\_SCROLL** Create a PDOStatement object with a scrollable cursor. Pass the PDO::FETCH\_ORI\_\* constants to control the rows fetched from the result set.

//ATTRIBUTS

- **PDO::MYSQL\_ATTR\_USE\_BUFFERED\_QUERY**
- **PDO::ATTR\_CURSOR**

## 1-2 Functions

### 1-2-1 Obj\_SGDB\_Connect( \$ClassSGBD = '', \$BaseName = NULL, \$Host = NULL, \$User = NULL                           , \$PassWord = NULL, \$DriverOptions = array() )

Creates a object with an instance representing a connection to a database

For MySQL and PgSQL, you can use specifics function by call function GetLinkDB() for obtain resource connection.

#### Parameters:

- **\$ClassSGDB**
  - o **Empty string by default**  
In this case, if PDO with mysql is available, PDO with mysql is used  
or if PDO with pgsql is available, PDO with pgsql is used  
or if PgSQL is available, PgSQL is used  
or if MySQLi is available, MySQLi is used  
or MySQL is used.
  - o **'PDO\_MySQL' or 'PDO\_PgSQL' or 'PgSQL' or 'MySQLi' or 'MySQL'** (case-insensitive)
- **\$BaseName**
  - o **Empty string by default**  
In this case, the default database to be used
  - o **Specify the database to be used when performing queries**
- **\$Host**
  - o => <http://fr.php.net/manual/en/pdo.construct.php>
- **\$User**
  - o **Empty string by default**  
In this case, the default username is used
  - o **The user name**
- **\$PassWord**
  - o **Empty string by default** (no password)
  - o **Password**
- **\$DriverOptions=array()** (used if PDO) <http://fr.php.net/manual/en/pdo.construct.php>

#### Return:

Returns an object DataBase Obj\_PDO, or Obj\_PgSQL or Obj\_MySQLi or MySQL or exception if error encountered.

#### Note:

Once the object creates, the user can set the attribute ModeAutoReconnect:

MyObjectDataBase->ModeAutoReconnect = array(TRUE or FALSE, 0 or 1 or 2);

TRUE for activate mode auto reconnect (cf see above Failure to free query)

0 => running in silent mode if reconnection is being

1 => display simply alert message if reconnection is being

2 => display alert message with driver specific error code and driver specific error string

## Forms multipages

### 1-2-2 Object DataBase->GetLinkDB()

Avalaible for MySQL and PgSQL for obtain resource connection.

### 1-2-3 Object DataBase->Reconnect()

- Create an new instance representing a connection to a database for an existing object DataBase.
- Rebuild all prepares statements

### 1-2-4 Object DataBase->PrepareReket( \$ArrayReket, \$ArrayListBindParam = array() )

- PDO or MySQLi or PgSQL
  - Prepares an SQL statement to be executed by the PDOStatement::execute() or mysql\_stmt\_execute() or pg\_execute methods.
  - Binds a PHP variables to a corresponding question mark placeholder in the SQL statement that was use to prepare the statement. These variable are bound as a reference and will only be evaluated at the time that Object DataBase->Launch is called.
- MySQL
  - Savues the settings parameters for mysql\_query()

#### Parameters:

- **\$ArrayReket( \$TableName, \$QueryString, \$DriverOptions = array() )**
  - **\$TableName**  
Specify the name of table to be used when performing queries
  - **\$QueryString**  
This must be a valid SQL statement for the target database server
- **Note:**  
**For query or stored procedures with MySQL, The parameters must represented with '?' (markquestion)**  
**For query or functions with Postgres, The parameters must represented either with '?' (markquestion) either with '\$1', '\$2', etc...**
- **\$DriverOptions (if use PDO) <http://fr.php.net/manual/en/pdo.prepare.php>**  
This array holds one or more key=>value pairs to set attribute values for the PDOStatement object that this method returns. You would most commonly use this to set the *PDO::ATTR\_CURSOR* value to *PDO::CURSOR\_SCROLL* to request a scrollable cursor. Some drivers have driver specific options that may be set at prepare-time.
- **\$ArrayListBindParam( list array)**  
**for each list array**
  - **\$Variable**  
Name of the PHP variable to bind to the SQL statement parameter
  - **\$DataType**  
Explicit data type for the parameter using the *PDO::PARAM\_\** constants. Defaults to **PDO::PARAM\_STR**. To return an INOUT parameter from a stored procedure, use the bitwise OR operator to set the *PDO::PARAM\_INPUT\_OUTPUT* bits for the *data\_type* parameter.
  - **\$Length**  
Length of the data type *PDO::PARAM\_STR*. To indicate that a parameter is an OUT parameter from a stored procedure, you must explicitly set the length.
  - **\$DriverOptions (if use PDO) <http://fr.php.net/manual/en/pdostatement.bindparam.php>**

#### Return:

Returns an string that contains the index of request or exception if error encountered.

## Forms multipages

### 1-2-5 Object DataBase->Launch( \$IdentReket )

This function is optional. It helps to know oneself number rows obtained after a SELECT or the number of rows affected by an INSERT, UPDATE or DELETE or stored procedure or function.

If PDO or Postgres or MySQLi, runs a PDOStatement::execute() or a mysqli\_stmt\_execute().

If MySQL runs a mysql\_query()

#### Parameters:

- **\$IdentReket**

String that specify the index of request returned by function PrepareReket()

#### Return:

Returns number rows obtained after a SELECT or number of rows affected by INSERT, UPDATE or DELETE or exception if error encountered.

This number is not always correct with PDO and MySQL for rows affected.

### 1-2-6 Object DataBase->FetchResults( \$IdentReket, \$FetchStyle = PDO::FETCH\_ASSOC,

**\$CursorOrientationOrRow = NULL, \$Offset = NULL** )

Fetch one results for a index of request. This function runs Launch() if not called by user, before fetch results.

#### Parameters:

- **\$IdentReket**

String that specify the index of request returned by function PrepareReket()

- **\$FetchStyle**

Controls how the next row will be returned to the caller. This value must be one of the *PDO::FETCH\_\** constants, defaulting to *PDO::FETCH\_ASSOC*.

- *PDO::FETCH\_ASSOC*: returns an array indexed by column name as returned in your result set

- *PDO::FETCH\_BOTH* (default): returns an array indexed by both column name and 0-indexed column number as returned in your result set

- *PDO::FETCH\_NUM*: returns an array indexed by column number as returned in your result set, starting at column 0

- *PDO::FETCH\_OBJ*: returns an anonymous object with property names that correspond to the column names returned in your result set

- *PDO::FETCH\_LAZY*: combines *PDO::FETCH\_BOTH* and *PDO::FETCH\_OBJ*, creating the object variable names as they are accessed

- **\$CursorOrientationOrRow (if PDO)** <http://fr.php.net/manual/en/pdostatement.fetch.php>.

if **postgres**, Row number in result to fetch. Rows are numbered from 0 upwards. If omitted, next row is fetched.variable names as they are accessed

- **\$Offset (if PDO)** <http://fr.php.net/manual/en/pdostatement.fetch.php>.

#### Return:

Returns a array of one rows has be fetched or a empty array if no more rows or exception if error encountered.

## Forms multipages

**1-2-7 Object DataBase->FetchResultsMore( \$IdentReket, \$Limit = 0,  
\$FetchStyle = PDO::FETCH\_ASSOC, \$IndexColumnOrCursorOrientation = NULL,  
\$CtorArgsOrOffset = NULL)**

Fetch more results for a index of request. This function runs Launch() if not called by user, before fetch results.

### Parameters:

- **\$IdentReket**  
String that specify the index of request returned by function PrepareReket()
- **\$Limit**  
Specify the count number of rows fetched. If value of \$Limit is 0, all rows are returned. If value of \$Limit is 1, this equates to function FetchResults().
- **\$FetchStyle**  
Controls how the next row will be returned to the caller. This value must be one of the *PDO::FETCH\_\** constants, defaulting to *PDO::FETCH\_ASSOC*.
  - *PDO::FETCH\_ASSOC*: returns an array indexed by column name as returned in your result set
  - *PDO::FETCH\_BOTH* (default): returns an array indexed by both column name and 0-indexed column number as returned in your result set
  - *PDO::FETCH\_NUM*: returns an array indexed by column number as returned in your result set, starting at column 0
  - *PDO::FETCH\_OBJ*: returns an anonymous object with property names that correspond to the column names returned in your result set (available only with **\$Limit != 0**)
  - *PDO::FETCH\_LAZY*: combines *PDO::FETCH\_BOTH* and *PDO::FETCH\_OBJ*, creating the object variable names as they are accessed (available only with **\$Limit != 0**)
- **\$IndexColumnOrCursorOrientation (if PDO)** <http://fr.php.net/manual/en/pdostatement.fetchAll.php>.
- **CtorArgsOrOffset (if PDO)** <http://fr.php.net/manual/en/pdostatement.fetchAll.php>.

### Return:

Returns a array of \$Limit rows fetched or all rows fetched if value of \$Limit is 0 or greater than all rows fetched. If error encountered, return exception.

## **Forms multipages**

### **1-2-8 Object DataBase->FreeResult( \$IdentReket )**

This function is optional. It Frees stored result memory for the request pointed par string index \$IdentReket.

#### **Parameters:**

- **\$IdentReket**  
String that specify the index of request returned by function PrepareReket()

#### **Return:**

**Returns if error encountered, return exception.**

### **1-2-9 Object DataBase->EraseReket( \$IdentReket )**

This function erase the request pointed par string index \$IdentReket and frees memory allowed.

#### **Parameters:**

- **\$IdentReket**  
String that specify the index of request returned by function PrepareReket()

#### **Return:**

**Returns if error encountered, return exception.**

### **1-2-10 Object DataBase->Close\_Obj\_SGDB( )**

This function is optional. This function close the connexion and frees memory allowed by all request.  
By default, the connexion is closed by the destructor \_\_destruct().

### **1-2-11 Object DataBase->CreateTable( \$TableName, \$TableDescript )**

This function create a table in DataBase.

#### **Parameters:**

- **\$TableName**  
String that specify the name of table to create
- **\$TableDescript**  
Array that specify the description of the table (attributes, index, etc cf SQL) table to create

#### **Return:**

**Returns TRUE or exception if error encountered.**

### **1-2-12 Object DataBase->DropTable( \$TableName )**

This function drops a table in DataBase.

#### **Parameters:**

- **\$TableName**  
String that specify the name of table to create

#### **Return:**

**Returns TRUE or exception if error encountered.**

## Forms multipages

**1-2-13 Object DataBase->GetFullInfoFields( \$TableName , \$ListFields = array()**

Get an array where each line is an associative array that contains the full infos on fields of table \$TableName, specified in array \$ListFields. If \$ListFields is a empty array , all fields of table are returned

These infos are obtained from the table schema

## Parameters:

- **\$TableName**  
String that specify the name of table
  - **\$ListFields**  
Array that contains list of fields

### Return:

**Returns** Array of results

## Forms multipages

### 1-2-14 Object DataBase->GetMiddleInfoFields( \$TableName , \$ListFields = array())

Get an array where each line is an associative array that contains the middle infos on fields of table \$TableName, specified in array \$ListFields. If \$ListFields is a empty array , all fields of table are returned  
These infos are obtained from the table schema

```
//      for each fields, keys array associative are:  
//          'COLUMN_NAME'  
//          , 'ORDINAL_POSITION'  
//          , 'COLUMN_DEFAULT'  
//          , 'IS_NULLABLE',  
//          , 'DATA_TYPE'  
//          , 'CHARACTER_MAXIMUM_LENGTH'  
//          , 'CHARACTER_OCTET_LENGTH'  
//          , 'NUMERIC_PRECISION'  
//          , 'NUMERIC_PRECISION_RADIX'  
//          , 'NUMERIC_SCALE'  
//          , 'CHARACTER_SET_NAME'  
//          , 'DATE_TIME_PRECISION'  
//          , 'COLLATION_NAME'  
//          , 'COLUMN_TYPE'  
//          , 'COLUMN_KEY'
```

#### Parameters:

- **\$TableName**  
String that specify the name of table
- **\$ListFields**  
Array that contains list of fieds

#### Return:

Returns Array of results

## **Forms multipages**

### **1-2-15 Object DataBase->LoadTable( \$TableName, \$FileData, \$DescriptData )**

This function load data into table from a file that contains data.

#### **Parameters:**

- **\$TableName**  
String that specify the name of table to load
- **\$FileData**  
Specify the full filename of data
- **\$DescriptData**  
Array that descript format of data in \$FileData (cf SQL).

#### **Return:**

Returns TRUE or exception if error encountered.

## Forms multipages

### 2 – ERRORS HANDLING (SCRIPTS 'DEFERROR.PHP' AND 'DEFTTRACE.PHP')

#### Preamble:

**It is essential to be able to quickly find its mistakes**

It is possible to run with or without exceptions (default: Obj\_Error::\$ModeException = TRUE).

#### If you work without exceptions:

You must set Obj\_Error::\$ModeException = FALSE.

In this case, the function **Obj\_error::Run(\$Message, \$CodeError, \$UserFunc = NULL)** call the function **Obj\_Error::MonitorError(\$Message, \$CodeError, \$File, \$Line, \$Context)**.

The 'UserFunc' is defined user function to generate a particular response to an error.

If runtime errors encountered, **Obj\_Error::MonitorError(...)** is called.

#### If you work with exceptions (default mode):

Obj\_Error::\$ModeException = TRUE.

The class **Obj\_Exception** defined by extending the built-in exception.

You can generate an exception by three methods:

- **Obj\_Error::ThrowException(\$MsgError, \$Code, \$File = NULL, \$Line = NULL, \$UserFuncThrow = NULL, \$UserFuncUncaught = NULL)**

- **new Obj\_Exception(\$Message, \$Code, \$File = NULL, \$Line =NULL, \$UserFuncThrow = NULL, \$UserFuncUncaught = NULL)**

- **Obj\_Error::Exception(\$Message, \$Code, \$File = NULL, \$Line =NULL, \$UserFuncThrow = NULL, \$UserFuncUncaught = NULL)**

Function **UserFuncThrow(\$ObjectException)** sets a user-defined error function which be called during each catch construct exception.

If an exception is not caught, a function Fatal Error will be issued with an "Uncaught Exception ..." message, unless a handler has been defined with **\$UserFuncUncaught(\$ObjectException)** .

#### NOTE:

**For define your own way of handling errors during runtime for a critical error, you must set:**

- **Obj\_Error::\$UserFuncThrow = 'myUserFuncThrow';**

- **Obj\_Error::\$UserFuncUncaught = 'myUserFuncUncaught';**

**In this case, it is unnecessary to specify these functions in Obj\_Error::Exception(...) and new Obj\_Exception(....)**

**The class Obj\_Exception use two function: Caught(\$UserFunc = NULL) and Uncaught():**

- **Caught(...)** is called in catch block. The 'UserFunc' can be defined for a particular custom response.

- **Uncaught(...)** is called when the exception is not caught. If specify, UserFunccaught is called.

**A User defined Exception class can be defined by extending the built-in Obj\_Exception class.**

**In this case, it is highly recommended that it also call parent::\_\_construct() to ensure all available data has been properly assigned.**

## Forms multipages

For make trace of error, the function `Obj_Trace::WriteTrace` write log file informations on error, nature, script name and numer line and a dump of all functions called with their variables parameters and all objets and variables used with their values. It is possible to display backtrace in mode reverse by set

`Obj_Error::$BackTrace = TRUE;`

### example:

Legende:

```
'$object ##' indique debut de l'objet N° ##
'}## $object' indique fin de l'objet N° ##
'$object ## already traced (recursive) !! indique object N°## deja decrit
'$array ##(' indique debut du tableau N° ##
')## $array' indique fin du tableau N° ##
```

26/06/2008 08:32:08

UncaughtException Laurini  
ERROR N°1024->[E\_USER\_NOTICE]  
SCRIPT: 'C:\Program Files\EasyPHP 2.0b1\www\geography\Geography.php' AT LINE: 184

TraceAsString (reverse) :

```
#4 {main}
#3 C:\Program Files\EasyPHP 2.0b1\www\geography\TestGeography.php(46): GetCommune('91660')
#2 C:\Program Files\EasyPHP 2.0b1\www\geography\Geography.php(284): Obj_SGDB->BuildReket('codes_postaux_f...', 'SELECT
code_pos...')
#1 C:\Program Files\EasyPHP 2.0b1\www\geography\Geography.php(184): Obj_SGDB->LaunchException('N?: 0 => prep...', 1024,
'C:\Program File...', 184)
#0 C:\Program Files\EasyPHP 2.0b1\www\geography\Geography.php(101): Obj_Error::ThrowException('N?: 0 => prep...', 1024,
'C:\Program File...', 184)
```

TRACE (reverse) :

```
+ function UncaughtException (
+   object :: 'Obj_Exception'  $object 1{
+     $FuncUserThrowException = $array 1(
+       [0] = 'Obj_SGDB'
+       ,[1] = 'CheckUpSGDB' )$array 1
+     ,$FuncUserUncaughtException = 'FuncGeoUnCaught'
+     ,$BackTrace_Exception = $array 1(
+       ['+] => $array 2(
+         ['file'] => 'C:\Program Files\EasyPHP 2.0b1\www\geography\TestGeography.php'
+         ,['line'] => 46
+         ,['function'] => 'GetCommune'
+         ,['args'] => $array 3(
+           [0] = '91660' )$array 3 )$array 2
+       ,['++'] => $array 2(
+         ['file'] => 'C:\Program Files\EasyPHP 2.0b1\www\geography\Geography.php'
+         ,['line'] => 284
+         ,['function'] => 'BuildReket'
+         ,['class'] => 'Obj_SGDB'
+         ,['object'] => object :: 'Obj_SGDB'  $object 2{
+           $BaseName = 'pierrelaurfr'
+           ,$CountReket = 0
+           ,$TabReket = $array 3( )$array 3
+           ,,$JokerLike = '%' }$object 2
+         ,['type'] => '->'
+         ,['args'] => $array 3(
+           [0] = 'codes_postaux_france'
+           ,[1] = 'SELECT code_postal , commune FROM codes_postaux_france WHERE code_postal = ? ORDER BY code_postal,
commune' )$array 3 )$array 2
```

## Forms multipages

```
+ ,['+++] => $array 2(
+   ,['file'] => 'C:\Program Files\EasyPHP 2.0b1\www\geography\Geography.php'
+   ,['line'] => 184
+   ,['function'] => 'LaunchException'
+   ,['class'] => 'Obj_SGDB'
+   ,['object'] => object :: 'Obj_SGDB'  $object 2 already traced (recursive) !!
+   ,['type'] => '>'
+   ,['args'] => $array 3(
+     [0] = 'N°: 0 => prepare '
+     ,[1] = 1024
+     ,[2] = 'C:\Program Files\EasyPHP 2.0b1\www\geography\Geography.php'
+     ,[3] = 184 )$array 3 )$array 2
+ ,['++++'] => $array 2(
+   ,['file'] => 'C:\Program Files\EasyPHP 2.0b1\www\geography\Geography.php'
+   ,['line'] => 101
+   ,['function'] => 'ThrowException'
+   ,['class'] => 'Obj_Error'
+   ,['type'] => '::'
+   ,['args'] => $array 3(
+     [0] = 'N°: 0 => prepare '
+     ,[1] = 1024
+     ,[2] = 'C:\Program Files\EasyPHP 2.0b1\www\geography\Geography.php'
+     ,[3] = 184 )$array 3 )$array 2
+ ,['+++++'] => $array 2(
+   ,['file'] => 'C:\Program Files\EasyPHP 2.0b1\www\def\DefError.php'
+   ,['line'] => 234
+   ,['function'] => 'Obj_Exception'
+   ,['class'] => 'Obj_Exception'
+   ,['object'] => object :: 'Obj_Exception'  $object 1 already traced (recursive) !!
+   ,['type'] => '>'
+   ,['args'] => $array 3(
+     [0] = 'N°: 0 => prepare '
+     ,[1] = 1024
+     ,[2] = 'C:\Program Files\EasyPHP 2.0b1\www\geography\Geography.php'
+     ,[3] = 184
+     ,[4] = Var is null
+     ,[5] = Var is null )$array 3 )$array 2 )$array 1
+ ,$_Context_Exception = Var is null
+ ,$_message = 'UncaughtException Laurini'
+ ,$_code = 1024
+ ,$_file = 'C:\Program Files\EasyPHP 2.0b1\www\geography\Geography.php'
+ ,$_line = 184 )$object 1
+
+ )
++"C:\Program Files\EasyPHP 2.0b1\www\def\DefError.php" => line : 211
++ function Uncaught (
++ )
++"C:\Program Files\EasyPHP 2.0b1\www\def\DefError.php" => line : 161
++ function BuildAndCallFuncUser (
++   'FuncGeoUnCaught'
++ )
++"C:\Program Files\EasyPHP 2.0b1\www\def\DefError.php" => line : 94
++ function call_user_func_array (
++   'FuncGeoUnCaught'
++   ,$array 1(
++     [0] = object :: 'Obj_Exception'  $object 1 already traced (recursive) !! )$array 1
++   ,
++ )
++ function FuncGeoUnCaught (
++   object :: 'Obj_Exception'  $object 1 already traced (recursive) !!
++ )
+++
+++
+)
```

## Forms multipages

DUMP GLOBALS:

```
$array1(
    ['GLOBALS'] => ['GLOBALS'] recursive'
    ,['_POST'] => $array2( )$array2
    ,['_GET'] => $array2( )$array2
    ,['_COOKIE'] => $array2( )$array2
    ,['_FILES'] => $array2( )$array2
    ,['_SERVER'] => $array2(
        ['TMP'] => 'C:/Program Files/EasyPHP 2.0b1/tmp'
        ,['HTTP_ACCEPT'] => 'image/gif, image/x-bitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash,
application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */*'
        ,['HTTP_REFERER'] => 'http://127.0.0.1/geography'
        ,['HTTP_ACCEPT_LANGUAGE'] => 'fr'
        ,['HTTP_UA_CPU'] => 'x86'
        ,['HTTP_ACCEPT_ENCODING'] => 'gzip, deflate'
        ,['HTTP_USER_AGENT'] => 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Mozilla/4.0 (compatible; MSIE 6.0;
Windows NT 5.1; SV1) ; Mozilla/4.0(Compatible Mozilla/4.0(Compatible-EmbeddedWB 14.59 http://bsalsa.com/ EmbeddedWB- 14.59
from: http://bsalsa.com/ ; .NET CLR 1.1.4322)'
        ,['HTTP_HOST'] => '127.0.0.1'
        ,['HTTP_CONNECTION'] => 'Keep-Alive'
        ,['PATH'] => 'C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Program Files\Fichiers
communs\Adaptec Shared\System;C:\Program
Files\QuickTime\QTSystem\;C:\PROGRA~1\EASYPH~1.0B1\Apache\bin;C:\PROGRA~1\EASYPH~1.0B1\PHP5'
        ,['SystemRoot'] => 'C:\WINDOWS'
        ,['COMSPEC'] => 'C:\WINDOWS\system32\cmd.exe'
        ,['PATHEXT'] => '.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH'
        ,['WINDIR'] => 'C:\WINDOWS'
        ,['SERVER_SIGNATURE'] => "
        ,['SERVER_SOFTWARE'] => 'Apache/2.2.3 (Win32) PHP/5.2.0'
        ,['SERVER_NAME'] => '127.0.0.1'
        ,['SERVER_ADDR'] => '127.0.0.1'
        ,['SERVER_PORT'] => '80'
        ,['REMOTE_ADDR'] => '127.0.0.1'
        ,['DOCUMENT_ROOT'] => 'C:/Program Files/EasyPHP 2.0b1/www'
        ,['SERVER_ADMIN'] => 'admin@localhost'
        ,['SCRIPT_FILENAME'] => 'C:/Program Files/EasyPHP 2.0b1/www/geography/TestGeography.php'
        ,['REMOTE_PORT'] => '1318'
        ,['GATEWAY_INTERFACE'] => 'CGI/1.1'
        ,['SERVER_PROTOCOL'] => 'HTTP/1.1'
        ,['REQUEST_METHOD'] => 'GET'
        ,['QUERY_STRING'] => "
        ,['REQUEST_URI'] => '/geography/TestGeography.php'
        ,['SCRIPT_NAME'] => '/geography/TestGeography.php'
        ,['PHP_SELF'] => '/geography/TestGeography.php'
        ,['REQUEST_TIME'] => 1214461927 )$array2
    ,['ErrorMonitoring'] => Var is null
    ,['ExceptionMonitoring'] => Var is null
    ,['CodePostal'] => '91660'
    ,['_OBJ_SGDB_'] => object :: 'Obj_SGDB' $object2 already traced (recursive) !! )$array1
```

## Forms multipages

### 3 - FORMS

You must run the script /from/Sample\_More\_Forms.php for starting your application.

#### CALLING PARAMETERS :

<b>P_Proc = string</b>	String for identify application
<b>P_ScriptAfterValid = string (optional)</b>	String full filename user to be called when form is validated A script to send the results of form is available ("UserSendEmail.php") *1
<b>P_UserFuncControl = string (optional)</b>	String full filename user script which contains the callback functions These functions are called after field button submit. They allows a control personalized It contains also the functions to be called where field link error no exists. *1
<b>P_PagesEnum = string</b>	String contains list of pages for each page: title page; full filename page;option  option => 'on' or 'off' or 'opt' 'on' for page obligatory 'opt' for page optional 'off' for page hidden or removed  character ';' is the separator between each pages and ';' is the separator between title page, full_filename and option  example: " title_page_1 ; full_filename_page_1;on , title_page_2; full_filename_page_2; opt , title_page_3 ;full_filename_page_3;on , title_page_4; full_filename_page_4; on *1

**\*1 Lists pages, callback and validate functions may not be processed at the remote server**

These scripts below are writed for the demonstration:

#### Scripts P\_ScriptAfterValid

- "UserValidForms.php"
- "UserSendEmail.php "

#### Scripts P\_UserFuncControl

- "CallUserFunc\_Forms"
- "Call UserFunc\_AdrEmail.php"
- "Call UserFunc\_Crypt"

## Forms multipages

### 3-1 Demonstration

#### 3-1-1 - *Forms Incription*

Two pages: inscript.php, Result\_inscript.php and UserFuncs: CallUserFunc\_inscript.php, UserValidInscript.php

##### Script *P\_ScriptAfterValid UserValidInscript.php*

- Display the results of the form

##### Script *P\_UserFuncControl CallFuncUser\_inscript.php* proceeds controls below:

- CheckUp field 'Identifiant' [first page]
- CheckUp field 'Password' and confirm 'Password'[first page]
- CheckUp field 'Civilite' [first page]
- CheckUp field 'Birth dat' [first page]
- CheckUp field 'Lastname' and 'Firstname[first page]
- CheckUp field 'Phone fixe' [first page]
  - ChecUp for phone number (USA, Canada, France and DOM-TOM, some countries of the CEE)
- CheckUp field 'Phone Mobile' [first page]
  - ChecUp for phone number (France and DOM-TOM, Spain)
- CheckUp field 'Email' and confirm'Email' [first page]
- CheckUp field 'Lastname' and 'Firstname' [first page]
- CheckUp field 'Adresse\_1' and 'Adresse\_2' [first page]
- CheckUp field 'Ville' [first page]
  - ChecUp and select cities for France and DOM-TOM
- CheckUp field 'Code\_Postal' [first page]
  - ChecUp and select codes for France and DOM-TOM
- CheckUp field 'Pays' [first page]
  - Select countries of the world

#### 3-1-2 - *Forms Send Email*

Two pages: GetAdrEmail.php, Contact.php and UserFuncs: CallUserFunc\_AdrEmail.php, UserSendEmail.php

##### Script *P\_ScriptAfterValid UserSendEmail.php*

- Send the results of the form (mail)

##### Script *P\_UserFuncControl CallFuncUser\_AdrEmail.php* proceeds controls below:

- CheckUp field 'Adr\_Email' [first page]
- CheckUp field 'Key\_for\_Crypt' [first page]
- CheckUp field 'Email\_Crypted' [first page]

## **Forms multipages**

### **3-1-3 Forms More Pages**

Four pages: Page\_1.php, Page\_2.php, Page\_3.php, Page\_4.php and UserFuncs : CallUserFunc\_Forms.php, UserValidForms.php.

#### **Script P\_ScriptAfterValid UserValidForms.php**

- Display array the results of the form

#### **Script P\_UserFuncControl CallFuncUser\_Forms.php** proceeds controls below:

- CheckUp field 'Prenom' [first page] is different of 'paul'
- Assign, if field 'Nom' [first page] is 'Laur' and field 'Prenom' is blank, 'Prenom' to 'Pierre'
- CheckUp fields 'Select\_Liste\_1' [second page] with 'Select\_Liste\_2' [Page 2]  
and alert if if 'Select\_2' and 'choix\_2'
- CheckUp fields 'Select\_CheckBox\_1' [second page] no field link error  
begin with 'x' and 'Select\_Liste\_2' [Page 2]
- CheckUp and alert if fields 'Prenom' [first page] begin with 'x' and 'Select\_Liste\_2' [second page]  
is 'Select\_3'

### **3-1-4 DownLoad**

Tow pages: DownLoad.php and CounterDownLoad.php

### **3-1-5 Encoding -Decoding**

One page: crypt.php and UserFunc CallUserFunc\_Crypt.php

#### **Script P\_UserFuncControl CallFuncUser\_Crypt.php** proceeds controls below:

- CheckUp field 'Key\_for\_Crypt1' [first page]
- CheckUp field 'Texte\_Crypted' [first page]
- CheckUp field 'Texte\_To\_Crypt' [first page]

### **3-1-6 Demo SQL**

One page: Page\_DemoSQL.php and UserFunc CallUserFunc\_DemoSQL.php

### **3-1-7 Demo SQL with multiples stored procedures**

One page: Page\_DemoStoredProcedure.php and UserFunc CallUserFunc\_DemoStoredProcedure.php

### **3-1-8 SUDOKU**

exemple for use array values in a field TEXT

One page: Sudoku.php and UserFunc CallUserFunc\_Sudoku.php and process\_sudoku.php

## Forms multipages

### 3-2 Syntax For Scripts Form

#### It is necessary to have a script for each page of the form

- a script must always begin with command below

(after <head>) :

```
<?php
//Get object Obj_DaemonForm and get Obj_Form for the current page
$ObjForm = _GetDaemon(TRUE/FALSE->)FormsPage(_PAGE_);
// TRUE for mode debugging. Default, /FALSE is positionned
// _PAGE_ contains current page
if (!$ObjForm->AlreadyInitialized())      //test if already initialized
{ //Define fields (cf Objects for forms)

}
?>
<!—affichage du titre de la page --!>
<title><?php echo _GetDaemon()->GetTitlePage(_PAGE_); ?></title>
```

- After <body>:

```
<?php echo _GetDaemon()->StartForm(_PAGE_,[enctype]); ?>
example: enctype is optional ["multipart/form-data"] / ["text/plain"]
<?php echo _GetDaemon()->StartForm(_PAGE_,"multipart/form-data"); ?>
```

- within body view of fields

```
<?php echo $ObjForm->HTMLText("Field_Name"); ?>    for get and view field
<?php echo $ObjForm->HTMLTextMsg("Field_Name"); ?>  for view field link error (optional)
etc ...
```

- End body Forms

```
<?php echo _GetDaemon()->EndForm(); ?>
</body>
</html>
```

## Forms multipages

### 4- FORMS CLASS OVERVIEW

provides member functions for working with forms

#### 4-1 CLASS 'OBJ\_DAEMONFORM'

**OBJ\_DAEMONFORM** is the scheduler of system.

**USAGE:** The OBJ\_DAEMONFORM object provides member functions for working with forms

##### SYNOPTIC

FUNCTIONS	USAGE
AffichPage(\$page)	Affich page (\$page) if (\$page < 0) => abort forms
EndForm ()	Close form cf Syntax for script page (page 4)
FormsPage(\$Page = -1)	return reference Obj_Form for page \$Page If \$Page == -1 return reference array(Obj_Form) for all pages
GetAbort	return Abort (true or false)
GetCurrentPage()	return number current page
GetDaemon()	return object Obj_DaemonForm
GetNextPage(\$page)	return next page after \$page return false if no next page
GetTitlePage(\$Page)	return title page for \$Page
GetTypePage(\$Page)	return type for page \$Page ('on','off','opt')
ResultForms	return result forms (body and array files to send as a array ordered map)
SetAbort (\$type)	Set Abort (true or false) for abort forms
SetCurrentPage(\$Page)	Set current Page with page \$Page
SetTypePage(\$Page, \$Type)	Set type for page \$Page
StartForm (\$Page, \$EncType)	Start form cf Syntax for script page (page 5)

#### 4-1-1 FUNCTIONS

##### 4-1-1-1 AffichPage

Affich page \$page If \$page < 0 => abort forms  
example:

```
_GetDaemon()->AffichPage($page))
```

##### 4-1-1-2 EndForms

Close form (cf syntax for script page (page 5)  
example:

```
_GetDaemon()->EndForm()
```

##### 4-1-1-3 FormsPage

Return reference of Obj\_Form for page \$Page  
If \$Page == -1, return array of Obj\_Form for all pages  
example:

Get Obj\_Form for the second page  
\$ObjForms = \_GetDaemon()->FormsPage(2)

## **Forms multipages**

### **4-1-1-4 GetAbort**

Return Abort

example:

```
$CurrentPage = _GetDaemon()->GetAbort(): ;
```

### **4-1-1-5 GetCurrentPage**

Return number current

example:

```
$CurrentPage = _GetDaemon()->GetCurrentPage(): ;
```

## Forms multipages

### 4-1-1-6 \_GetDaemon

Return Obj\_DaemonForm. If not exists, it is created

### 4-1-1-7 GetNextPage

Return next page after \$page. return false if no next page

example:

```
$NextPage = _GetDaemon()->GetNextPage($Page);
```

### 4-1-1-8 GetTitlePage

Return title for Page \$Page (cf syntax for script page (page 5)

example:

```
$Title = _GetDaemon()->GetTitlePage($Page);
```

### 4-1-1-9 GetTypePage

Return type for page \$Page.

\$Type is either 'opt' for optional

'on' for active

'off' not active

example:

```
$Type = _GetDaemon()->GetTypePage($Page);;
```

### 4-1-1-10 ResultForms (cf 'Forms More pages – Page\_4)

Return Array result form

Array['body'] => string which contains all fields with their names and values

['file'][field-name] => filename uploaded

['file'][type] => type file

['array'] => array indexed by nd by number that contains all results form

For field 'Obj\_Text' (numeric or date or string)

The first element contains status (ok if 1) and either two elements or an array of two elements (if it is an array value)

the first element of these two elements contains characters entered and the second element of these two elements contains the resulting value converted

For field 'Obj\_Select', 'Obj\_Radio' and 'Obj\_CheckBox'

The first element contains status (ok if 1) and either one element or an array of one elements (if it is a multiple checked) which contains the label[s] of choice(s) checked

example:

indexed by fieldname      ['Nom']      => [0]=1      [1]=laur

indexed by number      [0]      => [0]=1      [1]=laur

indexed by namefield      ['Date-2']      => [0]=1      [1]=02/25/2007      [2]=2007-02-25 00:00:00 (date au format iso)

indexed by number      [1]      => [0]=1      [1]=02/25/2007      [2]=2007-02-25 00:00:00

['Montant-1']      => [0]=~Montant-1 non renseigné![1]="

[2]      => [0]=~Montant-1 non renseigné![1]="

['Montant-2']      => [0]=1      [1]=2222      [2]=2222 (valeur numérique)

[3]      => [0]=1      [1]=2222      [2]=2222 (valeur numérique)

['Select\_Liste1']      => [0]=1      [1]=[0]=Choix\_1 [1]=Choix\_3 [2]=Choix\_8 (multiple checked)

[4]      => [0]=1      [1]=[0]=Choix\_1 [1]=Choix\_3 [2]=Choix\_8

['Select\_Liste2']      => [0]=1      [1]='select\_2' (not multiple checked)

[5]      => [0]=1      [1]='select\_2'

['Prenom']      => [0]=1      [1]=[0]='Tom' [1]='Peter' (if several firstnames)

[6]      => [0]=1      [1]=[0]='Tom' [1]='Peter'

example:

```
$ResultForm = _GetDaemon()->ResultForms();;
```

## Forms multipages

### 4-1-1-11 SetAbort

Set Abort

example:

```
_GetDaemon()->SetAbort($type); ;
```

### 4-1-1-12 SetCurrentPage

Set current page with \$Pages

example:

```
_GetDaemon()->SetCurrentPage($Page); ;
```

### 4-1-1-13 SetTypePage

Set type page \$Pages with \$Type (cf GetTypePage)

example:

```
_GetDaemon()->SetTypePage($Page, $Type); ;
```

### 4-1-1-14 StartForm

Start form for page \$Pages. the second parameter is optional

(cf syntax for script page (page 5))

example:

```
_GetDaemon()->StartForm($Page, "multipart/form-data"); ;
```

## Forms multipages

### 4-2 CLASS 'OBJ\_FORM'

**USAGE:** The OBJ\_FORM object provides member functions for working with fields form

If error is encountered, all functions return NULL

#### SYNOPTIC

FUNCTIONS	USAGE
GetArrayFields()	return reference of all fields of <b>OBJ_FORM</b>
GetDebug()	return TRUE if mode is debugging
GetStatusForm()	return status form (cf SetStatusForm)
Is_Initialized()	return TRUE if form is already initialised
Is_Redisplay()	return TRUE if form is to redisplay
Is_valid()	return TRUE if form is valid
SetDebug(TRUE/FALSE)	Set mode debug for form page
SetSpacesAllowed(TRUE/FALSE)	Allows spaces in fields name
SetStatusForm()	Set status form Obj_form::FORM_VIRGIN_ => form virgin Obj_Form::FORM_confirmed => form valid on get obj_form::FORM_REDISHOW => form to redisplay obj_form::FORM_REDISHOW_IMAGE => form to redisplay for display image

# Forms multipages

## 4-2-1 FUNCTIONS

### 4-2-1-1 GETARRAYFIELDS

#### DESCRIPTION

This function return the reference of array all form fields.

```
$MyArray = $ObjForm->GetArrayFields();
```

### 4-2-1-2 GETDEBUG

#### DESCRIPTION

This function return TRUE if mode is debugging

```
$ObjForm->GetDebug();
```

### 4-2-1-3 GETSTATUSFORM

#### DESCRIPTION

This function return the form status.

```
$ObjForm->GetStatus();
```

#### REMARKS

These values may be:

- Obj\_Form::FORM\_VIRGIN => form virgin
- Obj\_Form::FORM\_CONFIRMED => form valid on get
- Obj\_Form::FORM\_REDISPAY => form to redisplay
- Obj\_Form::FORM\_REDISPAY\_IMAGE => form to redisplay for image

### 4-2-1-4 Is\_INITIALIZED

#### DESCRIPTION

This function check if form already initialised.

```
$ObjForm->Is_Initialized();
```

#### RETURN VALUE

This function return TRUE if form is already initialised. Otherwise return FALSE.

### 4-2-1-5 Is\_REDISPAY

#### DESCRIPTION

This function check if form is to redisplay.

```
$ObjForm->Is_Redisplay();
```

#### RETURN VALUE

This function return TRUE if form is to redisplay. Otherwise return FALSE.

### 4-2-1-6 Is\_VALID

#### DESCRIPTION

This function check if form is valid.

```
$ObjForm->Is_Valid();
```

#### RETURN VALUE

This function return TRUE if form is already initialised. Otherwise return FALSE.

## Forms multipages

### 4-2-1-7 SETDEBUG

#### DESCRIPTION

This function set or erase the mode debugging.

```
$ObjForm->SetDebug($bMode);
```

#### EXAMPLES

Set mode debugging: `$ObjForm->SetDebug(TRUE);`

Erase mode debugging: `$ObjForm->SetDebug(FALSE);`

### 4-2-1-8 SETSPACEALLOWED(TRUE/FALSE)

#### DESCRIPTION

This function allows use spaces for fields names.

**The use of spaces in the field names requires more resources execution time than the instruction submit replaces the characters space field names by the underscore.**

```
$ObjForm->SetSpaceAllowed(TRUE);
```

### 4-2-1-9 SETSTATUSFORM

#### DESCRIPTION

This function set the form status.

```
$ObjForm->SetStatusForm(Obj_Form::FORM_REDISPLA);
```

#### REMARKS

These values may be:

- Obj\_Form::FORM\_VIRGIN => form virgin
- Obj\_Form::FORM\_CONFIRMED => form valid on get
- Obj\_Form::FORM\_REDISHOW => form to redisplay
- Obj\_Form::FORM\_REDISHOW\_IMAGE => form to redisplay for image

## Forms multipages

### 4-2-2 FIELDS

#### 4-2-2-1 ATTRIBUTES FIELD

##### SYNOPTIC ATTRIBUTES

*on => available, off => not available | value between [...] are the default value | system => used by system*

ATTRIBUTES	BUTTON	CHECKBOX	RADIO	SELECT	TEXT
<b>nature</b>	[input] button	system <input>	system <input>	system <select>	[input] textarea
<b>type</b>	['button'] 'submit' 'reset' 'upload_image'*1 'upload_file'*1	off <checkbox>	off <radio>	off	['Text'] 'password' 'hidden' 'file'
<b>value</b>	label_button image	list pre_checked	list pre_checked	list pre_checked	init value
<b>func_user</b>	on*2	on*2	on*2	on*2	on*2
<b>func_user_err</b>	on*3	on*3	on*3	on*3	on*3
<b>size</b>	off	off	off	count select_display	maxi/mini
<b>format</b>	off	off	off	off	['s'] 'e' 'p...' *4 'n####.# #' 'aaaa-mm-jj' 'jpg jpeg gif'*5
<b>look</b>	on	on	on	on	on
<b>accesskey</b>	on	off	off	off	system*6
<b>list</b>	off	on	on	on	system*7
<b>multiple</b>	off	[TRUE]/FALSE	off	[TRUE]/FALSE	*10
<b>optional</b>	[FALSE]/TRUE	[FALSE]/TRUE	[FALSE]/TRUE	[FALSE]/TRUE	[FALSE]/TRUE
<b>disable</b>	[FALSE]/TRUE	[FALSE]/TRUE	[FALSE]/TRUE	[FALSE]/TRUE	[FALSE]/TRUE
<b>readonly</b>	[FALSE]/TRUE	[FALSE]/TRUE	[FALSE]/TRUE	[FALSE]/TRUE	[FALSE]/TRUE
<b>id</b>	on	on	on	on	on
<b>status</b>	on	on	on	on	on
<b>javascript</b>	on	on	on	on	on
<b>link</b>	on*8	system	system	system	system *9
<b>namemsg</b>	system	system	system	system	system
<b>valueinit</b>	system	system	system	system	system

\*1 => if type is 'upload\_image' image uploaded is displayed inside button. Attribute 'value' must contain <img src ='MyImage', ....> for default image with height and width and border for image to upload.  
=> if type is 'upload\_file' or 'upload-image':

Ctrl + Click display inside new window the file or the image uploaded

\*2 => callback user function called after button submit for personalized control (cf next page)

\*3 => callback user function called for display error, if no field link error specify (cf nextpage)

\*4 => format contains a pattern regular expression match with function php preg\_match(\$pattern).

if \$pattern is preceded by exclamation mark, preg\_match must return 0 for validate string.

\*5 => if type text is 'file', contains the formats for files allowed

\*6 => if type text is 'text', and format is 'n###, contains the rules for numeric value

\*7 => if type text is 'text' and format is 'n###, contains array for numeric value (cf 4-2-2-2-5 CONSTRUCT OBJECT OBJ\_TEXT)

\*8 => if type is 'upload\_image' or 'upload\_file', contains name of field text type 'file'

\*9 => if type text is 'file', contains field name of hidden field which contains the source full file name to upload.

=> if this a hidden field for a field text type 'file' to upload, contains the field name button type 'upload\_image'

\*10=>must be to true if this is a field Text of array values

## Forms multipages

- **For each field, it is possible to use function defined by user for personalized control (**P\_UserFuncControl**).**

The function type 'call\_user\_func\_array()' to be called. Class methods may also be invoked statically using this function by passing array(\$classname, \$methodname) to this parameter.

Parameters: field value, reference to object **OBJ\_FORM** that contains all fields of this script

- **For each button (valid, eras, etc...) it is possible to use function defined by user for personalized control (**P\_ScriptAfterValid**).**

The function type 'call\_user\_func\_array()' to be called. Class methods may also be invoked statically using this function by passing array(\$classname, \$methodname) to this parameter.

Parameters: field name of button, reference to object **OBJ\_FORM** that contains all fields of this script

- **For each field, it is possible to use function defined by user for display error message.**

The function type 'call\_user\_func' to be called. Class methods may also be invoked statically using this function by passing array(\$classname, \$methodname) to this parameter.

Parameters: string describing error, reference to object **OBJ\_FORM** that contains all fields of this script

- **For each field, it is possible to use function JavaScript.**

example:

```
$ArrayJavaScript =
    array ( "onKeyDown" => "MyFunction_1",
            "onBlur"    => "MyFunction_2",
            "onKeyUp"   => "MyFunction_3");
```

Parameters:

The parameters are set automatically by the monitor and are:

ObjForm	=> reference of form object
ObjItem	=> reference of form field
MaxLength	=> length allowed
MsgName	=> Messsage name of field message (if error encoutered)
Idx	=> index of value if this field is an array values

exemple: onBlur = MyFunction\_2( this.form, this,MaxLength, MsgName, Idx)

- **For each field, it is possible to put one or more attributes 'look'.**

example:

```
$MyFieldObject("look" => "class=\"Class Name\" size=\"##\" col=`##` \" ....
$MyFieldObject("look" => " size=\"##\" style =\"background:#999999;color:#FFFFFF;\" \" ....
$MyFieldObject("look" => array("class=\"Class1\"", "class=\"Class1\"", "class=\"Class1\""), size=\"##\""
cf 4-2-2-2-13 SetFieldStatus: (Obj_Form::DEFFORM_WARNING and (Obj_Form::DEFFORM_ERROR)
- the first element is for display a field or a message information
- the second element for display a field with status warning or a warning message
- the third element for display a field with status error or a error message
```

## Forms multipages

### 4-2-2-2 FUNCTIONS

#### USAGE

#### SYNOPTIC

FUNCTION	BUTTON	CHECKBOX	RADIO	SELECT	TEXT	COMMENT
Add_Button(...)	on	off	off	off	off	Create object Obj_Button
Add_CheckBox(...)	off	on	off	off	off	Create object Obj_CheckBox
Add_Radio(...)	off	off	on	off	off	Create object Obj_Radio
Add_Select(...)	off	off	off	on	off	Create object Obj_Select
Add_Text(...)	off	off	off	off	on	Create object Obj_Text
DeleteField(...)	on	on	on	on	on	Delete field
Is_Clicked(...)	on	off	off	off	off	Return TRUE if button clicked
DumpForm	on	on	on	on	on	Dump results form
GetElement(...)	off	off	off	off	on	allow access to a value for a field of array values
GetFieldList(...)	off	on	on	on	on	Return list choice or numeric value or date format iso
GetFieldStatus	on	on	on	on	on	Return status field
GetFieldValue(...)	off	on	on	on	on	Return field value
GetFieldValueMsg(...)	on	on	on	on	on	Return field link error value
GetMagic(...)	on	on	on	on	on	Return attribute field
HTMLText(...)	on	on	on	on	on	Build field code HTML
HTMLTextMsg(...)	on	on	on	on	on	Build field link error code HTML
SetFieldList(...)	off	on	on	on	off	Set new list choice
SetFieldStatus	on	on	on	on	on	Set status field
SetFieldValue(...)	on	on	on	on	on	Set new field value
SetFieldValueMsg(...)	off	on	on	on	on	Set new field link error value
SetMagic(...)	on	on	on	on	on	Set attribute field

## Forms multipages

### 4-2-2-2-1 CONSTRUCT OBJECT *OBJ\_BUTTON*

#### DESCRIPTION

Build object for field button

**GetDaemon()**->\$ObjForm(PAGE\_):

\$ObjForm->Add\_Button(\$NameFieldButton, \$Attributes, \$FieldLinkError = FALSE);

#### RETURN VALUE

This function return **TRUE** if no error, else return **NULL**

The error messages

- name already exist
- name not present
- value attribute 'nature' incorrect
- value attribute 'type' incorrect
- attribute not available

#### EXAMPLES

Build button field '**Continue**' without field link error and with callback user function "**myControl**".

\$ObjForm->Add\_Button("Continue",array("type"=>"submit", "func\_user" =>"myControl"));

**Note:** function **myControl** must be in script php and the name of this script must defined with parameter *P\_UserFuncControl*

Build button field '**erase**' without field link error.

\$ObjForm->Add\_Button("erase",array("type"=>"reset"));

Build button field '**Return**' with function javascript and with label 'Previous Page', without field link error.

\$ObjForm->Add\_Button("Return",array("value"=> "Previous Page",  
"javascript"=> array("onClick" => "GoPrevious")));

Build button field '**Graphic**' without field link error, for display a image in file '**FileImage**'

\$Imag = "<img src='FileImage' width='50' height='50' border='0' alt='Return'>";

\$ObjForm->Add\_Button("Graphic",array("value" => \$Imag));

Build button field '**Graphic**' for display a field text ('type' => 'file') named '**FileImageUpload**' (**file image upload**)

\$Imag = "<img src='DefaultImage.jpg' width='50' height='50' border='0'>";

\$ObjForm->Add\_Button("Graphic",array("type" => "upload\_image", "value" => \$Imag,  
"link"=>"FileImageUpload"));

**Note:** The field named '**FileImageUpload**' must be created before the field button,  
otherwise, generate error!

## Forms multipages

### 4-2-2-2-2 CONSTRUCT OBJECT *OBJ\_CHECKBOX*

#### DESCRIPTION

Build object for field checkbox

**GetDaemon()->\$ObjForm(PAGE\_):**

Build object for field checkbox

**\$ObjForm->Add\_Radio(\$NameFieldCheckbox, \$Attributes, \$FieldLinkError = FALSE);**

#### RETURN VALUE

This function return TRUE if no error, else return NULL

The error messages

- name already exist
- name not present
- value attribute 'multiple' incorrect for multiple choices
- attribute 'list' not present
- attribute 'list' is not array
- attribute 'list' contains element which is not Obj\_Choice
- attribute not available

#### NOTE

**If multiple selection is false, attribute 'value' is always a array  
(for use javascript, the name of field is followed with '[]')**

#### EXAMPLES

Build Checkbox Field named 'MyCheckBox', not optional, with field link error.

`$ObjForm->Add_CheckBox("MyCheckBox",array("list"=>$Choice), array());`

#### LIST CHOICE FOR OBJECT RADIO, CHECKBOX

For each field type Obj\_Radio or Obj\_CheckBox, a object Obj\_Choice must be created before Add\_ChecBox() or Add\_Radio().

Parameters:

1=> Value button radio or checkbox

2=> TRUE if pre-checked

3=> Label button radio or checkbox

        If not present, this label must be by user on the script

4=> Must be 'left' or 'right' for align label, if Label present

5=> Design view

        example:

`$Choice = array( new Obj_Choice("Mr",TRUE,"Mr","left", "class =\"MyClassFirstChoice\""),`

`new Obj_Choice("Mme",FALSE,"Mme","left"),`

`new Obj_Choice("Mlle",FALSE,"Mlle","left") );`

`$Choice = array( new Obj_Choice("M",TRUE),`

`new Obj_Choice("F") );`

## Forms multipages

### 4-2-2-2-3 CONSTRUCT OBJECT *OBJ\_RADIO*

#### DESCRIPTION

Build object for field radio

**GetDaemon()->\$ObjForm(PAGE\_):**

```
$ObjForm->Add_Radio($NameFieldRadioName, $Attributes, $FieldLinkError = FALSE);
```

#### RETURN VALUE

This function return TRUE if no error, else return NULL

The error messages

- name already exist
- name not present
- more options choice pre-checked
- attribute 'list' not present
- attribute 'list' is not array
- attribute 'list' contains element which is not Obj\_Choice
- attribute not available

#### EXAMPLES

Build Radio Field named '**MyRadio**', not optional, without field link error.

```
$ObjForm->Add_Radio("MyRadio",array("list"=>$Choice, "func_user_err"=>"MyFuncError"));
```

**Note:** function **MyFuncError** must be in script php and the name of this script must defined with parameter *P\_UserFuncControl*

cf CONSTRUCT OBJECT *OBJ\_CHECKBOX* for LIST CHOICE FOR OBJECT RADIO, CHECKBOX

## Forms multipages

### 4-2-2-2-4 CONSTRUCT OBJECT *OBJ\_SELECT*

#### DESCRIPTION

Build object for field select

**GetDaemon()->\$ObjForm(PAGE\_):**

**\$ObjForm->Add\_Select(\$NameFieldSelect, \$Attributes, \$FieldLinkError = FALSE);**

#### RETURN VALUE

This function return TRUE if no error, else return NULL

The error messages

- name already exist
- name not present
- value attribute 'multiple' incorrect for multiple select
- attribute 'list' not present
- value attribute 'size' incorrect for view select
- attribute 'list' is not array
- attribute 'list' contains element which is not Obj\_List
- attribute not available

#### NOTE

Attribute 'size' means number of option to view.

If attribute 'size' is -1 ("size" => -1): all options are viewed

If attribute 'size' is a number < all options, a window unrolling is used.

**If multiple selection is false, attribute 'value' is always a array  
(for use javascript, the name of field is followed with '[]')**

#### EXAMPLES

Build Select Field named '**MySelect**', not optional, with 2 options select, with field link error.

**\$ObjForm->Add\_Select("MySelect",array("list"=>\$List, "size"=> 2), array());**

#### LIST SELECT FOR OBJECT SELECT

**For each field type Obj\_Select, a object Obj\_List must be created before Add\_Select().**

Parameters:

1=> \$OptGroupLabel    label OptGroup for option  
                      must be FALSE if end option

2=> \$OptGroupLook    design view OptGroup  
3=> \$Label;           label option  
4=> \$PreSelected;    TRUE if pre-checked  
5=> \$Value;           value option, if "", contain \$Label  
2=> \$Look;           design view:

example:

```
$List = array( new Obj_List("", "", "Select_1"),
               new Obj_List("", "", "Select_2"),
               new Obj_List("", "", "Select_3", TRUE) );
```

```
$List = array( new Obj_List("First List", "", "Select_1"),
               new Obj_List("", "", "Select_2"),
               new Obj_List("Second List", "", "Select_3", TRUE),
               new Obj_List("", "", "Select_4") );
```

## Forms multipages

### 4-2-2-2-5 CONSTRUCT OBJECT *OBJ\_TEXT*

#### DESCRIPTION

Build object for field textt

**GetDaemon()->\$ObjForm(PAGE\_):**

**\$ObjForm->Add\_Text(\$NameFieldText, \$Attributes, \$FieldLinkError = FALSE);**

#### REMARKS

attribute 'multiple' must be to true if this field is a array (cf sudoku.php)

#### Value available for attribute 'format'

's' for string value

'p' followed by \$pattern for perform a regular expression match on string with function php 'preg\_match()'

if \$pattern is a empty, all character are allowed

if \$pattern is not empty, preg\_match() must return !0 for validate string

if \$pattern is not empty and is preceded by exclamation mark, preg\_match() must return 0 for validate string

Note:

Epression '/[^a]/' means that you are looking a character other than 'a', thus a string 'ba' will be validate.

To look the presence prohibited characters, it is most efficient use '!/[a]/' and test the return value 0 , for validate string

'e' for Email adress

'n' for numeric value

"n" => number

"n####" => integer with 4 digits

"n###.-" => number signed with 4 digits

"n##.#"- => number signed with 4 integers and 2 decimals

"n0.#"- => number signed decimal

"n.-" => number signed with no limit

'aaaa-mm-jj [hh:mm:ss]' for date ('-' or '/' or ':' or '' as separator)

'mm-aaaa-jj [hh:mm:ss]' for date ""

'jj-mm-aaaa [hh:mm:ss]' for date ""

'jpeg|gif|jpg' , 'pdf|docetc...' => list extension allowed for file uploaded

#### RETURN VALUE

This function return **TRUE** if no error, else return **NULL**.

The error messages

- name already exist
- name not present
- value attribute 'type' incorrect
- value attribute 'nature' incorrect
- value attribute 'size' incorrect
- value attribute 'format' incorrect
- attribute not available

## Forms multipages

### EXAMPLES

Build numeric Field named '**Code**', not optional, number digits between 5 and 2 with field link error.

```
$ObjForm->Add_Text("Code", array("format"=>"np! /[^0-9]/", "size"=>array(5,-2)), array());
```

**use return !preg\_match("/[^0-9]/", ..... for validate Code**

Build numeric Field named '**Code**', not optional, two digits between 0 et 2 with field link error.

```
$ObjForm->Add_Text("Code", array("format"=>"np^[0-2]{2,2}$"), array());
```

**use return preg\_match("^[0-2]{2,2}\$", ..... for validate code**

Build Text Field named '**Name**', not optional, number chars allowed between 40 and 2, without field link error.

```
$ObjForm->Add_Text("Name", array("size"=> array(40,2), "func_user_err"=>"MyFuncError"));
```

**Note:** function **MyFuncError** must be in script php and the name of this script must defined with parameter **P\_UserFuncControl**

Build Text Field named '**FirstName**', optional, number chars allowed between 20 and 2, with field link error initialised with "**max = 20c, min = 2c**".

```
$ObjForm->Add_Text("FirstName", array("size"=> array(20,2), "optional"=>true), array("value"=>"max = 20c, min = 2c"));
```

Build Text Field named '**Text**', not optional, number chars allowed between 1000 and 5, on 50 cols and 20 rows, with message and with field link error initialised with "**Please be concise**".

```
$ObjForm->Add_Text("Text", array("look" => "cols=\"50\" rows=\"10\" wrap=\"virtual\" style = \"background:#FF0000;color:#FFFFFF;\"", "nature" => "textarea", "size"=> array(1000,5)), array("value" => "Please be concise"));
```

Build numeric array Field named '**Number**', not optional, 4 digits and 2 decimals, between 9999 and -2000 with field link error.

```
$ObjForm->Add_Text("Number", array("multiple"=>true, "format"=>"n####,##-", "size"=>array(9999,-2000), array()));
```

Build Field named '**FileImageUpload**', for upload file image with field link error, with max size file to 500ko. If size attributeis not present, default size (php.ini) is the max size for upload)

```
$ObjForm->Add_Text("FileImageUpload", array("type" => "file", "format"=>"jpg|jpeg|gif", "size"=> 500000), array());
```

**cf (Obj\_Button for display File Image)**

Build Email Field named '**Email**', not optional, with field link error and. function user for control

```
$ObjForm->Add_Text("Email", array("format"=>"e", "func_user" =>"myControl"), array());
```

**Note:** function **myControl** must be in script php and the name of this script must define with parameter **P\_UserFuncControl**

### NOTE (CF SAMPLE PAGE\_3.PHP IN *Forms (more pages)*)

For field numeric or date, it is possible to get value numeric or date converted in format iso (YYYY-MM-DD hh:mm:ss) by call function GetMagic(\$NameField, "list") which return a array.

if type is numeric, array contains:

```
array[0] => numeric value  
array[1] => '+' or '-' (if present)  
array[2] => ',' or '.' (if present)  
array[3] => digits integers  
array[4] => digits decimal
```

contain a string if error encoutered

if type is date, array contains:

```
array[0] => "YYYY-MM-DD hh:mm:ss" date formatted iso for SQL  
array[1] => "YYYY" numbyear
```

## **Forms multipages**

array[2] => "MM"	number month
array[3] => "DD"	number day
array[4] => "hh"	hours
array[5] => "mm"	minutes
array [6] => "ss"	secondes

```
$Array = $ObjForm->GetFieldList("MyNameField");
if (!is_array($Array))
    error encountered
else
    $Result = $Array[0];
```

## Forms multipages

### 4-2-2-2-6 DELETEFIELD

#### DESCRIPTION

return TRUE if field \$Name is deleted  
**\$ObjForm->DeleteField(\$Name);**

#### RETURN VALUE

This function return TRUE or return FALSE,

#### EXAMPLES

Delete field named \$Name  
**\$bResult = \$ObjForm->DeleteField(\$Name);**

DeleteField(\$Ident)

### 4-2-2-2-7 IS\_CLICKED

#### DESCRIPTION

return TRUE if field button named \$Name is clicked  
**\$ObjForm->Is\_Clicked (\$Name);**

#### RETURN VALUE

This function return TRUE or return FALSE,

#### EXAMPLES

Test if button named \$Name is clicked  
**\$bClick = \$ObjForm->Is\_Clicked(\$Name);**

### 4-2-2-2-8 GETELEMENT

#### DESCRIPTION

This function allow for fields Obj\_Text to acces an one value for a field of array values  
**\$ObjForm->GetElement(\$Name, 3);**

#### EXAMPLES

The first element is 0, the second element is 1,..... (idem array php)  
Get value of second value for a field which accept several firstnames:  
**\$FirstName = \$ObjForm->GetValue(GetElement("firstname", 1;**

Display first firstname with HTMLText

**\$ObjForm->HTMLText(GetElement("firstname", 0; or \$ObjForm->HTMLText("firstname", 0**

For Display an error message for second firstname

**\$ObjForm->SetFieldStatus(GetElement("firstname", 1), 'sorry, firstname is not valid' );**

### 4-2-2-2-9 GETFIELDLIST

#### DESCRIPTION

This function return either the list choices (attribute 'list') for objects Obj\_Radio, Obj\_CheckBox and Obj\_Select or numeric value or date converted in format iso for named \$Name.

**\$ObjForm->GetList(\$Name);**

#### RETURN VALUE

This function return the list choices (attribute 'list') for objects Obj\_Radio, Obj\_CheckBox and Obj\_Select. named \$Name. If error encountered, return NULL

#### EXAMPLES

Get list choices of field named "MySelect":

**\$ListChoices = \$ObjForm->GetList("MySelect");**

### 4-2-2-2-10 GETFIELDSTATUS

#### DESCRIPTION

**\$ObjForm->GetFieldStatus(\$Name);**

#### RETURN VALUE

This function return the value of attribute 'status' for a field object named \$Name. If error encountered, return NULL. This value can be either TRUE either a string that contains an error message

#### EXAMPLES

Get Status of field named \$Name

**\$Status = \$ObjForm->GetFieldStatus(\$Name);**

### 4-2-2-2-11 GETFIELDVALUE

#### DESCRIPTION

**\$ObjForm->GetFieldValue(\$Name);**

#### RETURN VALUE

This function return the value of attribute 'value' for a field object named \$Name. If error encountered, return NULL

#### EXAMPLES

Get Value of field named \$Name

**\$Value = \$ObjForm->GetFieldValue(\$Name);**

## Forms multipages

### 4-2-2-2-12 GETFIELDVALUEMSG

#### DESCRIPTION

```
$ObjForm->GetFieldValueMsg($Name);
```

#### RETURN VALUE

This function return the value of attribute 'value' for a field link error object named **\$Name**. If error encountered, return **NULL**.

#### EXAMPLES

Get Value of field named **\$Name**:

```
$ValueMsg = $ObjForm->GetFieldValueMsg($Name);
```

### 4-2-2-2-13 GETMAGIC

#### DESCRIPTION

This function return the value of attribute **\$Attribute** for a field named **\$Name**.

```
$ObjForm->GetMagic($Name, $Attribut);
```

#### RETURN VALUE

This function return the value of attribute **\$Attribute** for a field named **\$Name**.

If error encountered, return **NULL**.

#### EXAMPLES

Get value of attribute 'size' for object named "MyName"

```
$Size = $ObjForm->GetMagic("MyName","size");
```

## Forms multipages

### 4-2-2-2-14 HTMLTEXT

#### DESCRIPTION

This function build code HTML for a field named \$Name.

**\$ObjForm->HTMLText(\$Name, \$Index = 0);**

If this a CHECKBOX or RADIO, \$Index is this index of case to display (0 à n)

If this a TEXT array of fields values, \$Index is this index of value to display (0 à n)

#### RETURN VALUE

This function return a string code HTML.

If error encountered, return **NULL**

#### EXAMPLES

Get code HTML for a field named "MyName";

**\$String = \$ObjForm->HTMLText("MyName");**

Get code HTML for third box of field CHEKBOX named "MyName";

**\$String = \$ObjForm->HTMLText("MyName", 2);**

Get code HTML for second value of field TEXT named "MyName";

**\$String = \$ObjForm->HTMLText("MyName", 1);**

### 4-2-2-2-15 HTMLTEXTMSG

#### DESCRIPTION

This function build code HTML for a field link error named \$NameMsg.

**\$ObjForm->HTMLTextMsg(\$NameMsg);**

#### RETURN VALUE

This function return a string code HTML.

If error encountered, return **NULL**

#### EXAMPLES

Get code HTML for a field link error named "MyNameMsg";

**\$String = \$ObjForm->HTMLTextMsg("MyNameMsg");**

### 4-2-2-2-16 SETLIST

#### DESCRIPTION

Set the list choices (attribute 'list') for object Obj\_Radio, Obj\_CheckBox and Obj\_Select. named \$Name.

**\$ObjForm->SetList(\$Name, \$NewListChoices);**

#### RETURN VALUE

This function return **TRUE** or return **NULL** if error,

#### EXAMPLES

Set the list choices with \$NewListChoices for object named "MySelect"

**\$bOk = \$ObjForm->SetList("MySelect", \$NewListChoice);**

## Forms multipages

### 4-2-2-2-17 SETFIELDSTATUS

#### DESCRIPTION

Set the status of attribute 'status' for a field object named **\$Name** with **\$NewStatus**.  
**\$NewValueStatus** can be either **TRUE** either a string which contains a error message  
**\$ObjForm->SetFieldStatus(\$Name, \$NewStatus = TRUE, \$Type = Obj\_Form::DEFFORM\_ERROR);**

#### RETURN VALUE

This function return **TRUE** or return **NULL** if field object named **\$Name** not exists

#### EXAMPLES

Set Status of field named "**MyName**" with "error on your answer" and warning  
**\$bRes = \$ObjForm->SetFieldStatus("MyName", "error on your answer", Obj\_Form::DEFFORM\_WARNING);**

Set Status of field named "**MyName**" with "error Fatal on your answer"  
**\$bRes = \$ObjForm->SetFieldStatus("MyName", "error on your answer");**

Set Status of field named "**MyName**" with **TRUE**  
**\$bRes = \$ObjForm->SetFieldStatus("MyName");**

## Forms multipages

### 4-2-2-2-18 SETFIELDVALUE

#### DESCRIPTION

Set the status of attribute 'value' for a field object named \$Name with \$NewValue.  
**\$ObjForm->SetFieldValue(\$Name, \$NewValue);**

#### RETURN VALUE

This function return TRUE or return NULL if field object \$Name not exists

#### EXAMPLES

Set Value of field named "MyName" with 10-30-2007

**\$bRes = \$ObjForm->SetFieldValue("MyName", "10-30-2007);**

### 4-2-2-2-19 SETFIELDVALUENAME

#### DESCRIPTION

Set the status of attribute 'value' for a field object link error object named \$NameMsg with \$NewValueMsg.  
**\$ObjForm->SetFieldValueMsg(\$NameMsg, \$NewValueMsg);**

#### RETURN VALUE

This function return TRUE or return NULL if field object \$NameMsg not exists

#### EXAMPLES

Set Value of field object link error named "MyNameMsg" with "please your firstname"

**\$bRes = \$ObjForm->SetFieldValue("MyNameMsg", "please your firstname");**

## Forms multipages

### 4-2-2-2-20 SETMAGIC

#### DESCRIPTION

Set the value of attribute **\$Attribute** for a field named **\$Name**.

If attribute **\$Attribute** not exists and **\$bForce** is to TRUE, the attribute **\$Attribute** is created.

```
$ObjForm->SetMagic($Name, $Attributes, $NewValue, $bForce = FALSE);
```

#### RETURN VALUE

This function return TRUE or return NULL if error.

#### NOTE:

The use of this function makes it possible to modify any attribute of a field. Used ill-advisedly, this function can generates errors.

#### EXAMPLES

Set value of attribute 'size' for object named "MyName", with max length to 20 and min length to 2:

if attribute 'size' no exists, it is created

```
$bOk = $ObjForm->SetMagic("MyName", "size", array(20,2), TRUE);
```

## Forms multipages

### 5 – FOLDERS (MUST BE VERIFIED)

```
Directory 'www'  
    " CountSession.txt  
    " Demo.php  
    " index_laur.php (a renommer en 'index.php')  
    " livres.php  
    " windowlivre.php  
www/def/[folder]  
    " DefChampForm.php  
    " DefDaemonForm.php  
    " DefDownLoad.php  
    " DefError.php  
    " DefForm.php  
    " DefMySQL.php  
    " DefMySQLi.php  
    " DefPDO.php  
    " DefPgSQL.php  
    " DefSendEmail.php  
    " DefSQL.php  
    " DefTrace.php  
    " DefUtil.php  
www/SQL/[folder]  
    " BuildTableSQL-codes_postaux_france.php  
    " BuildTableSQL-postgres-codes_postaux_france.php  
    " BuildTableSQL-postgres-regions_world.php  
    " BuildTableSQL-regions_world.php  
    " CodesPostauxFrance.csv  
    " DemoStoredProcedure.php  
    " Import_pgadmin_CodesPostauxFrance.csv  
    " Import_pgadmin_RegionsWorld.csv  
    " RegionsWorld.csv  
www/DownLoad/[folder]  
    " CounterDownLoad.php  
    " DownLoad.php  
    " DownLoad.txt  
www/form/[folder]  
    " Address_Postal.php  
    " CallUserFunc_Address_Postal.php  
    " CallUserFunc_AdrEmail.php  
    " CallUserFunc_Crypt.php  
    " CallUserFunc_DemoSQL.php  
    " CallUserFunc_Forms.php  
    " CallUserFunc_sudoku.php  
    " Contact.php  
    " Crypt.php  
    " GetAdrEmail.php  
    " Page_1.php  
    " Page_2.php  
    " Page_3.php  
    " Page_4.php  
    " Page_DemoSQL.php  
    " process_sudoku.php  
    " Result_Address_Postal.php  
    " Sample_More_Forms.php  
    " sudoku.php  
    " UserContact.php  
    " UserSendEmail.php  
    " UserValidAddress_Postal.php  
    " UserValidForms.php  
www/geography/[folder]  
    " geography.php  
www/images/[folder]  
    " enveloppe1.gif  
    " livre_1.gif  
    " lunette.jpg  
    " nous_r1.gif  
www/upload/[folder]
```