IFT3150 Projet d'informatique

Guillaume Poirier-Morency p1053380

7 mai 2015

Énoncé

> Améliorer un *micro-framework* web écrit en Vala nommé Valum dans le but de pouvoir l'utiliser dans un environnement de production et réaliser un travail pratique afin d'en démontrer son efficacité.

Le projet pourra être suivi à partir de la catégorie Valum de mon blog personnel. On pourra y consulter les mises à jours hebdomadaires en anglais.

Le travail pratique sera soumis par le superviseur du projet, Dr. François Major, et pourra être en lien avec son champs de recherche.

Contexte

Le projet se nomme Valum, il s'agit d'un micro-framework web écrit en Vala, un langage de programmation moderne qui se compile en code C à l'aide de GObject, une librairie qui permet un style de programmation orienté-objet en C.

Vala est très intéressant comme langage:

- vitesse d'exécution de l'ordre du code machine
- gestion assistée de la mémoire avec comptage de référence
- écosystème GLib
  - beaucoup de librairie sont écrite sur GObject (GStreamer)
- permet d'écrire *bindings* facilement avec des librairies écrites en C
  - simple annotations sur du code Vala
  - génération des *bindings* par introspection (GObject)
- constructions modernes:
  - fermetures

  - fonctions lambdas
  - signaux
  - modèle de traitement asynchrone

Le projet a été initié il y a quatre ans par Antono Vasiljev et j'ai repris le développent du projet pendant l'hiver 2014. Le *framework* est basé sur l'approche de routage utilisée par Sinatra et Express, deux autres *micro-framework* écrits en Ruby et JavaScript respectivement.

Valum est un *micro-framework* composé d'un mécanisme de routage puissant basé sur des *callbacks* et de composantes facilitant le traitement des différents aspects du protocole HTTP. Des construction pour router à partir d'expression régulières et d'un systèmes de règles permet décrire efficacement ce processus.

VSGI permet d'écrire des programmes qui deviendront des processus actif prêt à reçevoir et traiter des requêtes HTTP. On peut donc *scaler* une application.

L'avantage de développer un *micro-framework* est que l'outil est conçu pour s'intégrer dans un écosystème de logiciels déjà existant au lieu de tout fournir (imposer...) au développeur.

Depuis l'hiver 2014, quelques nouvelles fonctionnalités et améliorations ont été apportées:

- améliorations du routage
  - paramètres typés inspiré de Flask
  - routage par callback (une fonction détermine si la route match)
  - scoping des expressions régulières
  - extraction des groupes nommés d'une expression régulière
- début de la spécification VSGI
  - abstraction pour les requêtes, réponses, applications et serveurs
- FastCGI à l'aide du début de spécification de VSGI
  - testé sous lighttpd
- intégration de `GLib.Application` dans l'implémentation des serveurs
- waf, un système de build en Python
- meilleure intégration de CTPL, un moteur de *templating* simpliste
- intégration continue avec Travis CI
- début de documentation valum-framework.readthedocs.org

Travail à réaliser

L'objectif est de le rendre stable et il y a beaucoup de travail à faire pour atteindre un *release*:

- VSGI, un *middleware* qui permet de communiquer avec différents protocoles de communication existants entre un serveur HTTP et une application web.
  - terminer la spécification et la distribuer indépendamment
  - implanter CGI, un protocol commun
  - implanter SCGI, un protocol de communication simple basé sur les *streams*
  - extraire dans une librairie distribuée indépendamment afin de permettre sa réutilisation pour contruire d'autres *frameworks* ou applications web
- Mustache, un moteur de templating puissant conçu initialement pour Ruby
  - implanter de manière indépendante afin de pouvoir s'ajouter à l'écosystème GLib
  - analyse lexicale et syntaxique et compilation *on the fly*
- tests unitaire et mesure de qualité
  - couverture des tests avec gcov
  - analyse de performance par banc d'essai
- publication des résultats
  - consultation des communautés (reddit, *mailing lists*, ...)
  - lancement d'une version stable

La spécification fonctionnelle complète sera rédigée accompagnée d'un plan de développement et publié sur mon blog dans la catégorie Valum.

Réaliser un travail pratique avec l'outil développé. L'énoncé sera donné par le superviseur du projet et s'appliquera spécifiquement à son champs de recherche.

- calcul distribué (JSON-RPC, GSL)
- interface usager (HTML5 & CSS3)