Version 1.0


# ***Documentation du Générateur de données Universelles***

# Révision du document

| Auteurs | Note | Date |
|---|---|---|
| Brice Harismendy<br>Geoffroy Berry<br>Corentin Couvry | Création de la documentation | 27/03/2017 |

# Table des Matières

# 1 Le Générateur de Données Universelles

## 1.1 Le programme principale

### 1.1.1 Principe générale du générateur

Le programme principale du générateur est exécuté en entrant la ligne suivante :

php generer.php <Fichier de paramétrage> <lieu de sortie des document (optionnel)>

Generer.php est le coeur du générateur, il est utilisé par l'interface et peu donc être utilisé en ligne de commande.

### 1.1.2 Déroulement du programme de génération

Tout d'abord le programme via une suite de « require('') ; » appelle toutes les bibliothèques qui vont servir tout a long du programme .

Ensuite Les variables importantes sont initialisé comme le lieu de sortie des fichiers … Une étape important ce produit ensuite, la validation du fichier de paramétrage via Parametre.xsd.

Si l'étape précédente a été validée, le programme commence la génération des données en créant le fichier de rapport puis en lisant le fichier section <Table> par section <Table>. Pour chaque <Table></Table> du fichier de paramétrage on initialise le nombre de ligne à générer puis le nom de la table, ainsi que la Seed de génération si elle est présente. Tout ceci via la section <Parametre></Parametre> de <Table></Table>.

Ensuite le générateur créer une seed puis la place dans le rapport et regarde le nombre de ligne à générer, si ce nombre est strictement supérieur à 100 000 la génération est divisé en plusieurs passage pour éviter qu'il y ait trop de données en mémoire et soulager le programme (si par exemple l'on demandait 100 000 000 000 d'âge sans un système comme celui-ci l'OS aurait tué le générateur car il consommerait trop de ressources).

Enfin la génération peu débuter, le programmer lit ligne par ligne la section <Champs></Champs> de <Table></Table> et applique des traitements pour chaque type (Numérique, Dictionnaire, ou, Formule), une fois les données générées (maximum 100 000) elles sont écrites dans le fichier en fonction des formats de sortie souhaités, pour finir les données sont traitées pour préparer les variables qui survireront au rapport quand la génération est finie, l'on écrit le rapport puis on indique le temps qu'a durée la génération.

## 1.2 Le fichier de configuration XML

Le fichier est encadré par les balises <Generateur></Generateur>.

Pour chaque nouvelle table on utilise la balise <Table></Table>.
Pour chaque table il y a 2 partie pour les configurer :

- La déclaration des champs

- La définition des réglages

### 1.2.1 La déclaration des champs

Pour chaque table il faut déclarer les champs entre les balises <Champs></Champs>
entre ces balises nous déterminons les données grâce à la balise <Donnee />
la configuration se fait via les attributs :

-“Type” qui prend en valeur “Dictionnaire”, “Numerique”, “Formule” ou “IMC” (ce dernier est temporaire et permet de calculer l’IMC et sera enlevé lorsque le type formule sera plus poussé)

- “NomColonne” qui prend comme valeur le nom de la donnée que l’on veut générer (“Noms”, “Prenoms”, “Age”,....)

- “NomPerso” prend comme valeur un nom choisi par l’utilisateur.

- “ModeGeneration” prend pour valeur “unique”, “uniforme”(aléatoire), “Normale” ou “Binomiale”. Attention, pour le moment le mode de génération de toutes les données sont généré de manière aléatoire.

- “Null” prend une valeur en pourcent ou en entier et permet de remplacer un certain nombre de donnée par “Null” (ex : “6%”, “60”).

--pour les numériques il y a 4 autre attribut :

- Tout d’abord il y a “Min” et “Max” qui permettent de donner des bornes au nombre généré. De plus si on veut générer des nombres à virgule on utilise l’attribut “NbDecimale”

- Ensuite il y a l'attribut “codage” qui sert à créer une colonne à partir de la colonne générer elle se comporte comme suit : “valeur;donneRemplacement”.

--Pour le type Formule :

-Un attribut “Formule” permet de mettre une formule de la forme “ax+b”

### 1.2.2 la définition des réglages

Pour définir les réglages il y a 4 balises :

-<Seed/> qui prend pour attribut “valeur” qui lui contient une seed de génération si cette balise est absente une seed est généré, exemple :
<Seed valeur="123" />.

-<Sortie /> qui contient les formats de sortie (CSV/SQL/XML) si on veut par exemple avoir une sortie au format CSV on met : “CSV="True"”,exemple :
<Sortie CSV="True" SQL="True"/>.

-<Nbligne/> qui a un attribut “valeur” qui contient le nombre de ligne que l’on souhaite pour cette table, exemple <Nbligne valeur="1000000"/>.

-<NomTable/> qui a pour attribut “nom” qui permet de nommer la table, exemple : <NomTable nom="test2"/> .

## 1.3 Le fichier de validation xsd

Le fichier Parametre.xsd est localisé par défaut dans le même répertoire que le fichier generer.php qui l’appelle pour valider le fichier fournit en paramètre, si il est considéré comme invalide le générateur s’arrête sinon, cela permet de gérer une grande partie des erreurs.

## 1.4 Les librairies

Toutes les librairies du générateur sont contenues dans le dossier « Librairie », les librairies présente actuellement sont :

- codage.php
- FoncEcrireCsv.php
- FoncEcrireRapport.php
- FoncEcrireSql.php
- FoncEcrireXml.php
- genererDico.php
- genererFormule.php
- genererNumerique.php
- imc.php
- LesNulls.php
- make_seed.php
- MDGAleatoire.php
- TestNull.php

Elles sont appelées au lancement de generer.php avec la syntaxe suivante :

```
require('Librairie/<nom de la librairie>');
```

## 1.5 Les Dictionnaires

Les dictionnaires sont des données cohérentes qui sont stockées dans des fichiers texte à raison d'une donnée par ligne. Attention il doivent respecter une règle de nommage, en effet leur nom doit être Dico_<lenom>.txt tout en minuscule, voici une ligne de fichier de paramétrage qui appelle un dictionnaire :

<Donnee Type="Dictionnaire" NomColonne="noms" NomDictionnaire = "noms" ModeGeneration="uniforme"/>

ici on appelle le dictionnaire nommé « Dico_noms » dans le dossier Dictionnaire.

Pour les dictionnaires contenant des données cohérentes la structuration des données est faite de cette manière : data  discriminant

Actuellement seul prénoms et villes sont gérés pour demander la gestion des dépendances il faut ajouter l'attribut GenererDependance="True"

## 1.6 Les types de données

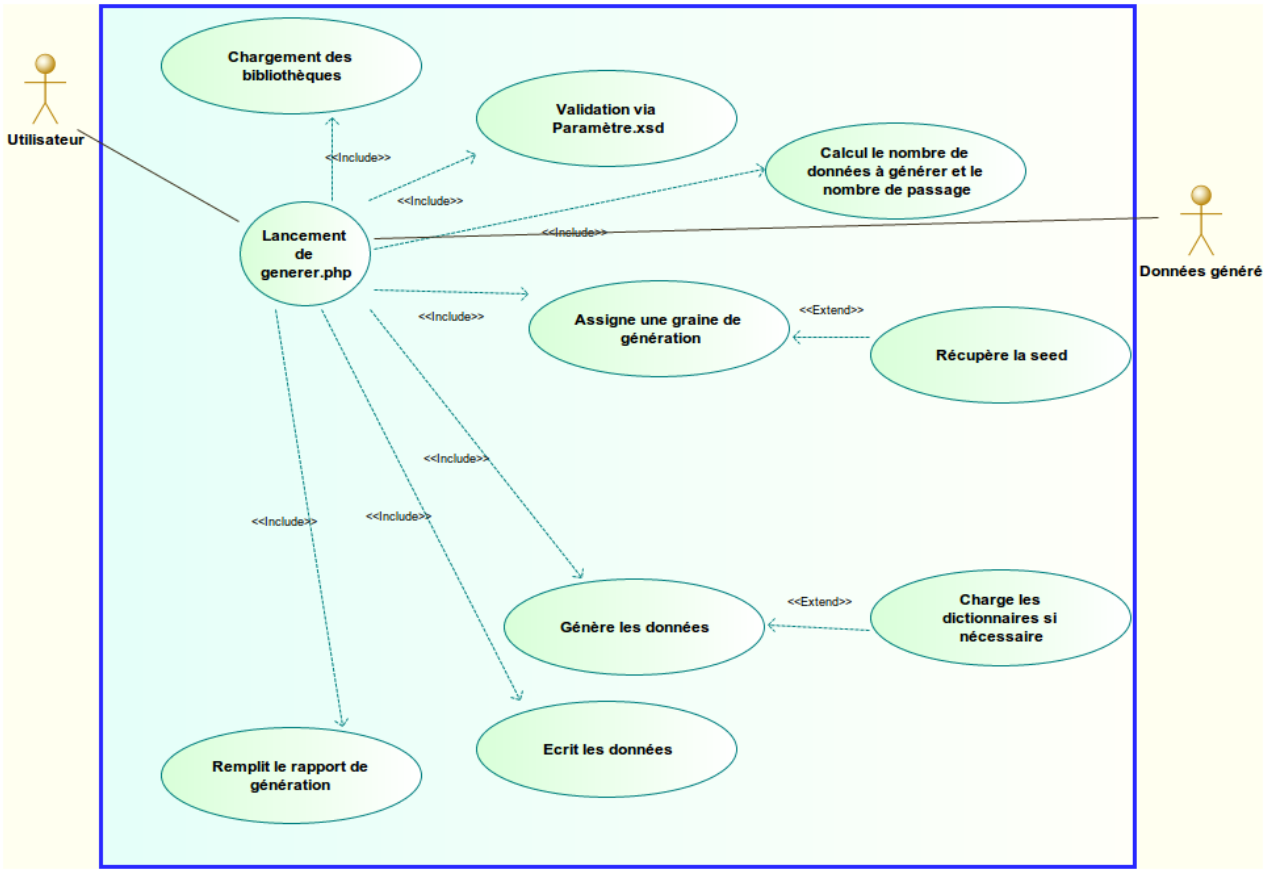
Les types de données sont parcourus dans generer.php via un switch qui parcour le fichier de paramétrage en cherchant l'attribut « Type » dans la section <Champs></Champs> , pour chaque cas de ce switch on applique des traitement a la ligne <Donnee /> en fonction des besoins, par exemple pour le type « numérique » on commence par récupérer les donnése de paramétrage du type comme le minimum, le nom de la colonne … Ensuite, on demande à un sous programme (dans le dossier Librairie), pour cela on ajoute une ligne à la matrice  $donnees comme ceci :

$donnees[$lignemat] = array();

enfin on fait un array_unshift pour nommer notre colonne si c'est la première fois qu'on fait une génération. Et on enrichit la matrice qui servira pour faire le rapport de génération.

## 1.7 Architecture du Générateur

Voici une image qui présente le fonctionnement du générateur :




# 2 L'interface du générateur

## 2.1 Fonctionnement générale

L'interface utilisateur vient s'ajouter au générateur de données et a été faite pour un serveur LAMP (Linux Apache Mysql Php) en effet l'interface utilise le répertoire /tmp du système comme point de sortie du générateur avant que l'utilisateur ne les télécharges.

L'interface propose un formulaire a taille dynamique et valide les données entrées dedans et permet à l'utilisateur d'ajouter autant de colonne/table qu'il souhaite. Et avant le lancement de la génération l'interface vérifie que les données indispensable à cette dernière sont remplies.

Enfin on peut importer un fichier de création de table SQL ou un fichier de configuration XML qui remplira le formulaire en conséquence.

La génération n'est lancé que quand l'utilisateur clique sur générer, des exemples sont présent en bas de page.

L’interface utilise le javascript pour la gestion des erreurs et l’ajout/suppression de champs dans l’interface, de l’Ajax pour les scripts d’import de fichier, du php pour le traitement des données, et Bootstrap pour l’interface et ce en respectant la norme xhtml.

Tout les scripts de l’interfaces sont dans le dossier « scripts », les feuilles de style sont dans le dossier « Style » et les images dans le dossier « ressource »

## 2.2 Les exemples

Les exemples présent dans l’interface sont codé de manière séquentielle dans le fichier exemple.js chaque exemple est codé de la même manière. On commence par remettre à zéro le formulaire puis on donne des valeurs a certains champs et on exécute les scripts de gestion de l’interface en conséquence et ce dans une fonction isolé.

Une fois que vous avez codé votre exemple pour l’ajouter à l’interface vous devez mettre dans la balise </div ayant pour classe « panel-body » a l’intérieur du  ayant pour id « Exemples » vous devez entrer le code HTML suivant :

```
<button type="submit" id="<l’id de votre exemple>" class="btn btn-lg btn-default" onclick="Nomdelafonctiondevotreexemple();">Nom de votre exemple</button>
```

## 2.3 L’importation de document

Le bouton de chargement du document est dans la  ayant pour id « chargementDeFichier » dans cette div il y a un bouton (button) suivit par une balise input qui sert à restreindre les types de fichier sélectionnable.

Ensuite, quand un fichier est sélectionné, il est chargé et intégré au formulaire via le script importation.js qui va dialoguer avec le serveur si le fichier est un .sql ou un .txt en effet dans ce cas on appelle le script SQLversXML.php qui convertit le fichier en xml et le récupère pour l’intégrer à l’interface en utilisant recuperation.js.

## 2.4 La validation des champs

Chacun des champs de type texte du formulaire est validé via une fonction dans validation.js toutes la validation ce font via des expressions régulière adapté au champ concerné si le champ est mauvais on appelle la fonction LockGenerer() qui a pour effet de bloquer le bouton « generer » en bas de page et si le champs est bon elle appelle UnLockGenerer() qui si le bouton est bloqué va le débloquer.

Enfin avant le lancement de la génération une vérification de remplissage est effectué sur les champs nécessaire à la génération.

## 2.5 La gestion du formulaire

Le formulaire est géré de manière dynamique via les scripts de GestionForm.js. Ce fichier contient toutes les méthodes qui servent à la gestion formulaire comme les onChange des listes déroulantes, l'ajout de tables, de colonnes, ou encore la suppression de colonnes, de tables, ou d'option dans les listes déroulantes

## 2.6 Création du fichier de paramétrage

Lorsque le formulaire est valide il est envoyé au serveur via la méthode post à un script pour créer un dossier temporaire dans le /tmp du serveur puis le fichier de paramétrage est écrit dans ce dossier ensuite la génération est lancé avec en paramètre le chemin vers le fichier de paramétrage et le chemin du dossier temporaire pour le paramètre de sortie, pour finir le tout est compressé et proposé au téléchargement.

## 2.7 Architecture de l'interface

Voici une image montrant comment fonctionne l'interface :