Tri alphanumérique avec chiffres romains

Introduction

Ce document est une courte explication technique de la fonction de tri codée pour le projet EMAN.

Description du problème

Dans le cadre du projet EMAN, qui englobe plusieurs corpus de documents classés de différentes manières, le besoin s'est fait sentir de la possibilité de trier des contenus d'une façon qui n'est pas fournie par les fonctions de tri classiques de PHP (et à notre connaissance, aussi étrange que cela paraisse, des autres langages de programmation non plus, à l'exception peut-être de Python).

Formulé grossièrement, il s'agit de trier des contenus selon leur titre, de façon alphanumérique classique, sachant qu'ils peuvent contenir des caractères et des chaînes assez mal gérés par les fonctions de tri basiques de PHP (sort, usort, etc.) :

- des majuscules accentuées (É, À, Ç, etc.)
- des signes de ponctuation (crochets, parenthèses, accolades, etc.)
- des chiffres romains
- éventuellement des caractères exotiques provenant de langues anciennes ou peu usitées (sanscrit, hébreu, grec, etc.)

Tous ses éléments, en particulier les chiffres romains, peuvent être situés n'importe où dans la chaîne de caractère (et pas seulement au début, comme il est fréquent).

De plus, les chaînes à trier ne sont pas forcément homogènes : certaines peuvent contenir une numérotation romaine et d'autres arabe, dans la même liste.

Ce qui nous intéresse ici, ce sont les chiffres romains (CR dans la suite du texte), des solutions techniques relativement simples existant pour les autres problèmes.

Solution

Malgré nos recherches, et à notre grande surprise, aucune bibliothèque de fonction existante ne répondait à nos besoins. Nous avons donc codé une fonction ad hoc.

Comme EMAN utilise Omeka, nous disposons du framework Zend qui comporte une intéressante fonction de conversion de chiffres entre différentes notations (documentation ici), parmi d'autres fonctions de traitement numérique.

Cette fonction demanderait à être réécrite pour pouvoir être universalisée, et pourrait très probablement être optimisée. En l'état, elle remplit sa fonction au sein du projet EMAN, et le temps manque pour la rendre plus portable et plus rapide.

La solution que nous avons choisie peut être décrite décrire comme suit :

- constitution d'un tableau contenant les chaînes à trier
- détection des suites de caractères correspondant à des CR
- stockage des CR trouvés dans un tableau indexé de la même façon que le tableau original (mêmes valeurs de clefs) avec leur équivalent arabe
- conversion des CR trouvés en chiffres arabes dans le tableau initial
- tri classique sur tableau initial avec conservation des valeurs de clefs
- remplacement dans le tableau original des chaînes en notation arabe par les chaînes avec CR correspondantes

D'autres solutions ont été envisagées, mais présentaient des désavantages par rapport à celle-ci (complexité plus grande, adaptation plus difficile, bibliothèques externes supplémentaires, etc.)

La fonction

Voici les parties pertinentes du code de la fonction, commentées. En production, elle est beaucoup plus complexe, pour couvrir tous les cas d'utilisation (tris par dates, tableau normal ou tableau d'objets, paramètres du thème Omeka, etc.)

```
/*
  La fonction prend comme paramètres :

 - le tableau à trier, ici un tableau d'objets
 - le champ sur lequel trier, donc ici le nom du membre de chaque objet du tableau à utiliser pour le tri
 - l'ordre de tri, ascendant ou descendant
 - les caractères à ignorer lors du tri
*/

function eman_sort_array($array, $field, $order = "a", $toReplace = ['[', ']']) {

  // Tableau vide, rien à trier
  if (empty($array)) : return $array; endif;

  // Tableaux principal et temporaire et comptage des chaînes de CR trouvées
  $tri = $temp = [];
  $nbRomans = $maxRomanLength = 0;

  // Nous bouclons sur les éléments du tableau
  foreach ($array as $i => $item) {
    // Cette expression régulière détecte les CR. Elle exclut les caractères seuls suivis d'un point pour gérer les cas des initiales i.e : V. Hugo grâce à un « negative lookahead » (https://www.regex-tutorial.org/positive-and-negative-lookahead-assertions.php)
    preg_match("/\b(?=[MDCLXVI]+(?!\.)\b)M{0,4}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3})\b/u", $item->{$field}, $matches);
    // Si nous ne trouvons pas de CR, l'élément reste inchangé
    if ($matches) {
      // Seul le premier CR de chaque chaîne nous intéresse
      $roman = $matches[0];
      // Conversion d'un CR en notation arabe grâce à la classe Zend dédiée
      $number = @new Zend_Measure_Number($roman, Zend_Measure_Number::ROMAN);
      @$number->convertTo(Zend_Measure_Number::DECIMAL);
      $decimal = @$number->getValue();
      // Copie des chaînes à remplacer passées en paramètre
      $replacement = $toReplace;
      // Ajout du CR aux chaînes à remplacer
      array_unshift($replacement, $roman);
      // Création d'un tableau temporaire ne contenant que des éléments vides
      $tmp = array_fill(0, count($replacement) -1, '');
      // Ajout du chiffre arabe à ce tableau
      array_unshift($tmp, $decimal);
```

```
      // Stockage dans notre tableau temporaire de la chaîne sans les caractères à exclure et avec les CR convertis en notation arabe
      $tri[$i] = str_replace($replacement, $tmp, strip_tags(str_replace(' ', '', $item->{$field})));
      // Stockage de la longueur maximale trouvée dans l'ensemble du tableau, pour analyse ultérieure
      $maxRomanLength = strlen($roman) > $maxRomanLength ? strlen($roman) : $maxRomanLength;
      // Incrément du nombre de CR trouvés dans l'ensemble du tableau d'objets
      $nbRomans++;
    } else {
      // Si pas de CR dans la chaîne, nous ne remplaçons que les caractères à exclure passés en paramètres
      $replacement = array_fill(0, count($toReplace), '');
      $tri[$i] = str_replace($toReplace, $replacement, strip_tags(str_replace(' ', '', $item->{$field})));
    }
    // Stockage dans notre second tableau temporaire de la chaîne avec les caractères à exclure passés en paramètres supprimés
    $temp[$i] = str_replace($toReplace, $replacement, strip_tags(str_replace(' ', '', $item->{$field})));
  }
  // Si le nombre de CR trouvés ET la longueur maximale sont supérieurs à 1,
  // nous trions en tenant compte des CR, donc le premier tableau temporaire,
  // sinon nous trions par ordre alphabétique normal, donc le second tableau temporaire
  if (! ($nbRomans > 1 && $maxRomanLength > 1)) {
    $tri = $temp;
  }
  // Nous travaillons en français et en Unicode
  setlocale(LC_COLLATE, 'fr_FR.UTF-8');
  // Création d'un objet "trieur", aussi en français (https://www.php.net/manual/fr/class.collator.php)
  $c = new Collator('fr_FR');
  // ... avec les paramètres adaptés
  $c->setAttribute(Collator::NUMERIC_COLLATION, Collator::ON);
  $c->setAttribute(Collator::FRENCH_COLLATION, Collator::ON);
  // Nous trions en conservant les valeurs de clefs, et en passant le champ de tri, l'ordre de tri et l'objet trieur à la fonction de tri.
  // Ces variables doivent être disponibles pour évaluer les deux chaînes à chaque itération
  uasort($tri, function($a, $b)  use ($field, $order, $c) {
                // Nos tableaux sont bien formés, nous n'avons qu'à comparer les deux éléments grâce au "trieur" PHP
                    $order == 'd' ? $x = $c->compare($b, $a) : $x = $c->compare($a, $b);
                    return $x;
                    });
  // Le tableau est trié, nous pouvons remplacer les chaînes en notation arabe par celles avec les CR (i.e les chaînes originales)
  foreach ($tri as $i => $value) {
    $tri[$i] = $array[$i];
  }
  // Le tri est terminé, nous renvoyons le tableau trié comme résultat
  return $tri;
}
```