Antoine de Roquemaurel

Stage effectué chez Memobox – 2G technologies
Du 10 Avril 2012 au 22 Juin 2012

Pour M. Denis Mallet
Pour M. Patrick Magnaud

Rapport de stage

Refonte du système multilingue pour une application web

— Annexes —

Memobox – 2G Technologies
12 Bv de l'Europe
31850 – Montrabé

Version du 15 juin 2012

Antoine de Roquemaurel

Stage effectué chez Memobox – 2G technologies
Du 10 Avril 2012 au 22 Juin 2012

Pour M. Denis Mallet
Pour M. Patrick Magnaud

Rapport de stage
Refonte du système multilingue pour une application web

— Annexes —

Memobox – 2G Technologies
12 Bv de l'Europe
31850 – Montrabé

Table des annexes

Annexe B

Glossaire

Adobe AIR Adobe Integrated Runtime, anciennement nommé Apollo, est une machine virtuelle multiplateforme, multilangage, multi interface qui s'exécute sur le système d'exploitation, qui permet de développer une application native avec des technologies Web.

AJAX Asynchronous Javascript and XML, c'est un ensemble de technologies destinées à réaliser de rapides mises à jour du contenu d'une page Web, sans qu'elles nécessitent le moindre rechargement visible par l'utilisateur de la page Web. Les technologies employées sont diverses et dépendent du type de requêtes que l'on souhaite utiliser, mais d'une manière générale le Javascript est constamment présent.

CSS Cascading Style Sheets, un langage permettant de décrire la présentation des documents HTML.

Expressions régulières Chaîne de caractères qui décrit un ensemble de chaînes de caractères possibles selon une syntaxe précise. Elles sont aujourd'hui utilisées par les informaticiens dans l'édition et le contrôle de texte ainsi que dans la manipulation des langues.

GFT Gestion Financière des Télécoms(TEM en anglais). Ensemble des outils et ressources qui ont pour objectif d'optimiser et de rationaliser la gestion des dépenses télécoms (Mobile / Fixe, Voix / data, VoIP) des entreprises.

JavaScript Langage de programmation de scripts principalement utilisé dans les pages web interactives côté client.

HTML Hypertext Markup Language, format de données conçu pour représenter les pages web.

Méthodes Agiles Groupes de pratiques pouvant s'appliquer aux projets de développement en informatique. Les méthodes agiles impliquent au maximum le client et permettent une grande réactivité à ses demandes. Elles visent la satisfaction réelle du besoin du client en priorité aux termes d'un contrat de développement.

MySQL Système de gestion de base de données libres, principalement couplé à des serveurs web.

PHP Hypertext Preprocessor, langage de scripts libre principalement utilisé pour produire des pages web dynamiques via un serveur HTTP.

SaaS Software as a Service, technologie consistant à fournir des services ou des logiciels informatiques par le biais du Web et non plus dans le cadre d'une application de bureau ou client-serveur.

SQL Structured Query Language, pseudo langage informatique (de type requête) standard et normalisé, destiné à interroger ou à manipuler une base de données relationnelle.

Annexe C

Liste des figures

Annexe D

Liste des tableaux

Annexe

E

Liste des codes sources

Annexe F

Code source

F.1 Moteur de traduction

```php
<?php
class TLanguage {
  private static $instance;
  private $dico = array();
  private $sTranslate = '';

  public static function singleton() {
    if (!isset (self :: $instance)) {
      $c = __CLASS__;
      self :: $instance = new $c ();
      return (self :: $instance);
    }
    return (self :: $instance);
  }

  public static function TraducExist($xsLangue) {
    return (($xsLangue == 'fr') || ($xsLangue == 'en') || ($xsLangue == 'es'))
        ;
  }

  //Necessaire pour l'API gOwlReport
  public function getLang($xsValue,$xbDebugMode=false){
    $bEtatDebugMode = DEBUG_MODE;
    if($xbDebugMode)
      $this->_xbDebugMode = TRUE;
    $sTraduction = $this->getTranslation($xsValue);
    $this->_xbDebugMode = $bEtatDebugMode;
    return $sTraduction;
  }

  private function pushInDico($idRes, $xaValue) {
    if(isset($xaValue['Language'])) {
      $this->dico[$idRes] = $xaValue['Language'];
      $this->dico[$idRes]['TYPE_RES'] = $xaValue['Type_RES'];
    }
  }

  /**
   * Retourne la langue de traduction
   */
  public function getCurrentLanguage() {
    $sLangReturn = '';
    $access_manager = TDataAccessManager :: singleton();

```

```
44      if ($access_manager->sessionLoaded()) { // on recupere la langue de la
            session
45        $sLangReturn = $access_manager->getContextManager(CONTEXT_LANGUE,
              ALL_MODULE);
46        if(!$sLangReturn) {
47          try {
48            $G_Langue = $access_manager->getWebObject('G_Langue');
49            $sLangReturn = $G_Langue->getText();
50          } catch (exception $e) {
51            $sLangReturn = null;
52          }
53
54        }
55      } else { // on repere la langue du navigateur
56        $sLangReturn = ($this->returnLanguage());
57        if (!self::TraducExist($sLangReturn))
58          $sLangReturn = LANG_DEFAULT;
59      }
60
61      return ($sLangReturn);
62    }
63    /**
64     * Retourne un array traduit contenant toutes les traductions
65     * @param $xaId Array contenant des constantes de traduction.
66     * @param $aMapping
67     * @return L'array contenant toutes les traductions
68     */
69    public function getArrayTranslate($xaId, $aMapping = NULL){
70      $aResult = array();
71      if(is_array($xaId)){
72        if(TArray::isArraySetAndSupTo0($xaId)){
73          foreach($xaId as $ind => $keyVal){
74            $keyVal = (($aMapping != NULL) && array_key_exists($keyVal,$aMapping
                  ))? $aMapping[$keyVal] : $keyVal;
75            $aResult[$ind] = $this->getTranslation($keyVal);
76          }
77        }
78      }else{
79        $keyVal = (($aMapping != NULL) && array_key_exists($xaId,$aMapping))?
              $aMapping[$xaId] : $xaId;
80        $aResult = $this->getTranslation($keyVal);
81      }
82      return $aResult;
83    }
84
85
86    /**
87     * Retourne une chaine traduite, on peut faire passer les variables
            a integrer dans la traduction
88     *  soit en paramee de methode getTranslation('TRANS_MONID', var1, var2,
            varn) ;
89     *  soit integres dans l'ID getTranslation('TRANS_MONID{{var1, var2, varn
            }}');
90     * @param $xIdToTranslate
91     * @return Chaine traduite
92     */
93    public function getTranslation($xIdToTranslate) {
94      $sRetour = '';
95      $i = 1;
96      $typeWordToTranslate = '';
```

```
97      $variables = func_get_args();
98      // Si on a fait passe les variables dans un tableau
99      if(isset($variables[1]) && is_array($variables[1])) {
100       $aVarBuff = $variables[1];
101       foreach($aVarBuff as $var) {
102         $variables[$i] = $var;
103         ++$i;
104       }
105     }
106
107     // on repere le type de mot a afficher (court long ou normal)
108     $typeWordToTranslate = $this->getTypeWord($xIdToTranslate);
109
110     //Si les variables sont passe en ID grace a  {var1, var2, var3}}
111     if(strpos($xIdToTranslate, '{{')!==false){
112       $aVariables = array();
113       $aVariablesBuff = array();
114       preg_match_all('#{{(.+)}}#isU', $xIdToTranslate, $aVariablesBuff);
115       $xIdToTranslate = preg_replace('#{{(.+)}}#isU', '', $xIdToTranslate);
116
117       $sVariablesToSplit = $aVariablesBuff[1][0];
118       if(strpos($sVariablesToSplit, ',')!==false){
119         $aVariables = explode(',', $sVariablesToSplit);
120       } else {
121         $aVariables = $sVariablesToSplit;
122       }
123
124       return ($this->getTranslation($xIdToTranslate, $aVariables));
125     }
126
127     // si on concatene plusieurs constante avec CONST1++ ++CONST2
128     if(strpos($xIdToTranslate, '++')!==false){
129       $aId_res = explode('++',$xIdToTranslate );
130       $forbidden='"\'\?()*:/@|<>-';
131       foreach($aId_res as $id_res_temp){
132         $sRetour .= (strcspn(trim($id_res_temp),$forbidden)!==0)? $this->
              getTranslation($id_res_temp) : $id_res_temp;
133       }
134       return $sRetour;
135     }
136
137     //Si la cle existe en Constante elle est utilisee en priorite
138     if (defined($xIdToTranslate)) {
139       if (defined($xIdToTranslate.'_'.$this->getCurrentLanguage()))
140         $xIdToTranslate = $xIdToTranslate.'_'.$this->getCurrentLanguage();
141       return constant($xIdToTranslate);
142     }
143
144     // si la clef n'est pas dans le dico, on la cherche et on l'ajoute au dico
145     if(!isset($this->dico[$xIdToTranslate])) {
146       $access_manager = TDataAccessManager :: singleton();
147       $datalink = $access_manager->getDataLink();
148       $value = $datalink->RES_getDicoTranslation($xIdToTranslate, $this->
            getCurrentLanguage());
149       $this->pushInDico($xIdToTranslate, $value);
150     }
151
152     // si la clef est trouve dans le dico, on peut l'afficher
153     if(isset($this->dico[$xIdToTranslate][$this->getCurrentLanguage()])) {
```

```
154       $sRetour = $this->getValueKeyExist($xIdToTranslate, $typeWordToTranslate
              );
155     } else {
156       $sRetour = $this->getValueKeyNotExistInCurrentLang($xIdToTranslate,
              $typeWordToTranslate);
157     }
158
159     // on parse toutes les variables et constantes
160     $sRetour = $this->parserVariables($sRetour, $variables);
161     $constants = $this->getConstants($sRetour);
162     if(isset($this->dico[$xIdToTranslate][$this->getCurrentLanguage()]))
163       $sRetour = $this->parserConstantes($sRetour, $constants);
164     else
165       $sRetour = $this->parserConstantes($sRetour, $constants, 'fr');
166
167     return $sRetour;
168   }
169   /**
170    * Retourne la valeur d'une clef existante dans le dico en fonction du type
           (long short ou normal)
171    * @param String $xIdToTranslate La clef
172    * @param String $xsTypeWordToTranslate Le type
173    */
174   private function getValueKeyExist($xIdToTranslate, $xsTypeWordToTranslate,
          $xsLang='') {
175     if($xsLang == '') {
176       $xsLang = $this->getCurrentLanguage();
177     }
178
179     $sRetour = '';
180     if($xsTypeWordToTranslate == '#SHORT#' &&
181       $this->dico[$xIdToTranslate]['short'][$xsLang] != '') {
182       $sRetour = $this->dico[$xIdToTranslate]['short'][$xsLang];
183     } else if($xsTypeWordToTranslate == '#LONG#' &&
184       $this->dico[$xIdToTranslate]['long'][$this->getCurrentLanguage()] != '')
              {
185       $sRetour = $this->dico[$xIdToTranslate]['long'][$xsLang];
186     } else {
187       $sRetour = $this->dico[$xIdToTranslate][$xsLang];
188     }
189
190     return $sRetour;
191   }
192   /**
193    * Retourne l'affichage d'une clef qui n'existe pas dans la langue courante
194    * @param String $xIdToTranslate La clef
195    * @param String $xsTypeWordToTranslate Le type
196    */
197   private function getValueKeyNotExistInCurrentLang($xIdToTranslate,
          $xsTypeWordToTranslate) {
198     $sRetour = '';
199     // la clef n'existe pas pour la langue courante, on cherche si elle existe
            en francais
200     if(isset($this->dico[$xIdToTranslate]['fr'])) {
201       if(DEBUG_MODE)
202         $sRetour = '{'.$this->getValueKeyExist($xIdToTranslate,
              $xsTypeWordToTranslate, 'fr').'}';
203       else
204         $sRetour = $this->getValueKeyExist($xIdToTranslate,
              $xsTypeWordToTranslate, 'fr');
```

```
205       //la clef est inexistante.
206       } else {
207         TUtil::DebugToFile('Constante '.$xIdToTranslate. ' non trouve', '',
              DEBUG_PATH.'DebugTranslation.txt', true);
208         if(DEBUG_MODE)
209           $sRetour = '='.$xIdToTranslate.'=';
210         else
211           $sRetour = $xIdToTranslate;
212       }
213
214       return $sRetour;
215     }
216     /**
217      * Retourne le type de mot d'un ID (SHORT, LONG ou NORMAL)
218      * @param String $xIdToTranslate L'id
219      */
220     private function getTypeWord($xIdToTranslate) {
221       $sRetour = 'NORMAL';
222       if(strpos($xIdToTranslate, '#')!==false){
223         $aTypeWordBuff = array();
224         if(preg_match_all('#\#(.+)\##isU', $xIdToTranslate, $aTypeWordBuff) ==
              1) {
225           $sRetour = $aTypeWordBuff[0][0];
226         }
227       }
228
229       return $sRetour;
230     }
231
232     /**
233      * Parse une chaine de caractere en remplacant les constantes par leurs
          valeurs
234      * @param String $xstring Chaine a parser
235      * @param Array $xaconstants Constantes
236      */
237     private function parserConstantes($xstring, $xaconstants, $xslanguage='') {
238       if($xslanguage == '') {
239         $xslanguage = $this->getCurrentLanguage();
240       }
241       $sRetour = $xstring;
242       $access_manager = TDataAccessManager :: singleton();
243       $datalink = $access_manager->getDataLink();
244
245       for($i=0; $i < count($xaconstants) ; ++$i) {
246         if(!isset($this->dico[$xaconstants[$i]])) {
247           $value = $datalink->RES_getDicoTranslation($xaconstants[$i],
                $xslanguage);
248           $this->pushInDico($xaconstants[$i], $value);
249         }
250
251         // On donne la valeur aux constantes
252         if(isset($xaconstants[$i])) {
253           if(isset($this->dico[$xaconstants[$i]][$xslanguage]) &&
254           $this->dico[$xaconstants[$i]]['TYPE_RES'] == 'ITEM') {
255             $sRetour = preg_replace('#\[(.+)\]#iSU',
256               $this->dico[$xaconstants[$i]][$xslanguage], $sRetour, 1);
257           } else if(isset($this->dico[$xaconstants[$i]]['fr'])) {
258             $sRetour = preg_replace('#\[(.+)\]#iSU',
259               $this->dico[$xaconstants[$i]]['fr'], $sRetour, 1);
260           } else {
```

```
261           $sRetour = preg_replace('#\[(.+)\]#iSU', '$1', $sRetour, 1);
262         }
263       }
264     }
265
266     return $sRetour;
267   }
268
269   /**
270    * Parse une chaine de caractere en remplacant les variables par leurs
          valeurs
271    * @param String $xstring Chaine a  parser
272    * @param Array $xavariables Variables
273    */
274   private function parserVariables($xstring, $xavariables) {
275     $sRetour = $this->parserPlurialOrSingular($xstring, $xavariables);
276     for($i=1; $i < count($xavariables)  ; ++$i) {
277       // On donne la valeur aux variables
278       if(isset($xavariables[$i])) {
279         $sRetour = preg_replace('#<(.+)>#iSU', $xavariables[$i], $sRetour, 1);
280       }
281     }
282
283     return $sRetour;
284   }
285
286   /**
287    * Met au pluriel ou au singulier une chaine en fonction des variables qu'
          elle contient.
288    * @param String $xstring Chaine a  accorder contenant des <variable> et {
          singulier|pluriel}
289    * @param Array $xaVariables Tableau de variables
290    * @return La chaine accordee
291    */
292   private function parserPlurialOrSingular($xstring, $xaVariables) {
293     $aStringSplitVariables = preg_split('#\<[a-z]{1,}\>#i', $xstring);
294     // Met au pluriel ou au singulier
295     for($i=1; $i < count($xaVariables) ; ++$i) {
296       if(isset($aStringSplitVariables[$i])) {
297         if($xaVariables[$i] > 1) { // Pluriel
298           $aStringSplitVariables[$i] = preg_replace('#\((.+)\|(.+)\)#isU', '$2
              ',
299                                     $aStringSplitVariables[$i]);
300         } else {
301           $aStringSplitVariables[$i] = preg_replace('#\((.+)\|(.+)\)#isU', '$1
              ',
302                                     $aStringSplitVariables[$i]);
303         }
304       }
305      }
306
307     //On Reconcatene la string en remettant les <value> servants de
          separateurs precedemment
308     $sRetour = '';
309     $i=0;
310     foreach ($aStringSplitVariables as $astring) {
311       $sRetour .= $astring;
312       if($i < count($aStringSplitVariables)-1 ){
313         $sRetour .= '<value>';
314       }
```

```
315       ++$i;
316     }
317     return $sRetour;
318   }
319
320   /**
321    * Obtient un tableau de constantes a  partir d'une chaine de caractere
          contenant des [constante]
322    * @param string $xsValueToParse Chaine a  parser
323    */
324   private function getConstants($xsValueToParse) {
325     $aConstants = array();
326     $aConstantesBuff = array();
327     preg_match_all('#\[(.+)\]#isU', $xsValueToParse, $aConstantesBuff);
328     $aConstants = $aConstantesBuff[1];
329
330     return $aConstants;
331   }
332
333   /**
334    * Retourne la langue du navigateur
335    */
336   private function returnLanguage() {
337     $retRL = LANG_DEFAULT; // Langue par defaut
338     if (isset ($_SERVER)) {
339       if (isset ($_SERVER['HTTP_ACCEPT_LANGUAGE'])) {
340         $retRL = explode(',', $_SERVER['HTTP_ACCEPT_LANGUAGE']);
341         $retRL = substr($retRL[0], 0, 2);
342       }
343     }
344     if (!$this->traducExist($retRL))
345       $retRL = LANG_DEFAULT;
346
347     return strtolower($retRL);
348   }
349 }
350 ?>
```

Listing F.1. TLanguage.class.php

F.2 Memolanguage

F.2.1 Le serveur

```
1  <?php
2  class DataBase {
3    public function __construct($xsDbName, $xsDbHost, $xsDbUser, $xsDbPassword)
       {
4      $this->connect($xsDbName, $xsDbHost, $xsDbUser, $xsDbPassword);
5    }
6
7    public function connect($xsDbName, $xsDbHost, $xsDbUser, $xsDbPassword) {
8      $this->db = new PDO('mysql:host='.$xsDbHost.';dbname='.$xsDbName.'',
9                          $xsDbUser, $xsDbPassword);
10     $this->db->setAttribute(\PDO::ATTR_ERRMODE, \PDO::ERRMODE_EXCEPTION);
11   }
12 }
```

Listing F.2. database.class.php

```php
<?php
class DatabaseSearch extends Database {
  public function __construct() {
    parent::__construct(DBNAME, DBHOST, DBUSER, DBPASS);
  }

  public function getValueConstantsFr($xaConstants, $xsLanguage, $xsTable = DBTABLENAMENOUVELLE) {
    $aRetour = $xaConstants;
    $i = 0;

    if(isset($query)) {
        while($data = $query->fetch()) {
          $aRetour[$i] = $data;
          ++$i;
        }
    }

    return $aRetour;
  }

  public function getTranslateSearchText($xaKeyword, $xaLangResult, $xaLangSearch, $xbValide, $xsTable = DBTABLENAMENOUVELLE) {
    $aRetour = array();
    $i = 0;
    $fctWholeWord = util::testGet('whole');
    $fctRegex = util::testGet('regex');


      $txtQuery = "SELECT ID_RES, TranslatedText, Language, Valide ";
      if($xsTable == DBTABLENAMENOUVELLE)
      $txtQuery .= ", Type_RES, TranslatedText_short, TranslatedText_long ";


      $txtQuery .= "FROM ".$xsTable." WHERE ID_RES IN (SELECT ID_RES FROM ".$xsTable."
      WHERE " ;


      foreach($xaKeyword as $k) {
          $sKBuff = trim($k);
        if($fctRegex == 1) {// || $fctWholeWord == 1) {
          $txtQuery .= "lower(TranslatedText) REGEXP '$k' ";
        } else if($fctWholeWord == 1){
        $txtQuery .= 'lower(TranslatedText) REGEXP \'^'.$sKBuff.$sCarForbidden.'|'.$sCarForbidden.$sKBuff.
        '$|'.$sCarForbidden.$sKBuff.$sCarForbidden.'|^'.$sKBuff.'$\' ';
        } else {
          $txtQuery .= "lower(TranslatedText) LIKE '%$k%' ";
        }
        if($i < count($xaKeyword) - 1)
          $txtQuery .= ' AND ';

        ++$i;
      }
      if($xbValide != 2)
        $txtQuery .= "AND Valide='$xbValide'";
```

```
54
55        $txtQuery .= "AND Language='$xaLangSearch')";
56
57      for($i=0 ; $i < count($xaLangResult) ; ++$i) {
58        if($i == 0)
59          $txtQuery .= " AND (";
60        else
61          $txtQuery .= " OR ";
62
63          $txtQuery .= "Language='$xaLangResult[$i]'";
64
65          if($i == count($xaLangResult) - 1)
66            $txtQuery .= ")";
67        }
68        $txtQuery .= " ORDER BY ID_RES LIMIT 50";
69
70        $query = $this->db->prepare($txtQuery);
71
72          $query->execute();
73
74
75        while($data = $query->fetch()) {
76          $data['TranslatedText'] = Util::parserConstantes($data['TranslatedText
              '], $data['Language'], $this);
77          if($xsTable == DBTABLENAMENOUVELLE) {
78          $data['TranslatedText_short'] = Util::parserConstantes($data['
              TranslatedText_short'], $data['Language'], $this);
79          $data['TranslatedText_long'] = Util::parserConstantes($data['
              TranslatedText_long'], $data['Language'], $this);
80          }
81        $aRetour[$i] = $data;
82          ++$i;
83        }
84        $query->closeCursor();
85
86        return $aRetour;
87    }
88    public function getTraduceEffectue($id, $xsTable = DBTABLENAMENOUVELLE) {
89      $i = 0;
90      $aRetour = array();
91        $query = $this->db->prepare("SELECT * FROM ".$xsTable."
92        WHERE lower(ID_RES) = :keyword ORDER BY Language LIMIT 50");
93        $query->execute(array('keyword' => $id));
94        while($data = $query->fetch()) {
95          $aRetour[$i] = $data;
96          ++$i;
97        }
98        $query->closeCursor();
99
100       return $aRetour;
101   }
102
103   public function getTranslateSearchId($xsId, $xaLangResult, $xaLangSearch,
104                             $xbValide, $xsTable = DBTABLENAMENOUVELLE,
                                  $xbStrictID=0) {
105     $aRetour = array();
106     $i = 0;
107
108     $txtQuery = "SELECT ID_RES, TranslatedText, Language, Valide";
109     if($xsTable == DBTABLENAMENOUVELLE)
```

```
110       $txtQuery .= ", Type_RES, TranslatedText_short, TranslatedText_long ";
111
112     $txtQuery .= " FROM ".$xsTable."";
113     if($xbStrictID == 1) {
114       $txtQuery .= " WHERE lower(ID_RES) = :id ";
115     } else {
116       $txtQuery .= " WHERE lower(ID_RES) LIKE :id ";
117       $xsId = '%'.$xsId.'%';
118     }
119     // si $xbValide = 2 c'est qu'on demande les valides et les non valides
120       if($xbValide != 2) {
121         $txtQuery .= "AND Valide=:valide";
122       } else {
123         $txtQuery .= ":valide";
124         $xbValide = "";
125       }
126
127     for($i=0 ; $i < count($xaLangResult) ; ++$i) {
128       if($i == 0)
129         $txtQuery .= " AND (";
130       else
131         $txtQuery .= " OR ";
132
133         $txtQuery .= "Language='$xaLangResult[$i]'";
134
135         if($i == count($xaLangResult) - 1)
136           $txtQuery .= ")";
137       }
138     $txtQuery .= 'ORDER BY ID_RES LIMIT 50';
139       $query = $this->db->prepare($txtQuery);
140       $query->execute(array('id' => $xsId,
141                   'valide' => $xbValide));
142
143       while($data = $query->fetch()) {
144         $aRetour[$i] = $data;
145         ++$i;
146       }
147       $query->closeCursor();
148
149       return $aRetour;
150   }
151
152   public function getLanguagesId($xsId, $xsTable = DBTABLENAMENOUVELLE) {
153     $aRetour = array();
154     $i = 0;
155
156       $query = $this->db->prepare("SELECT ID_RES, TranslatedText, Language
            FROM ".$xsTable."
157       WHERE ID_RES = :id LIMIT 50");
158       $query->execute(array('id' => $xsId));
159
160       while($data = $query->fetch()) {
161         $aRetour[$i] = $data;
162         ++$i;
163       }
164       $query->closeCursor();
165
166       return $aRetour;
167   }
```

```
168   public function getRemainingTranslation($xsId, $xsTable =
          DBTABLENAMENOUVELLE) {
169     $aLanguageBool = array('fr' => false, 'en' => false, 'es' =>false);
170     $aRetour = array();
171     $i = 0;
172
173       $query = $this->db->prepare("SELECT Language FROM $xsTable
174       WHERE ID_RES = :id");
175       $query->execute(array('id' => $xsId));
176
177       while($data = $query->fetch()) {
178         $aLanguageBool[$data['Language']] = true;
179       }
180       if(!$aLanguageBool['fr']) {
181         $aRetour[$i] = 'fr';
182         ++$i;
183       }
184       if(!$aLanguageBool['en']) {
185         $aRetour[$i] = 'en';
186         ++$i;
187       }
188       if(!$aLanguageBool['es']) {
189         $aRetour[$i] = 'es';
190         ++$i;
191       }
192
193       $query->closeCursor();
194
195       return $aRetour;
196   }
197
198 }
```

Listing F.3. databaseSearch.class.php

```
1  <?php
2  require_once('class/util.class.php');
3  Util::CreateDefine();
4
5  require_once('class/database.class.php');
6  require_once('class/databaseSchemaTable.class.php');
7  require_once('class/databaseChanges.class.php');
8  require_once('class/databaseSearch.class.php');
9
10 require_once('class/page.class.php');
11 require_once('class/pageselect.class.php');
12 require_once('class/pageinsert.class.php');
13 require_once('class/pageupdate.class.php');
14 require_once('class/pagedelete.class.php');
15
16 $titrePage = Util::testGet('p');
17 switch($titrePage) {
18   case 'select':
19     $page = new PageSelect();
20     break;
21   case 'insert':
22     $page = new Pageinsert();
23     break;
24   case 'delete':
25     $page = new PageDelete();
```

```
26      break;
27    case 'update':
28      $page = new PageUpdate();
29      break;
30    default:
31      exit();
32  }
33
34  $page->display();
```

Listing F.4. index.php

```
1   <?php
2
3   abstract class Page {
4     protected $sText;
5     protected $db;
6     private $dbSchemaTable;
7
8     public function __construct() {
9     }
10
11    public function display() {
12      echo $this->sText;
13    }
14
15    abstract protected function initText() ;
16
17  }
```

Listing F.5. page.class.php

```
1   <?php
2
3   class PageSelect extends Page {
4     private $sKeyword;
5     private $sTypeSearch;
6     private $aLangResult;
7     private $aLangSearch;
8     private $bValide;
9     private $sTable;
10    private $bSearchStrictId;
11    private $dbSchemaTable;
12
13    public function __construct() {
14      parent::__construct();
15      $this->dbSchemaTable = new DatabaseSchemaTable();
16      $this->db = new DatabaseSearch();
17
18      $this->sKeyword = addslashes(Util::testGet('k'));
19      $this->sTypeSearch = addslashes(Util::testGet('type'));
20      $this->aLangResult = explode('|', addslashes(Util::testGet('langResult')))
          ;
21      $this->aLangSearch = addslashes(Util::testGet('langSearch'));
22      $this->bValide = addslashes(Util::testGet('valide'));
23      $this->sTable = addslashes(Util::testGet('table'));
24      $this->bSearchStrictId = addslashes(Util::testGet('searchStrictId'));
25      if($this->bSearchStrictId == '') {
26        $this->bSearchStrictId = 0;
27      }
```

```
28      if($this->sTable == '' && !($this->sTypeSearch == "structTableMaxLength")) {
29        $this->sTable = DBTABLENAMENOUVELLE;
30      }
31
32      if($this->sKeyword == ''&& !($this->sTypeSearch == "structTableMaxLength") )
33        exit();
34
35      $this->initText();
36    }
37
38    protected function initText() {
39      switch($this->sTypeSearch) {
40        case 'id':
41          $this->setTextSearchId();
42          break;
43        case 'text':
44          $this->setTextSearchText();
45          break;
46        case 'remainingTranslation':
47          $this->setTextRemainingTranslation();
48          break;
49        case 'structTableMaxLength' :
50          $this->setTextMaxLength();
51          break;
52      }
53    }
54
55    private function setTextSearchText() {
56      $aKeywords = explode('-', $this->sKeyword);
57      $aTranslate = $this->db->getTranslateSearchText($aKeywords, $this->
           aLangResult, $this->aLangSearch,
58             $this->bValide, $this->sTable);
59      $this->textDisplaySearch($aTranslate);
60    }
61
62    private function setTextSearchId() {
63      $aTranslate = $this->db->getTranslateSearchId($this->sKeyword, $this->
           aLangResult,
64               $this->aLangSearch,$this->bValide, $this->sTable, $this->
                  bSearchStrictId);
65      $this->textDisplaySearch($aTranslate);
66    }
67
68    private function textDisplaySearch($xaTranslate) {
69      foreach($xaTranslate as $aTranslate) {
70        $this->sText .= $aTranslate['ID_RES'];
71        $aLangueTranslate = $this->db->getTraduceEffectue($aTranslate['ID_RES'],
             $this->sTable);
72        $this->sText .= SEPARATEURCOLONNE;
73        $this->sText .= $aTranslate['Language'];
74
75        $this->sText .= SEPARATEURLANGUE;
76        foreach($aLangueTranslate as $langue) {
77          $this->sText .= $langue['Language'];
78          $this->sText .= " ";
79        }
80        $this->sText .= SEPARATEURLANGUE;
81        $this->sText .= SEPARATEURCOLONNE;
```

```
82        $this->sText .= $aTranslate['Valide'];
83        $this->sText .= SEPARATEURCOLONNE;
84
85        /* Si c'est la nouvelle table, c'est que le champ Type_RES existe */
86        if($this->sTable == DBTABLENAMENOUVELLE) {
87          $this->sText .= $aTranslate['Type_RES'];
88        }
89        $this->sText .= SEPARATEURCOLONNE;
90
91        $this->sText .= utf8_encode(Util::parserConstantes($aTranslate['
              TranslatedText'],
92                                $aTranslate['Language'], $this->db));
93
94        if($this->sTable == DBTABLENAMENOUVELLE) {
95          $this->sText .= SEPARATEURCOLONNE;
96          $this->sText .= $aTranslate['TranslatedText_short'];
97          $this->sText .= SEPARATEURCOLONNE;
98          $this->sText .= $aTranslate['TranslatedText_long'];
99        }
100       $this->sText .= SEPARATEURLIGNE;
101     }
102     if($xaTranslate == array()) {
103       $this->sText = "NO_FOUND";
104     }
105   }
106
107   private function setTextRemainingTranslation() {
108     $aTranslate = $this->db->getRemainingTranslation($this->sKeyword, $this->
            sTable);
109     foreach($aTranslate as $aTranslate) {
110       $this->sText .= $aTranslate;
111       $this->sText .= SEPARATEURLIGNE;
112     }
113   }
114
115   private function setTextMaxLength() {
116     $aMaxLength = $this->dbSchemaTable->getMaxLength();
117
118     foreach($aMaxLength as $maxlength) {
119       if($maxlength['COLUMN_NAME'] == 'TranslatedText_short' ||
120         $maxlength['COLUMN_NAME'] == 'TranslatedText' ||
121         $maxlength['COLUMN_NAME'] == 'TranslatedText_long') {
122           $this->sText .= $maxlength['CHARACTER_MAXIMUM_LENGTH'];
123           $this->sText .= ' ';
124       }
125     }
126   }
127 }
```

Listing F.6. pageselect.class.php

F.2.2 Le client

F.2.2.1 lib

```
1 var previousRequest;
2 var previousValue;
```

```
3  var tableau = document.getElementById("tableResults"+onglet);
4  var iterateurLanguages = 0, nbLangAffichage = 0;
5
6  var dataTraduceLang, dataCurrentLang, dataValide, dataType, dataValue;
7
8  var elementInfoTable = document.getElementById('infoTable'+onglet);
9  var elementinfos = document.getElementById('info'+onglet);
10
11 /**
12  * Effectue une requete et recupere les resultats
13  */
14 function getResults(keyword, xSearchInOldTable) {
15   clicTabForbidden();
16    elementInfos = document.getElementById('info'+onglet);
17    tableau = document.getElementById("tableResults"+onglet);
18   xTable = tableDb;
19   elementInfoTable = document.getElementById('infoTable'+onglet);
20   elementInfoTable.innerHTML = '';
21   tableau.innerHTML = '';
22   chargerConf();
23   nbResults = 0;
24   var xhr = new XMLHttpRequest();
25   keyword = keyword.replace(new RegExp(' ', 'g'), '-');
26   xhr.open('GET', adresseServeur+'?p=select&type='+typeSearch+'&k='+
        convertAccentsForGet(keyword)+
27       '&whole='+wholeWord+'&regex='+regex+'&langSearch='+
28         langSearch+'&langResult='+langResults+'&valide='+valide+'&table='+
              xTable);
29
30   if(xTable == 'RES_dicoLanguage') {
31     elementInfoTable.innerHTML = '<p><img src="img/loading.gif" alt=""/> '+
32       'Recherche en cours dans la nouvelle table</p>';
33   } else {
34     elementInfoTable.innerHTML = '<p><img src="img/loading.gif" alt=""/> '+
35       'Recherche en cours dans l\'ancienne table</p>';
36   }
37
38   if(keyword == '') {
39     elementInfoTable.innerHTML = '';
40     elementInfos.innerHTML = '';
41     clicTabAllow();
42     return ;
43   }
44
45   xhr.onreadystatechange = function() {
46     if (xhr.readyState == 4 && xhr.status == 200) {
47       displayResultsSearch(xhr.responseText, keyword, xTable,xSearchInOldTable
             );
48     }
49   };
50
51   xhr.send(null);
52
53   return xhr;
54 }
55 /**
56  * Informe l'utilisateur que des constantes ont ete trouves
57  */
58 function displayInfoCstNotFound(xkeyword) {
59   var textInfo = '';
```

```
60    textInfo = getInfoSearch(xkeyword);
61
62    if(tableDb == 'RES_dicoLanguage') {
63      tableau.innerHTML = '<div id="ajoutcstNonTrouve">Aucune constante n\'a &
          eacute;t&eacute; trouv&eacute;e ni '+
64        'dans la nouvelle table, ni dans l\'ancienne table<br />';
65    } else {
66      textInfo += '<div id="ajoutcstNonTrouve">Aucune constante n\'a &eacute;t&
          eacute; trouv&eacute;e dans l\'ancienne table<br />';
67    }
68
69    textInfo += '<center><a href="#" id="addCstAfterSearch">'+
70      'Ajouter une constante dans la nouvelle table</a></center></div>';
71
72    elementInfoTable.innerHTML = textInfo;
73
74  }
75
76  /**
77   * Informe l'utilisateur qu'aucune constante n'a ete trouvee dans les deux
        tables
78   * @param xTable
79   */
80  function displayInfoCstFound(xTable, xkeyword) {
81    var textInfo = '';
82    if(xTable != tableDb) {
83      textInfo = 'Aucune constante n\'a &eacute;t&eacute; trouv&eacute; dans la
          nouvelle table. <br/>';
84    }
85
86    textInfo = getInfoSearch(xkeyword);
87    if(nbResults > 1) {
88      textInfo += '<strong>'+(nbResults)+'</strong> constantes trouv&eacute;es
          dans '+
89        '<strong>'+nomTableToNewOrOld(xTable)+'</strong> table';
90    } else {
91      textInfo += '<strong>'+(nbResults)+'</strong> constante trouv&eacute;e
          dans '+
92        '<strong>'+nomTableToNewOrOld(xTable)+'</strong> table';
93    }
94
95    elementInfos.innerHTML = textInfo;
96  }
97
98  function getInfoSearch(xkeyword) {
99    var textInfo = '';
100   xkeyword = xkeyword.replace(new RegExp('-', 'g'), ' ');
101   textInfo += '<strong>Recherche</strong>: <span id="keywordTab'+onglet+'">'+
        xkeyword+'</span><br />';
102   textInfo += '<strong>Type de recherche</strong>: '+charTypeToString(
        typeSearch)+'<br />';
103   textInfo += '<strong>Mots entiers</strong>: '+boolWholeToString(wholeWord)+'
        <br /><hr/>';
104   document.getElementById('searchTab'+onglet).innerHTML = '<em>'+cutString(
        xkeyword.toLowerCase(),0,10)+'</em>';
105   return textInfo
106 }
107 /**
108  * Informe l'utilisateur qu'aucune constante n'a ete trouvee pour la nouvelle
        table
```

```
109  */
110 function displayInfoStartSearchOldTable(xkeyword) {
111   var textInfo = '';
112   textInfo = getInfoSearch(xkeyword);
113   textInfo += '<div id="cstNonTrouveNouvelle"><p>Aucune constante trouv&eacute
        ;e dans la nouvelle table</p></div>';
114   elementInfoTable.innerHTML = textInfo;
115 }
116
117 /**
118  *  Affiche les resultats d'une recherche
119  */
120 function displayResultsSearch(response, keyword, xTable, xsearchInOldTab) {
121   var titreCol,ligne;
122   var dataSearch = new Object();
123
124   tableau.innerHTML = "";
125   elementInfos.innerHTML = '';
126   if(keyword == '' ) {
127     elementInfos.innerHTML = '';
128   }
129   if (response.length) { // On ne modifie les resultats que si on en a obtenu
130     if(response == "NO_FOUND") { // aucune constante trouvee
131       if(xTable == 'RES_dicoLanguage'){
132         displayInfoStartSearchOldTable(keyword);
133         if(xsearchInOldTab) {
134           clicTabAllow();
135           addTab('Ancienne');
136           changeTab('Ancienne', nbTabOld);
137           getResults(keyword, false); //on lance la recherche dans l'ancienne
                table
138         }
139       } else {
140         displayInfoCstNotFound(keyword);
141         document.getElementById('addCstAfterSearch').onclick = function(){
              menuActionAddCst(null, keyword)};
142       }
143     } else {
144       elementInfoTable.innerHTML = '';
145
146       //zou, on remplit le tableau de toutes les constantes trouvees!
147       response = response.split('+++'); //chaque ligne
148       var responseLen = response.length;
149       var elementBouton;
150       ligne = tableau.insertRow(-1);
151
152       titreCol = ligne.insertCell(-1);
153       titreCol.innerHTML += "ID";
154       titreCol.className = "titreColonne";
155
156       titreCol = ligne.insertCell(-1);
157       titreCol.innerHTML += "";
158       titreCol.className = "titreColonne";
159
160       titreCol = ligne.insertCell(-1);
161       titreCol.innerHTML += "Valeur";
162       titreCol.className = "titreColonne";
163
164       titreCol = ligne.insertCell(-1);
165       titreCol.innerHTML += "";
```

```
166       titreCol.className = "titreColonne";
167
168       for (var i = 0 ; i < responseLen -1 ; ++i) {
169         dataSearch = getDataSearch(i, response); //on recupere toutes les
              donnees
170
171         nbLangAffichage = getNbLangToDisplay(dataSearch['TraduceLang']);
172         ++iterateurLanguages;
173
174         ligne = tableau.insertRow(-1);//on ajoute une ligne
175
176         /* on remplit le tableau avec toutes les cellules */
177         addCellsConstanteAndTraduce(i, ligne, responseLen, dataSearch, xTable)
              ;
178         addCellsValide(i, ligne, dataSearch, xTable);
179         addCellsTraduceValue(ligne, dataSearch, i, keyword);
180         addCellsBtn(i, ligne, dataSearch, xTable);
181
182         if(iterateurLanguages >= nbLangAffichage) {
183           iterateurLanguages = 0;
184         }
185       }
186       displayInfoCstFound(xTable, keyword);
187     }
188   }
189
190   clicTabAllow();
191 }
192
193 /**
194  * Retourne le nombre de langues qui devront etre affiche
195  * @param Les langues dans lesquels la constante est traduite
196  * @returns Le nombre de langue
197  */
198 function getNbLangToDisplay(xsLangTraduce, xiNbLangDemande) {
199   if(!xiNbLangDemande){
200     xiNbLangDemande = nbLangDemande;
201   }
202
203   return ((xsLangTraduce.length > xiNbLangDemande) ? xiNbLangDemande :
        xsLangTraduce.length);
204 }
205
206 /**
207  * Retourne toutes les donnees d'une recherche dans un tableau associatif
208  * @param xiNumberLine Le numero de la ligne
209  * @param xresponse La requete
210  * @returns {Object} Les donnees
211  */
212 function getDataSearch(xiNumberLine, xresponse) {
213   var data = new Object();
214   var values = new Object();
215
216   var allDatas = xresponse[xiNumberLine].split('---'); //chaque colonnes
217   data['Id'] = trim(allDatas[0]);
218   dataLang = allDatas[1];
219
220   /* on s'occupe des langues: Langues d'affichage et langue traduite existante
        dans la base */
221   data['TraduceLang'] = allDatas[1].split('###');
```

```
222   data['CurrentLang'] = trim((data['TraduceLang'])[0]);
223   data['TraduceLang'].shift();
224   data['TraduceLang'] = data['TraduceLang'][0].split(' ');
225   data['TraduceLang'].pop();
226
227   /* Les autres champs */
228   data['Valide'] = trim(allDatas[2]);
229   data['Type'] = trim(allDatas[3]);
230
231   values['short'] = (allDatas[5]);
232   values['normal'] = (allDatas[4]);
233   values['long'] = (allDatas[6]);
234   data['value'] = values;
235
236   return data;
237 }
238
239 /**
240  * Colorie les mots trouves dans le texte
241  * @param xsKeyword Mot a chercher
242  * @param xsSentence Phrase dans laquelle on doit chercher
243  * @returns La chane coloree
244  */
245 function colorWordFound(xsKeyword, xsSentence) {
246   var sReturn = xsSentence;
247   var aKeyword = xsKeyword.split('-');
248   var regexHighlighting;
249
250   for(var i = 0 ; i < aKeyword.length ; ++i) {
251     if(aKeyword[i].length > 2 && typeSearch == 'text') {
252       regexHighlighting = new RegExp('('+aKeyword[i]+')', "gi");
253       regexHighlighting.exec(sReturn);
254       sReturn = sReturn.replace(RegExp.$1, "<span class=\"keyword\">"+RegExp.
              $1+"</span>");
255     }
256   }
257
258   return sReturn;
259 }
```

Listing F.7. search.js

```
1  function actionAnnulerTraduction(numberLine, xinbreLineTotal) {
2    tableau = document.getElementById("tableResults"+onglet);
3    tableau.deleteRow(numberLine+2);
4    enCoursDeTraduction = false;
5
6    affichageLiensTraduction(xinbreLineTotal, true);
7  }
8
9  function actionValiderTraduction(numberLine, xsConstante, xinbreLineTotal) {
10   tableau = document.getElementById("tableResults"+onglet);
11   insertionNouvelleTraduction(xsConstante,
12   document.getElementById('language').options[document.getElementById('
        language').selectedIndex].value,
13   document.getElementById('valueTraduce').value, 'PHRA',
14   (document.getElementById('valideTraduce').checked) ? 1 : 0,
15   document.getElementById('shortvalueTraduce').value, document.getElementById(
        'longvalueTraduce').value);
16
```

```
17    tableau.deleteRow(numberLine+2);
18    enCoursDeTraduction = false;
19
20    affichageLiensTraduction(xinbreLineTotal, true);
21
22    rafraichir();
23  }
24
25  function actionVerifSizeString(xsType, element) {
26    if(!valueNotToLong(xsType, element.value)) {
27      document.getElementById('champ'+xsType+'TooLong').style.display = 'inline-
          block';
28    } else {
29      document.getElementById('champ'+xsType+'TooLong').style.display = 'none';
30    }
31  }
```

Listing F.8. actionTranslate.js

```
1   <html>
2   <head>
3       <title>MemoLanguage</title>
4       <link href="styles/global.css" rel="stylesheet" type="text/css"/>
5      <link href="styles/homeSearch.css" rel="stylesheet" type="text/css"/>
6
7      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
8
9      <script type="text/javascript" src="lib/AIRAliases.js"></script>
10     <script type="text/javascript" src="lib/systray.js"></script>
11     <script type="text/javascript" src="lib/menu.js"></script>
12     <script type="text/javascript" src="lib/variablesGlobales.js"></script>
13     <script type="text/javascript" src="lib/util.js"></script>
14     <script type="text/javascript" src="lib/actions/actionCopy.js"></script>
15     <script type="text/javascript" src="lib/actions/actionSearch.js"></script>
16     <script type="text/javascript" src="lib/actions/actionTabs.js"></script>
17     <script type="text/javascript" src="lib/actions/actionTranslate.js"></
           script>
18     <script type="text/javascript" src="lib/actions/actionUpdates.js"></script>
19
20
21     <script type="text/javascript" src="lib/addCells.js"></script>
22     <script type="text/javascript" src="lib/configuration.js"></script>
23     <script type="text/javascript" src="lib/maxLength.js"></script>
24  </head>
25  <body>
26  <!--
27    Recherche
28   -->
29  <div id="divSearch">
30    <div id="choixLangue">
31      Afficher dans les langues: <br />
32      <input type="checkbox" checked=checked onclick="
            actionChangeLanguagesResults()" id="chkSearch1"/>
33        <label for="chkSearch1">Fran ais </label><br />
34      <input type="checkbox" onclick="actionChangeLanguagesResults()"  id="
            chkSearch2"/>
35        <label for="chkSearch2">Anglais </label><br />
36      <input type="checkbox" onclick="actionChangeLanguagesResults()" id="
            chkSearch3"/>
37        <label for="chkSearch3">Espagnol</label><br />
```

```
38    </div>
39
40    <p>Chercher par :
41    <input class="typeSearch" type="radio"  name="type" value="id" id="id"
          onclick="actionChangeTypeSearch('id',true);"/>
42          <label for="id">id</label>
43
44    <input class="typeSearch" type="radio" name="type" value="text" checked="
          checked" id="text"
45                  onclick="actionChangeTypeSearch('text',true);" />
46          <label for="text">Texte</label>
47    </p>
48      <div id="optionsValides">
49        <input type="checkbox" id="estValide" onclick="actionValideSearch( );"
             checked="checked"/>
50          <label for="valide">Traduction valides</label>
51        <input type="checkbox" id="estNonValide" onclick="actionValideSearch( );
             " />
52          <label for="nonValide">Traduction non valides</label>
53      </div>
54
55      <div id="optionsText">
56        <input type="checkbox" id="wholeWord" onclick="actionChangeWholeWord( );
             "/>
57          <label for="wholeWord">Mots entiers</label>
58        <input type="checkbox" id="regex" onclick="actionChangeRegex( );"/>
59          <label for="regex">Expression reguliere</label>
60      </div>
61    <input id="search" type="text" onkeyup="actionSearch(true, false)"
          onKeyPress="if (event.keyCode == 13) actionSearch(true, true)"/><br />
62    <input id="btnSearch" type="button"
63          onclick="actionSearch(true, true)" value="Chercher !"/>
64  </div><p>
65       <span id="onglet">
66           </span>
67
68           <a href="#" class="info" onclick="actionDisplayTypeTab()"><img src="
                img/button_add.png" alt="+" id="btnPlus"/><span class="textinfo">
                Ajouter un onglet</span></a>
69            <select style="display:none" name="typeTab" id="typeTab">
70            <option value="Ancienne">Ancienne table</option>
71            <option value="Nouvelle"">Nouvelle table</option>
72           </select>
73           <input id="btnValideAddTab" type="button" style="display:none"onclick
                ="actionAddTab()" value="Ajouter" />
74         </p>
75   <div id="displayOnglets">
76   </div>
77
78    <script type="text/javascript" src="lib/traduction.js"></script>
79    <script type="text/javascript"src="lib/search.js"></script>
80     <script type="text/javascript" src="lib/tabs.js"></script>
81    <div id="footer"><span id="version">MemoLanguage version <strong><span id="
          numVersionML"></span></strong></span>
82    <span id="langueRecherche">Langue de recherche:  <strong><span id="
          langSearch"></span></strong></span>
83    </div>
84    <script type="text/javascript" src="lib/init.js"></script>
85      <script type="text/javascript">
86      addTab("Nouvelle");
```

```
87      addTab("Ancienne");
88
89      initTabs();
90      changeTab('Nouvelle', 1);
91      </script>
92  </body>
93  </html>
```

Listing F.9. Page HTML principale homeSearch.html

F.2.2.2 bin

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <application xmlns="http://ns.adobe.com/air/application/2.6">
3       <id>Outils.Translation</id>
4       <versionNumber>0.9</versionNumber>
5       <filename>MemoLanguage</filename>
6       <initialWindow>
7           <content>homeSearch.html</content>
8           <visible>true</visible>
9           <width>850</width>
10          <height>400</height>
11      <transparent>false</transparent>
12      <systemChrome>standard</systemChrome>
13      <x>400</x>
14      <y>500</y>
15      </initialWindow>
16      <icon>
17        <image16x16>img/logos/logo16.png</image16x16>
18        <image32x32>img/logos/logo32.png</image32x32>
19        <image48x48>img/logos/logo48.png</image48x48>
20        <image128x128>img/logos/logo128.png</image128x128>
21      </icon>
22  </application>
```

Listing F.10. Descripteur de fichier MemoLanguage-app.xml