[Klasse] Xml_Reader

am 01.10.2008 16:37 schrieb TBT

[Klasse] Xml_Reader

hier eine kleine Klasse von mir,
die es erlaubt XML Strukturen nach Wunsch zu parsen,
und dann mit Iterator Funktionalitäten auf die Inhalte zuzugreifen.

PHP Code

```php
<?php

/**
 * Klasse um XML Dateien zu parsen und per Iterator auf die Daten zuzugreifen
 *
 * @version  $Id: xml_reader.class.php,v 1.1 2008-10-01 07:47:00 sven Exp $
 * @author     Sven Denkert
 * @todo       Übernahme von Attributen der Tags
 **/

class Xml_Reader implements Iterator {

   private $data      = null;      // enthält die geparsten XML Daten
   private $def       = array();   // enthält den XML Aufbau

   private $pointer   = null;      // Pointer für die Iterator Methoden

   /**
    * Konstruktor
    *
    * @param    array $def - Definition des zu erwartenden XML Dokumentes
    * @return   void
    * @todo
    **/
   public function __construct( array $def ) {
      //
      $this->def = $def;
   }

   /**
    * Daten von einer Adresse laden und in unser internes Array parsen
    *
    * @param    string url - Adresse von welcher geladen wird
    * @param    int timeout - maximale Wartezeit
    * @return   bool erfolgreich ja/nein
    * @todo
    **/
   public function loadUrl( $url, $timeout=5 ) {
      // Timeout definieren
      $t = stream_context_create(array(
          'http' => array(
              'timeout' => $timeout
          )
      ) );
      // manche Server antworten ohne User-Agenten nicht
      ini_set( 'user_agent', 'Mozilla/5.0(compatible; Konqueror/3.5; Linux) KHTML/3.5.9(like Gecko) (Debian)' );
      $input = file_get_contents( $url, 0, $t );
      // erstmal die Kodierung ermitteln und korrekt setzen wenn angegeben
      if( preg_match( '#<\?xml.*encoding="([^"]*)"#Uis', substr( $input, 0, 200 ), $code ) ) {
         $input = iconv( $code[1], 'UTF-8', $input );
      }
      // internen Parser anwerfen
      return $this->loadData( $input );
   }

   /**
    * die Daten aus einem String laden
    * auch für interne Rekursion nötig
    *
    * @param    string input - Eingabezeichenkette
    * @return   bool erfolgreich ja/nein
    * @todo
    **/
   public function loadData( $input ) {
      //
      $answer = $this->parse( $this->def, $input );
      $this->pointer = 0;
      return $answer;
   }

   /**
    * sucht einen bestimmten Tag, und gibt die Inhalte zurück
    *
    * @param    string text - zu durchsuchende Zeichenkette
    * @param    string search - gesuchter Tag
```

```
76     * @param    &array answer - Antwort Array
77     * @return   int Anzahl der gefundenen Datensätze
78     * @todo
79     **/
80    protected function find( $text, $search, array & $answer ) {
81       //
82       $answer = array();
83       // den gewünschten Tag extrahieren
84       if( preg_match_all( '#<('.preg_quote($search).')( [^>]*)?>(.*)</\\1>#isU', $text, $test ) ) {
85          $answer = $test[3];
86       }else{
87          // nochmal mit strpos arbeiten, da preg_match bei langen Texten versagt!
88          while( ( $start = strpos( $text, '<'.$search ) ) !== false ) {
89             $end = strpos( $text, '</'.$search.'>' );
90             $singletext = substr( $text, $start, $end-$start );
91             $braked = strpos( $singletext, '>' );
92             $singletext = substr( $singletext, $braked+1 );
93             $text = substr( $text, $end+10 );
94             $answer[] = $singletext;
95          }
96       }
97       //
98       return count( $answer );
99    }
100
101   /**
102    * parst die übergebenen Daten und erzeugt die interne rekursive Struktur
103    *
104    * @param    array def - Definition der zu lesenden XML-Struktur
105    * @param    string input - zu parsende Zeichenkette
106    * @return   bool erfolgreich ja/nein
107    * @todo
108    **/
109   private function parse( array $def, $input ) {
110      //
111      $a = false;
112      $answer = array();
113      if( is_array( $def ) ) {
114         foreach( $def as $key => $val ) {
115            if( is_array( $val ) ) {
116               if( $this->find( $input, $key, $answer ) ) {
117                  foreach( $answer as $a => $x ) {
118                     $c = count( $this->data );
119                     $this->data[$c][$key] = new Xml_Reader( $def[$key] );
120                     $this->data[$c][$key]->loadData( $answer[$a] );
121                  }
122                  $a = true;
123               }
124            }else{
125               if( $this->find( $input, $val, $answer ) ) {
126                  $this->data[0][$val] = $answer[0];
127                  $a = true;
128               }
129            }
130         }
131      }
132      return $a;
133   }
134
135   /**
136    * magische __get Funktion für den Zugriff auf die privaten Daten
137    *
138    * @param    string varname - Name der Membervariablen
139    * @return   mixed Daten / Objekte / null
140    * @todo
141    **/
142   public function __get( $varname ) {
143      return isset( $this->data[$this->pointer][$varname] )
144         ? $this->data[$this->pointer][$varname]
145         : null;
146   }
147
148   /**
149    * Rückgabe der Anzahl der enthaltenen(Sub)Datensätze
150    *
151    * @param    void
152    * @return   void
153    * @todo
154    **/
155   public function count()       {
156      return isset( $this->data ) ? count( $this->data ) : 0;
157   }
158
159   /**
160    * abstrakte Methoden für den Iterator
161    *
162    * @param    void
163    * @return   mixed
164    * @todo
```

```
165     **/
166     public function key()       {   return $this->pointer; }
167     public function valid()      {   return isset( $this->data[$this->pointer] ); }
168     public function current()    {   return isset( $this->data[$this->pointer] ) ? $this->data[$this->pointer] : false; }
169     public function next()       {   return isset( $this->data[++$this->pointer] ); }
170     public function prev()       {   return isset( $this->data[--$this->pointer] ); }
171     public function rewind()     {   return isset( $this->data[$this->pointer=0] ); }
172     public function end()        {   return isset( $this->data[$this->pointer=(count($this->data)-1)] ); }
173 }
174 ?>
```

Nutzung:

PHP Code

```
 1 // XML Aufbau definieren
 2 $def =
 3    array( 'urlset' =>
 4       array( 'url' =>
 5          array( 'loc', 'changefreq', 'lastmod', 'priority' ) ) );
 6 // URL zum abrufen
 7 $url = 'http:// ... DOMAIN ... /sitemap.xml';
 8 // Objekt erzeugen
 9 $obj = new Xml_Reader( $def );
10 // Daten laden mit 5 Sekunden Timeout
11 if( $obj->loadUrl( $url, 5 ) ) {
12    // Anzahl der enthalten URLs
13    echo "enthaltene URL: ".$obj->urlset->count()."\n";
14    // alle URL's ausgeben
15    do {
16       echo( $obj->urlset->url->loc )."\n";
17    }while( $obj->urlset->next() );
18 }
```