Generated Documentation

Contents

Package default Classes

Class xml_line

[line 30]

XML-LINE Version 0.2.11 == 09.04.2004

PHP-Klasse zum Abfragen und Ändern von XML-Dateien (C) 2004 Peter Bieling, pb@media-palette.de Einige Codezeilen, für den Aufruf der Parserfunktionen sind dem PHP-Manual entnommen.

- **Package** default
- **TODO** (!) Datei-Umbenennungsfunktionen, falls Zieldatei die Quelldatei ersetzen soll. -(!!!) Quell-XML wird nur aus einer Datei nicht aus String angenommen. -(!!!) Kopieren von ganzen XML-Abschnitten oder Elementen ist noch nicht möglich
- **Author** Peter Bieling < pb@media-palette.de>
- **Version** xml-line.php v 0.2.8

xml_line::$action_func

array = array() *[line 154]*

Name der aufzurufenden Funktion. Funktionen werden in einem Array gespeichert. $action_func[0] = Funktionsname $action_func[1] = Parameterliste (array) $action_func[2] = Kommando $action_func[3] = Ersetzungsstring

xml_line::$akt_name

string = "" *[line 57]*

Name des aktuellen Elements

xml_line::$breakset

boolean = false *[line 284]*

Internes Flag für die Highlight-Ausgabe. Merkt sich, ob vorher ein Zeilenumbruch gesetzt wurde.

xml_line::$csv_buffer

string = "" *[line 86]*

Speichert die CSV-Ausgabe, sofern sie (statt XML)gefordert ist, bzw. nicht in eine Datei geschrieben wird. Experimentell!!!

xml_line::$datafound

boolean = false *[line 277]*

Interner Schalter für die Highlight-Ausgabe. Merkt sich, ob Daten ausgegeben wurden

xml_line::$data_buffer

string = "" *[line 135]*

Expat liefert die Daten häufig in mehreren Stücken Hier wird das Ergebnis zwischengespeichert, bis der Data-String komplett ist

xml_line::$elc

array = array(array()) *[line 217]*

element-level-counter, enthält die Zählerwerte, der aktuellen Elemente.

[level]["elementname"]

xml_line::$elc_path_array

array = array() *[line 224]*

Kurzform des aktuellen Elementpfads (3,2,8) ohne Bezug auf die Elemente

xml_line::$el_counter

mixed = "" *[line 210]*

Element-Counter. Läuft nur mit, wenn er gesetzt wird. Kann für statistische Zwecke benutzt werden $el_counter[elname] = array($countall, $level => $countonthislevel

- **Var** (string or array)

xml_line::$error

array = array() *[line 232]*

Error - noch sträflich vernachlässigt ;-(Soll die Fehlermeldungen aufnehmen, die man dann abrufen kann. (noch in der Entwicklung)

xml_line::$file

string = *[line 43]*

Der Name der XML-Input-Datei

xml_line::$fill

array = array() *[line 94]*

Speichert Position als key und einen String als value für das Ersetzen oder Einsetzen von Text oder XML .Jede Funktion nutzt diesen Speicher selbst bei Bedarf.

- **Var** join("", $this->ls)=> Insertstring

xml_line::$fillattr

array = array() *[line 102]*

Speichert einzusetzende Attribute in einem Hash (name=>value), wenn ein oder mehrere Attribute hinzugefügt werden sollen

xml_line::$found_data

array = array() *[line 194]*

Speichert die gefunden Daten, Pfadinformationen und Attribute in einem Array Art der Rückgabe kann sich noch ändern, am besten mit print_r() selbst ansehen. ;-)

xml_line::$found_data_part

arry = array() *[line 202]*

Zähler für Textstücke innerhalb eines Elements mit weiteren untergeordneten Elementen, läuft nur mit, wenn er benötigt wird und das passende Element gefunden wurde.

- **Var** (key aus Prozessnr-pathstring-und-levelstring)

xml_line::$func_counter

integer = 0 *[line 162]*

Zählt die gesetzten Funktionen. Ermöglicht das Auseinanderhalten der verschiedenen Prozesse und die Zuordnung der Rückgabewerte.

xml_line::$highlight

boolean = false *[line 270]*

Interner Schalter für die Highlight-Ausgabe

xml_line::$ignore_default_data

boolean = false *[line 263]*

XML-Header und Kommentare ausgeben oder nicht. Wird über Methodenaufruf verändert

xml_line::$last_nowrite_line

string = "" *[line 71]*

Markiert das Ende von Bereichen, die nicht geschrieben (also de facto gelöscht) werden.

- **Var** join("",$this->ls);

xml_line::$level

integer = 0 *[line 127]*

Gibt den aktuellen Level der Elementhierarchie an

xml_line::$ls

array = array('type' => "", // S (START) D = (DATA)
// E (END) EOS (End of Start) A = (Attribute) X=(Doctype und Kommentare)
'level' => 0, // level of XML-Tree
'pathstring' => "", // element1/element2
'attrn' => "", // attributename if type='A'
'data' => "" // data of element or attribute
) *[line 180]*

Enthält die wichtigsten Informationen der aktuellen XML-Position.

xml_line::$of

integer = *[line 246]*

Outputfilehandler

- **Var** resource_id

xml_line::$outfile

string = "" *[line 50]*

Der Name der Output-Datei

xml_line::$output_action

mixed = "" [line 120]

Gibt an, ob bzw. was ausgegeben werden soll.
xml, csv, hixml oder Leerstring

xml_line::$parser

integer = [line 239]

Expat-Variable

- **Var** resource

xml_line::$path_array

array = array() *[line 143]*

Enthält alle Elemente des aktuellen Elementpfads einzeln als Array-Werte.
Damit kann dieser problemlos ergänzt oder gekürzt werden

- **Var** (root,level1,level2)

xml_line::$procdata

array = array(array()) *[line 173]*

Speicher für zusätzliche Prozessinformationen, 'datapart' z.B. ([1]['datapart']) merkt sich bei gemischten Inhalten, welcher Datenabschnitt ausgegeben werden soll 'limit' [1]['limit']=3 bedeutet: 3 Treffer werden gesucht, danach steigt die Funktion aus.
(runterzählen bis auf 0)

xml_line::$table_result

mixed = array() *[line 256]*

Soll bei Datensatzabfrage, das Ergebnis speichern: $table_result[Funktionsnr][counter]['key'] => "value" key ist die "Spaltenüberschrift" = Name des Elements.

Ist das Element untergeordnet ist key 'parent/child' Attribute werden ebenfalls in die Tabelle aufgenommen: 'element@attribut1'

xml_line::$temp_attr_buffer

array = array() *[line 112]*

Speichert die Attributhashs solange das Element aktuell ist. (Key ist der "pathname") Achtung: Wird wegen neuer Funktion get_record zur Zeit nicht gelöscht. (todo) Wichtig für Einfügeoperationen, damit auch beim Endelement noch bekannt ist, welche Attribute das Element hat.

xml_line::$testbuffer

string = *[line 36]*

Variable für Testausgaben.

xml_line::$write_flag

boolean = true *[line 64]*

Ausgabe wird nur geschrieben, wenn true gesetzt ist.

xml_line::$xml_buffer

string = "" *[line 78]*

Speichert die XML-Ausgabe, sofern sie gefordert ist, bzw. nicht in eine Datei geschrieben wird

Constructor function xml_line::xml_line($file, [$out = ""], [$outfile = ""]) *[line 294]*

Function Parameters:

- *string* **$file** XML-Quelldatei
- *string* **$out** optional Ausgabeformat (z.B. xml oder hixml)

- *string* **$outfile** optional Ausgabedatei

Constructor.

- **Access** public

function xml_line::add_attributes($arg, $insert) *[line 882]*

Function Parameters:

- *array* **$arg**
- *string* **$insert**

Fügt Attribute in einem Hash hinzu. Nutzt ebenfalls die Startelementmarke und nicht die Attributmarke, weil direkt auf das Expat-Attributarray zugegriffen wird, bevor dieses weiter verarbeitet wird.

(Das muss man sich klar machen, wenn zunächst Attribute zugefügt und gleichzeitig nach gleichnamigen Attributen gesucht wird!)

- **See** xml_line::get_data()
- **Access** public

function xml_line::append_data($arg, $value) *[line 786]*

Function Parameters:

- *array* **$arg**
- *string* **$value**

Hängt den neuen Inhalt an den Wert an Zusätzlich zur Parameterliste $arg wird ein Wert übergeben, der den gefundenen String erweitert.

- **Access** public
- **See** xml_line::get_data()

function xml_line::change_attribute_value($arg, $value) *[line 653]*

Function Parameters:

- *array* **$arg**
- *mixed* **$value** (string or integer)

Setzt eine Änderungsabfrage. Funktioniert ähnlich wie get_attribute. Die dort übergebenen Parameter werden hier als Array übergeben. Als zweiter Parameter kommt der String oder die Zahl, die den alten Wert ersetzt. Die Funktion wird sooft angewendet, wie sie auf die Abfrage trifft. Der entfernte Wert wird nicht gespeichert.

- **See** xml_line::get_attribute()
- **Access** public

function xml_line::change_data($arg, $value) *[line 772]*

Function Parameters:

- *array* **$arg**
- *string* **$value**

Ersetzt den gefundenen String durch einen neuen.

Zusätzlich zur Parameterliste $arg wird ein Wert übergeben, der den gefundenen String ersetzt.

- **Access** public
- **See** xml_line::get_data()

function xml_line::change_data_by_id($value, $replace, [$idname = "id"]) *[line 939]*

Function Parameters:

- *string* **$value**
- *string* **$replace**
- *string* **$idname** optional (anderer Atrributname, wenn nicht "id")

Ersetzt den Elementwert eines Elements mit einem bestimmten id-Attribut, z.B. change_data_by_id(15,"Nägel"); <Kiste id="1">Schrauben</Kiste>

- **See** xml_line::find_data_by_id()
- **Access** public

function xml_line::defaultData($parser, $ddata) *[line 432]*

Function Parameters:

- *integer* **$parser** resource_id
- *string* **$ddata**

Sorgt für die Ausgabe von Kommentaren und Doctype.

function xml_line::delete_attribute($arg) *[line 665]*

Function Parameters:

- *array* **$arg**

Löscht das angegebene Attribut.

- **Access** public
- **See** xml_line::get_attribute()

function xml_line::delete_content([$limit = 0], [$path = ""], [$pattern = ""], [$attrn = ""], [$elcount = ""]) *[line 832]*

Function Parameters:

- *integer* **$limit** optional Anzahl der gesuchten Treffer. Danach wird die Funktion beendet
- *string* **$path** optional root/element1/element2 (kann unvollständig sein, z.B. element2)
- *string* **$pattern** optional String, der in den gesuchten Daten enthalten sein muss
- *mixed* **$attrn** optional array(name=>value, name2=>value2) Attribut(e) müssen enthalten sein.
- *string* **$elcount** optional "1-6-2" oder "1-4-6:2" oder "1-6-{3 to 5}" oder "1-6-{3,5,8}"

Löscht den Inhalt des gesuchten Elements, indem die Ausgabe unterbrochen wird.

- **See** xml_line::get_attribute()
- **Access** public

function xml_line::delete_element([$limit = 0], [$path = ""], [$pattern = ""], [$attrn = ""], [$elcount = ""]) *[line 802]*

Function Parameters:

- *integer* **$limit** optional Anzahl der gesuchten Treffer. Danach wird die Funktion beendet

- *string* **$path** optional root/element1/element2 (kann unvollständig sein, z.B. element2)
- *string* **$pattern** optional String, der in den gesuchten Daten enthalten sein muss
- *mixed* **$attrn** optional array(name=>value, name2=>value2) Attribut(e) müssen enthalten sein.
- *string* **$elcount** optional "1-6-2" oder "1-4-6:2" oder "1-6-{3 to 5}" oder "1-6-{3,5,8}"

Löscht das gesuchte Element und seinen Inhalt, indem die Ausgabe unterbrochen wird.

- **See** xml_line::get_attribute()
- **Access** public

function xml_line::find_data_by_id($value, [$idname = "id"]) *[line 924]*

Function Parameters:

- *string* **$value**
- *string* **$idname** optional (anderer Atrributname, wenn nicht "id")

Findet und liefert den Elementinhalt mit einem bestimmten id-Attribut, z.B. find_data_by_id(15); <Kiste id="1">Schrauben</Kiste>

Shorthand für get_data(). Heißt "id" nicht "id" sondern "nr", kann der 2. Parameter gesetzt werden. Damit wird aus der Funktion eigentlich ein find_data_by_attribute_name_value.

- **Access** public

function xml_line::get_attribute([$limit = 0], [$path = ""], [$pattern = ""], [$attrn = ""], [$elcount = ""]) *[line 636]*

Function Parameters:

- *integer* **$limit** optional 0 bedeutet: kein Limit
- *string* **$path** optional
- *string* **$pattern** optional
- *string* **$attrn** optional
- *string* **$elcount** optional

Setzt die Abfrage für die Werte des Attributs oder der Attribute, die der Vorgabe entsprechen. Ohne Parameter werden alle Werte in das Ergebnis-Array geschrieben.

- **Access** public

function xml_line::get_data([$limit = 0], [$path = ""], [$pattern = ""], [$attrn = ""], [$elcount = ""]) *[line 740]*

Function Parameters:

- *integer* **$limit** optional Anzahl der gesuchten Treffer. 0 = keine Einschränkung
- *string* **$path** optional root/element1/element2 (kann unvollständig sein, z.B. element2)
- *string* **$pattern** optional String, der in den gesuchten Daten enthalten sein muss
- *mixed* **$attrn** optional array(name=>value, name2=>value2) Attribut(e) müssen enthalten sein.
- *string* **$elcount** optional "1-6-2" oder "1-4-6:2" oder "1-6-{3 to 5}" oder "1-6-{3,5,8}"

Ruft set_action auf und übergibt das Kommando "find_data" und die Suchkriterien als Parameter.

set_action wird bei fast jedem Event des Expat-Parsers aufgerufen, und ruft wiederum die funktion find_data (und die übrigen gesetzen Funktionen) auf.

- **Access** public

function xml_line::get_element_counter_array() *[line 1359]*

Experimentell.

Liefert das Array zurück, in dem die Elemente gezählt werden. Kann für statistische Zwecke genutzt werden. Die Funktion setzt voraus, dass der Zähler gesetzt wurde. Anzeige im Browser z.B. mit print_r(get_element_counter_array())

- **Access** public

function xml_line::get_output() *[line 1339]*

Liefert den Output als String, wenn diese Variante gewählt wurde.

Es wird geprüft, ob entweder der XML- oder der CSV-Buffer Inhalt hat.

- **Access** public

function xml_line::get_record([$limit = 0], [$path = ""], [$pattern = ""], [$attrn = ""], [$elcount = ""]) *[line 758]*

Function Parameters:

- *integer* **$limit**
- *string* **$path**
- *string* **$pattern**
- *mixed* **$attrn**
- *string* **$elcount**

Übergibt das Kommando, alle Elemente innerhalb dieses Elementes als Ergebnis zurückzuliefern.

Alle enthaltenen Elemente werden geliefert, in dem ein Zeiger gesetzt wird, der erst

wieder entfernt wird, wenn das Schlusstag gefunden wurde.

- **Access** public
- **See** xml_line::get_data()

function xml_line::insert_before_end($arg, $insert) *[line 908]*

Function Parameters:

- *array* **$arg**
- *string* **$insert**

Fügt einen String vor dem End-Tag eines Elements ein.

- **See** xml_line::get_data()
- **Access** public

function xml_line::insert_element($arg, $insert) *[line 866]*

Function Parameters:

- *array* **$arg**
- *string* **$insert**

Fügt Element(e) oder XML-Code an der gesuchten Stelle ein.

Der Inhalt wird direkt hinter dem Start-Tag eingesetzt. Man kann dadurch auch den Zwischenraum zwischen zwei Starttags füllen, selbt wenn dort kein Leerzeichen ist. **wort**text . Eingefügt wird so: neuer Text**wort**text Es können auch leere Elemente gefüllt werden: <textarea></textarea> <textarea>Text</textarea>

- **See** xml_line::get_data()
- **Access** public

function xml_line::insert_element_after_element($arg, $insert) *[line 895]*
Function Parameters:

- *array* **$arg**
- *string* **$insert**

Fügt XML-Code oder einen String hinter dem End-Tag eines Elements ein.

- **See** xml_line::get_data()
- **Access** public

function xml_line::replace_attribute($arg, $newattr) *[line 678]*
Function Parameters:

- *array* **$arg**
- *array* **$newattr**

Ersetzt das angegebene Attribut durch ein anderes.

- **Access** public
- **See** xml_line::get_attribute()

function xml_line::replace_content($arg, $replace) *[line 847]*

Function Parameters:

- *array* **$arg**
- *string* **$replace**

== replace_content Ist identisch mit delete_content. Zusätzlich werden Daten in einen Speicher (fill) geschrieben und dieser beim Erreichen der Marke in die Ausgabe eingefügt.

- **See** xml_line::get_data()
- **Access** public

function xml_line::replace_element($arg, $replace) *[line 816]*

Function Parameters:

- *array* **$arg**
- *string* **$replace**

Ist identisch mit delete_element. Zusätzlich werden Daten in einen Speicher (fill) geschrieben und dieser beim Erreichen der Marke in die Ausgabe eingefügt.

- **See** xml_line::get_data()
- **Access** public

function xml_line::set_action($func, [$arg = ""], [$command = ""], [$value = ""]) *[line 323]*

Function Parameters:

- *$func* **$func** string
- *$arg* **$arg** mixed (string or array)
- *$command* **$command** string
- *$value* **$value** mixed (string or array)

Setzt den Namen der Funktion, die auf den Stream zugreift und übergibt eine Parameterliste oder -string

function xml_line::set_element_counter([$value = true]) *[line 346]*

Function Parameters:

- *boolean* **$value** optional

Setzt den Elementcounter. Dadurch ist die Adressierung paralleler Elemente möglich.

- **Access** public

array function xml_line::xml_stream() *[line 396]*

Startet den Parser und gibt die gefundenen Daten oder false zurück.

- **Access** public

Appendices

Appendix A - Class Trees

Package default

xml_line

- xml_line

Appendix D - Todo List

In Package default

In **xml_line**

- (!) Datei-Umbenennungsfunktionen, falls Zieldatei die Quelldatei ersetzen soll. -(!!!) Quell-XML wird nur aus einer Datei nicht aus String angenommen. -(!!!) Kopieren von ganzen XML-Abschnitten oder Elementen ist noch nicht möglich

Index

C

X