G. Hunault
Master 1 Informatique

Angers, septembre 2014

Langages de scripts et

Frameworks de développement Web

1. Programmation RePhPePyRu

Dans le but de comparer les langages REXX, PHP, PERL, PYTHON 3+ et RUBY 1.9+ on se propose ici de réaliser la même action dans ces 5 langages. On suppose qu'on dispose au départ d'un fichier dotml.xml dont voici le contenu :

```
<?xml version="1.0" ?>

<graph id="xhtml">

   <node id="n1" label="html"  "/>
   <node id="n2" label="head"  "/>
   <node id="n3" label="body"  "/>
   <node id="n4" label="title" "/>

   <edge id="e1" from="n1" to="n2" />
   <edge id="e2" from="n1" to="n3" />
   <edge id="e3" from="n2" to="n4" />

</graph>
```

On voudrait écrire un sous-programme nommé **gr2dot** qui prend en paramètre le nom d'un fichier comme celui indiqué, qui teste si le fichier est présent et, le cas échéant, le transforme en un fichier graphe.dot comme celui de la page suivante, avec suffisamment de généralité pour s'appliquer à d'autres fichiers de graphe contenant des noeuds et des arcs.

Contenu du fichier graphe.dot

```
digraph xhtml {
       html -> head
       html -> body
       head -> title
}
```

a) Ecrire un sous-programme REXX de nom **gr2dot** qui réalise cette tâche ;

b) Ecrire un sous-programme PERL de nom **gr2dot** qui réalise cette tâche ;

c) Ecrire un sous-programme PHP de nom **gr2dot** qui réalise cette tâche ;

d) Ecrire un sous-programme PYTHON 3+ de nom **gr2dot** qui réalise cette tâche ;

e) Ecrire un sous-programme RUBY 1.9+ de nom **gr2dot** qui réalise cette tâche.

On ne demande pas d'écrire de code pour les tests unitaires.

2. Utilisation de Ruby et Rails

Vous devez aider un collègue à utiliser Ruby et Rails pour développer une petite application Web. Cette application se compose de trois panneaux de saisie. Dans le premier, on doit entrer le nom d'un graphe (une simple chaine de caractères, correspondant à "un mot"). Dans le second, on peut écrire une liste de noms correspondant à des sommets dans un graphe. Dans le troisième, on doit entrer des arcs à raisons d'un arc par ligne en utilisant la syntaxe `org -> dest` où *org* et *dest* sont des noms de sommets valides c'est-à-dire présents dans le premier panneau.

Les noms de sommets ont comme contrainte de se composer d'un bloc de 3 à 6 lettres toutes en majuscules puis d'un numéro de 4 chiffres exactement.

On admettra qu'on veut utiliser une base de données au format SqlLite nommée GRAPHES afin de stocker les informations associées aux panneaux de saisie. Cette base de données devra contenir une table GRAPHE qui contient les noms de graphes, une table SOMMETS qui contient par enregistrement le nom du graphe et le nom d'un sommet, et une table ARCS qui contient par enregistrement le nom du graphe, le nom des sommets origine et destination afin de stocker les informations associées aux panneaux de saisie.

2.1 Création

Donnez les instructions en ligne de commande et les explications nécessaires pour réaliser les manipulations suivantes :

- créer une nouvelle application nommée **GRAPHE** dont la partie interface XHTML se compose des trois panneaux de saisie indiqués ;

- créer la base de données et la remplir en ligne de commandes avec le graphe XHTML associé à la question 1. On ajoutera automatiquement un numéro de sommet et vous expliciterez le contenu des tables ;

- modifier l'application pour que les valeurs saisies dans les panneux soient stockées dans la base de données.

2.2 Vérifications

On veut maintenant bloquer la saisie si les informations ne correspondent pas aux contraintes indiquées Quel(s) fichier(s) faut-il modifier et quel(s) code(s) Ruby faut-il écrire ? Et si on voulait gérer les saisies en Javascript, que faudrait-il modifier ?

Quel(s) fichier(s) faut-il modifier et quel(s) code(s) Ruby faut-il écrire si on veut interdire la saisie d'un même nom de graphe une deuxième fois ?

Le code informatique et les explications que vous fournirez doivent permettre à votre collègue de faire les manipulations tout seul sur son poste où vous avez déjà installé tous les outils logiciels comme rvm, ruby, rails, rspec...

Il vaut mieux en écrire plus et bien détailler une réponse que de vouloir traiter toutes les questions, les énoncés étant volontairement concis et succincts de façon à vous obliger à inventer de vous-même ce qui manque, à choisir une solution et à la documenter.

Vous passerez sous silence tout ce qui concerne **git** et **heroku**.