GneatGeek

FormB

Introduction

FormB is a class for common online form processing written in PHP. It offers support for user error handling, email address validation, web form validation, different ways to send data via email, html email template support, database user IP address logging, methods to sanitize user input to protect against XSS attacks, and more!

Class synopsis

```
FormB {
```

Properties

```
private array $_errors;
private array $_required;
private resource $_handle;
```

Methods

```
public void __construct ( [ array $requiredVars = "", resource $mysqlHandle = "" ] )
final public static bool isSetNotEmpty ( mixed &$var )
final public void addError ( string $errorMessage )
final public bool checkErrors ( void )
public void processReq ( void )
private void setErr ( string $e )
public void printErrors ( void )
public static string printTxt ( string $requestVar [ , bool $return = FALSE ] )
public static string printVal ( string $requestVar [ , bool $return = FALSE ] )
public static void printOptions ( string $selectName, array &$options [ , string $defaultText
   = " ", bool $useKey = FALSE] )
public void sendMail ( string/array $emailRecipient, string $subject, file $template, string
 $sender )
public void lazyMail ( string/array $emailRecipient, string $subject, string $sender )
final public void validEmail ( string &$email, [ bool $chkDNS = TRUE ] )
static private string sanitizeInput ( string $string )
final static public array sanitizeArray ( array $arr )
final static public mixed sanitizeObject ( mixed $ob )
final static public void sanitizePOST ( void )
final static public void sanitizeGET ( void )
public string checkItBox ( string $string, array &$checkBoxArray )
public void checkIP ( string $tbl, string $urlOmitString [ , int $allowableNumAttempts = 3] )
private void logIP ( string $tbl, string $urlOmitString )
public void handleIP ( void )
private static void verConstant ( string $e )
private static string sanitizeUrlString ( string $string, string $needle )
private void connect2DB ( void )
private void closeDB ( void )
final public var getVar ( string $var )
}
```

Properties

_errors

The errors array

_required

The list of form (POST/GET) fields that are not allowed to be empty

_handle

The mysql resource handle to be used for the IP logging subsystem.

Table of Contents

Method Descriptions

__construct

Description

public void *__construct* ([array $requiredVars = "", resource *$mysqlHandle* = ""])

Constructor for the FormB class (Form Backend).

Parameters

requiredVars

String or array of the name(s) of form fields to be validated. (GET, POST, & COOKIE are all checked via $_REQUEST)

mysqlHandle

MySQL resource to be used with the *checkIP()* method.

Return Values

No value is returned

isSetNotEmpty

Description

final public static bool *isSetNotEmpty* (mixed *&$var*)

Determines wether or not a specified variable is set AND not empty.

Parameters

var

Name of the variable to check.

Return Values

Returns **TRUE** if the variable is set AND not empty, **FALSE** otherwise.

addError

Description

final public void *addError* (string *$errorMessage*)

Adds a custom error message to the $_errors array list. This method is used to add errors to the error handling system that are handled outside of the class to maintain consistancy.

Parameters

errorMessage

Complete error message to be added to the system.

Return Values

No value is returned.

checkErrors

Description

final public bool *checkErrors* (void)

Determines wether or not there are errors set.

Parameters

This method has no parameters.

Return Values

Returns **TRUE** if there are errors set, **FALSE** otherwise.

processReq

Description

public void *processReq* (void)

Runs through the *$_required* array and sets errors if any *$_REQUEST*[*$_required*] (GET, POST, and COOKIE combined) variables are empty or NULL.
Uses the *setErr()* method to actually validate and set errors.

Parameters

This method has no parameters.

Return Values
No value is returned.

setErr

Description
private void *setErr* (string *$e*)

Helper method for the *processReq()* method. Checks for empty or null strings in *$_REQUEST*[*$_required*] and sets errors for all empty/null strings found.

Parameters
e
Name of $_REQUEST variable to be checked.

Return Values
No value is returned.

printErrors

Description
public void *printErrors* (void)

Runs through the *$_errors* array and outputs them in valid HTML inside a division with the class of errors ( ). Call this method **where** the errors should be **displayed** on the page.

Parameters
This method has no parameters.

Return Values
No value is returned.

printTxt

Description
public static string *printTxt* (string *$requestVar* [, bool *$return* = FALSE])

Parses the *$_REQUEST[$e]* into 'value="*$_REQUEST[$e]*" and either prints it to the screen or returns it. This method is used to make forms more user friendly by making it easy to pre-fill already submitted data if a submission error were to occur.

***USES sanitizeInput() FOR PROTECTION AGAINST XSS ATTACKS!

** Do not mix and match GET/POST/COOKIE with the same field names or one will get overwritten by the other as defined in php.ini. Normally GPC GET, POST, and COOKIE.

Parameters
requestVar
Name of the *$_REQUREST* variable to be parsed.
return
Boolean value to determine wether to return the parsed string or to print it out to the screen directly.

Return Values
Returns a formatted value = "*" string if $return == **TRUE**

printVal

Description
public static string *printVal* (string *$requestVar* [, bool *$return* = FALSE])

Wraps the returned string from printTxt(); in a value tag.

Parameters
requestVar
Name of the *$_REQUREST* variable to be parsed.
return
Boolean value to determine wether to return the parsed string or to print it out to the screen directly.

Return Values

Returns a formatted value = “*” string if $return == **TRUE**

printOptions

Description

public static void *printOptions* (string *$selectName*, array *&$options* [, string *$defaultText* = " ", bool *$useKey* = FALSE])

Creates a complete drop down menu with the ability for it to auto-select the last known selection in case the form has an error during submission.

Parameters

selectName

Name of the *$_REQUREST* variable to be parsed.

options

Array of options to print out for the user to select.

defaultText

Non-Selectable default option text

useKey

Wether or not to use the array key as the option value.
array("Value" => "Display to user") etc

Return Values

No value is returned. But this method does print option tags to the screen.

sendMail

Description

public void *sendMail* (string/array *$emailRecipient*, string *$subject*, file *$template*, string *$sender*)

Sends an HTML email with a pre-defined format.

Parameters

emailRecipient

String or array of the email address(es) for the email to be sent to. Each string in the array is handled separately meaning multiple emails are sent as opposed to one email with multiple recipients. The formatting of this string must comply with RFC 2822. See examples below.

* user@example.com
* user@example.com,anotheruser@example.com
* User <user@example.com>
* User <user@example.com>, Another User <anotheruser@example.com>

subject

Subject of the email to be sent. It must comply with RFC 2047.

template

HTML and/or PHP template file as the message to be sent. This method allows a highly customizable email to be sent. It allows for complete control of the design of the email. See the included examples for help.

Sender

The sender of the message. See emailRecipient for guidelines.

Return Values

No value is returned.

lazyMail

Description

public void *lazyMail* (string *$e*, bool *$return*=FALSE)

Sends a simple array dump of the $_POST array via email. This is handy if you don't really care about formatting and don't want to create an email template (like in sendMail).

The unset command can be used on POST variables you don't want included.
unset($_POST[a], $_POST[b], …);

Parameters

emailRecipient

String or array of the email address(es) for the email to be sent to. Each string in the array is handled separately meaning multiple emails are sent as opposed to one email with multiple recipients.

The formatting of this string must comply with RFC 2822. See examples below.
* user@example.com
* user@example.com,anotheruser@example.com
* User <user@example.com>
* User <user@example.com>, Another User <anotheruser@example.com>

subject

Subject of the email to be sent. It must comply with RFC 2047.

Sender

The sender of the message. See emailRecipient for guidelines.

Return Values

No Value is returned.

validEmail

Description

final public void *validEmail* (string reference *&$email*, bool *$chkDNS*=TRUE)

Validates the supplied email address for proper format by standard regular expressions and checks the MX records to verify the domain of the email address if $chkDNS is set to TRUE.

Sets an error into the internal error handling system if the email address is not valid.

Parameters

email

The email address to be verified.

chkDNS

Boolean value to determine wether to check the MX records or not. Defaults to TRUE and it is recommended to use it for additional verification.

Return Values

No value is returned.

sanitizeInput

Description

static private string *sanitizeInput* (string *$string*)

Sanitizes user input by converting all applicable characters to HTML entities.
This is a simple yet effective way of removing XSS attack potential from web forms.
This method is NOT sufficient for CMS use where HTML will be reposted/executed!

Parameters

string

String to be sanitized.

Return Values

Sanitized string $string.

sanitizeArray

Description

final static public array *sanitizeArray* (array *$arr*)

Sanitizes user input via sanitizeInput();
This method handles arrays to be sanitized and recursively calls itself if there are sub-arrays found.

Parameters

arr

Array to be sanitized.

Return Values

Sanitized array $arr.

sanitizeObject

Description

final static public mixed *sanitizeObject* (mixed *$ob*)

This is the general sanitization method!
Sanitizes user input via sanitizeInput();
Determines whether the object passed is an array or not and handles the data accordingly.

Parameters

ob

Object to be sanitized.

Return Values

Sanitized object $ob.

sanitizePOST

Description

final static public void *sanitizePOST* (void)

Sanitizes the entire POST array via sanitizeArray();.

Parameters

This method has no parameters

Return Values

No Value is returned.

sanitizeGET

Description

final static public void *sanitizeGET* (void)

Sanitizes the entire GET array via sanitizeArray();.

Parameters

This method has no parameters

Return Values

No Value is returned.

checkItBox

Description

public string *checkItBox* (string *$string*, array *&$checkBoxArray*)

Checks an array of checkboxes to determine if the checkbox in question is checked or not.

Parameters

string

Value of the checkbox in question.

checkBoxArray

Array reference to the array of checkboxes to check against.

Return Values

Empty string if the checkbox is not checked and string 'checked=”checked”' if it is.

checkIP

Description

public void *checkIP* (string *$tbl*, string *$urlOmitString* [, int *$allowableNumAttempts* = 3])

Checks if the user has exceeded the allowed number of attempts in the last 30 minutes to correctly fill out your form.

Calls logIP() to log the user's IP address after checking their IP address.

** REQUIRES MYSQL **

SEE Proposed SQL Structure in default.sql!

Parameters

tbl

The database table to be used.

urlOmitString

Constant in the url ("/~username/" for example) to be omitted. This is used to determine which form the user is submitting to and exceeding.

allowableNumAttempts

The number of allowable attempts to submit the form before temporarily disabling a user to submit.

Return Values

No value is returned.

logIP

Description

private void *logIP* (string *$tbl*, string *$urlOmitString*="")

Logs a user's IP address. If the user submits or attempts to submit, log their IP address along with a timestamp and what form page they were on.

Called from checkIP()

** REQUIRES MYSQL **
SEE Proposed SQL Structure in default.sql!

Parameters

tbl

The database table to be used.

urlOmitString

Constant in the url ("/~username/" for example) to be omitted. This is used to determine which form the user is submitting to and exceeding.

Return Values

No value is returned.

handleIP

Description

public void *handleIP* (void)

Uses constants defined in config.ini to handle the IP checking automatically.

Opens a MYSQL connection, calls checkIP(), and closes the MYSQL connection. Don't use this if you already have an open/active connection.

DO NOT leave config.inc in the public_html folder if you set the config file constants for this method since that could expose private data!!!

** REQUIRES MYSQL **
SEE Proposed SQL Structure in default.sql!

Parameters

This method has no parameters.

Return Values

No value is returned.

verConstant

Description

private static void *verConstant* (string *$e*)

Helper method for verifying the constants are all set for use in handleIP. Throws an error if any of the necessary constants are not defined. The handleIP method calls this automatically.

Parameters

e

Name of the constant to be verified.

Return Values
No value is returned.

sanitizeUrlString

Description
private static string *sanitizeUrlString* (string *$string*, string *$needle*)

Helper method for stripping out a constant part of the URL. "/~username/" for example.
Parameters
string
The full string that needs the constant removed.
needle
The constant to be removed.
Return Values
A string with the constant removed.

connect2DB

Description
private void *connect2DB* (void)

Helper method for connecting to the database. Aids the handleIP() method.
Parameters
This method has no parameters.
Return Values
No value is returned.

closeDB

Description
private void *connect2DB* (void)

Helper method for closing the database. Aids the handleIP() method.
Parameters
This method has no parameters.
Return Values
No value is returned.

getVar

Description
final public mixed *getVar* (string *$var*)

Allows you to access internal/private variables for testing purposes.
Parameters
var
Name of the internal variable to be returned
Return Values
The value of the requested variable.

Line

Introduction

Line is the class for simple HTML email formatting used in conjunction with the FormB class. This class is **NOT** necessary to format the emails, but can greatly simplify the effort. **Please Note: This class will be removed from future versions!**

Example**

```
print(new line("Name: ",$_POST['first_name'] . " " . $_POST["last_name"],1));
```

Class synopsis

```
Line {

    Properties

        private string $_pre;
        private mixed $_data;
        private sting $_line;
        private bool $_s;

    Methods

        public __construct ( string $prefix, mixed $data, bool $state )
        private void verify ( void )
        private void sWrap ( void )
        public string __toString ( void )
}
```

Properties

_pre

The prefix to the data (IE. Phone: [for a phone number])

_data

The submission data that goes with the given prefix (IE. 555-555-1212 [for a phone number]

_line

Finalized version of the string to be printed out.

_s

Boolean used to determine whether to use a
 tag or a </p> tag at the end of the printed line.

Method Descriptions

__construct

Description

public void *__construct* (string *$prefix*, mixed *$data*, bool *$state*)

Constructor for the Line class.

Parameters

prefix

The prefix to the data (IE. Phone: [for a phone number])

data

The submission data that goes with the given prefix (IE. 555-555-1212 [for a phone number].

state

Boolean used to determine whether to use a
 tag or a </p> tag at the end of the printed line.

Return Values

No value is returned

verify

Description

private void *verify* (void)

Helper Method. Verifies the data being sent through has at least some data in it.

Parameters

This method has no parameters.

Return Values

No value is returned.

sWrap

Description

private void *swrap* (void)

Helper Method. Wraps the content to be printed into nice clean formatting.

Parameters

This method has no parameters.

Return Values

No value is returned.

__toString

Description

public string *__toString* (void)

Helper Method. Wraps the content to be printed into nice clean formatting.

Parameters

This method has no parameters.

Return Values

Returns the string to be printed.