Mathématiques en ligne avec GIAC et PHP

7 octobre 2007

1 But de ce document

GIAC (http ://www-fourier.ujf-grenoble.fr/~parisse/giac_fr.html) est une bibliothèque de calcul formel développée par BERNARD PARISSE, enseignant-chercheur à l'INSTITUT JOSEPH FOURIER de Grenoble. Habituellement, GIAC s'utilise avec le programme XCAS, mais nous allons voir comment la mettre en oeuvre pour développer une application client-serveur disponible sur internet, notamment pour des utilisations pédagogiques.

A ce document, on annexe une archive contenant les fichiers source suivants :

- phpgiac.i, phpgiag.h, phpgiac.cxx et compile
- giac_text.php et demoGiacPhp.xhtml
- giac_flash.php et moulinette

2 Architecture d'ensemble

La partie serveur sera construite à partir d'une extension pour le langage php qui permettra de soumettre des requêtes à GIAC à partir d'un script php. Le serveur doit être géré par un système de type Unix.

La partie client s'exécutera dans un navigateur, soit en Javascript, soit en Actionscript et communiquera avec le programme php.

Le scénario d'utilisation sera donc le suivant :

1. L'utilisateur charge une page au format html ou xhtml contenant un script
2. Sur le poste client, le script recueille la requête de l'utilisateur, puis l'envoie à un programme php s'exécutant sur le serveur
3. Le programme php charge l'extension giac, traite la demande de l'utilisateur et retourne la réponse
4. Le script récupère la réponse du serveur et l'affiche à l'endroit voulu dans la page html ou xhtml ou flash.

3 Extension phpgiac.so pour php

3.1 Principe d'une extension php

Une extension php permet d'étendre les possibilités de ce langage. L'instruction *dlopen(monExtension.so)* charge l'extension nommée monExtension et les fonctionnalités de celle-ci sont dés lors disponible en php.

3.2 Présentation du logiciel SWIG

L'objectif du projet SWIG (http ://www.swig.org) est de connecter des programmes écrits en C ou en C++ avec des langages de script. Ce logiciel permettra donc d'automatiser la construction d'une extension php pour GIAC.

3.3 Construction de l'extension phpgiac.so

3.3.1 Objectifs

Nous allons créer une extension php qui enrichira ce dernier des instructions suivantes :

- std : :string *giac_eval_txt(std : :string command) ;*
  permet de passer une instruction à GIAC (sous forme d'une chaîne de caractères) et retourne le résultat de l'évaluation. Par exemple, l'instruction php *$retour=giac_eval_txt(« derive(x^2) »)* donne la valeur *« 2x »* à la variable *$retour*.
- *std : :string giac_eval_mathml(std : :string command) ;*
  identique à la précédente, mais la valeur de retour est convertie au format MathML (http ://www.w3.org/Math).
- *std : :string giac_eval_cep(std : :string command)* ;
  Retourne une chaîne destinée au projet Casenpoche (http ://casenpoche.sesamath.net).
- *std : :string giac_archive_session()* et *void giac_unarchive_session(*std : :string *arch)* ;
  permettront d'utiliser le mécanisme des sessions de php afin que le serveur garde en mémoire le travail de tel ou tel utilisateur. La première instruction retourne une chaîne de caractères qui résume la session de travail avec GIAC. La seconde reconstitue l'historique de la session à partir de la chaîne de caractères précédemment retournée.

3.3.2 Code source de l'application

On supposera que Giac et Swig sont installés sur le serveur. Il faut créer d'abord un répertoire phpgiac qui contiendra les trois fichiers suivants

- phpgiac.i
- phpgiac.h
- phpgiac.cxx

Le fichier *phpgiac.i* servira à configurer le projet. En voici le contenu :

```
%module phpgiac
 %{
#include "phpgiac.h"
%}
/* Parse the header file to generate wrappers */
%include "std_string.i"
%include "phpgiac.h"
```

Le fichier *phpgiac.h* contient classiquement les entêtes :

```
 #include <string>
std : :string  giac_eval_txt(std : :string command) ;
std : :string giac_eval_mathml(std : :string command) ;
std : :string giac_eval_cep(std : :string command) ;
std : :string giac_archive_session() ;
void giac_unarchive_session(std : :string arch) ;
```

Enfin, le fichier phpgiac.cxx contient le code source en C++ :

```
#include "phpgiac.h"
#include <iostream>
#include <giac/giac.h>
#include <giac/plot.h>
#include <giac/input_lexer.h>
using namespace std ;
using namespace giac ;
std : :string giac_eval_txt(std : :string command){
  giac : :child_id=1 ;
  gen g(command) ;
```

```
  g=eval(g);
  return g.print();
}
std : :string giac_eval_mathml(std : :string command){
  giac : :child_id=1;
  gen g(command);
  g=eval(g);
  return gen2mathml(g);
}
string spread2cep(const matrice & m){
  string s="";
  int l=m.size();
  if (!l)
   return s;
  int c=m.front()._VECTptr->size();
  string lemarqueur="‘";
  for (int i=0;i<l;++i){
    for (int j=0;j<c;++j){
      s+=m[i][j][1].print()+lemarqueur+m[i][j][2].print()+lemarqueur;
    }
  }
  return s;
}

// interface pour CasEnPoche
std : :string giac_eval_cep(std : :string command){
   giac : :child_id=1;
  gen g(command);
  g=eval(g);
  if (g.type==_VECT && g.subtype==_SPREAD__VECT)
    return spread2cep(*g._VECTptr);
  else
    return g.print();
}

// Sessions
void elimine_char_null(string &s){
  int i=s.find('\0');
  while(i!=-1){
    s.replace(i,1," ");
   i=s.find('\0');
  }
  return;
}
std : :string giac_archive_session(){
  giac : :child_id=1;
  string s=archive_session(0);
  elimine_char_null(s);
  return s;
 }
void giac_unarchive_session(std : :string arch){
  giac : :child_id=1;
  string s(arch);
  gen g;
```

```
  unarchive_session_string(s,-1,g,0);
  return;
}
```

3.3.3 Construction de l'application

La construction se fait en trois temps :

1. On passe le code source de l'application à SWIG. A partir de là, celui ci élabore le code source plus complet de l'interface giac pour php. Cette étape se réalise par l'instruction :
   swig -php4 -c++ -noproxy phpgiac.i
2. On compile ensuite le paquet obtenu avec
   g++ 'php-config –includes' -c -o phpgiac_wrap.o phpgiac_wrap.cpp
   g++ 'php-config –includes' -c -o phpgiac.o phpgiac.cxx
   g++ -shared phpgiac.o phpgiac_wrap.cpp -o phpgiac.so 'php-config –includes' -lgmp -lgiac
   Remarque : le script « compile » fourni en annexe effectue les étapes 1 et 2.
3. Il ne reste plus qu'à copier la bibliothèque phpgiac.so obtenue dans le répertoire dédié aux extensions php. L'emplacement varie d'une distribution à l'autre. Sur mon système (Gnu/Linux Ubuntu), c'est le répertoire /usr/lib/php4/20050606/ .

4 Communiquer avec une application Javascript

On décrit ici une petite application qui permet à l'utilisateur distant de passer une instruction à GIAC, puis de visualiser le résultat au format MathML. L'application se compose d'un script php qui s'exécute sur le serveur ainsi que d'un fichier xhtml qui sera envoyé au poste client, incluant les instructions javascript .

4.1 Le script php

Code source du fichier giac_text.php :

```
<?php session_start();
// neutralise le cache
header("Expires :Sat,1 Jan 2000 00 :00 :00 GMT");
header("Cache-control : no-store, no-cache, must-revalidate");
header("Cache-Control : post-check=0, pre-check=0",false);
//on avertit le navigateur qu'on lui envoie du html
header("Content-Type :text/html; charset=iso-8859-1");
// Effacement éventuel de l'archive
if ($_POST["efface"]=="on")
  $archive="";
else
  $archive=$_SESSION["archive"];
// limite le temps de calcul à 1 seconde sur le serveur
set_time_limit(1);
// chargement du module giac par php
dl("phpgiac.so");
// on récupère le travail archivé
giac_unarchive_session($archive);
// on récupère l'instruction de l'utilisateur distant
$input=$_POST["in"];
// on sollicite giac
$retour=giac_eval_mathml($input);
// mise à jour de l'archive
$archive=giac_archive_session();
```

```
$_SESSION["archive"]=$archive
// envoi du résultat au poste client
print $retour ;
; ?>
```

Le mécanisme des sessions est très important. Par exemple, si l'utilisateur distant soumet une première requête à Giac en tapant l'instruction x :=2, qui affecte la valeur 2 à la variable x, il faudra que le serveur se rappelle de cette instruction lorsque l'utilisateur soumettra un peu plus tard une nouvelle requête impliquant la variable x.

4.2 Fichier xhtml

Code source de demoGiacPhp.xhtml. Le point clé est l'utilisation de l'objet XMLHttpRequest qui permet de communiquer avec le serveur sans devoir recharger la page entière chaque fois :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<html xmlns="http ://www.w3.org/1999/xhtml" xml :lang="en">
<head>
 <title>
  demoGiacPhp.xhtml
 </title>
 <script>
  function giac(){
   //l'objet XMLHttpRequest permet de communiquer avec le serveur
   var req = new XMLHttpRequest() ;
   var url="http ://localhost/html/giac_text.php" ;
   req.open('post',url, false) ;
   req.setRequestHeader('Content-Type','application/x-www-form-urlencoded') ;
   //Récupération de l'entrée de l'utilisateur
   var id=document.getElementById("in") ;
   var s=id.value ;
   // Envoi au serveur
   req.send("in="+encodeURIComponent(s)) ;
   // réception de la réponse du serveur
   var s=req.responseText ;
   // injection de la réponse en Mathml dans la page xhtml
   id=document.getElementById("resultat") ;
   while (id.firstChild)
     id.removeChild(id.firstChild) ;
   var range = document.createRange() ;
   range.selectNodeContents(id) ;
   var fragment = range.createContextualFragment(s) ;
   id.appendChild(fragment) ;
   return ;
 }
 function effaceHisto(){
  var req = new XMLHttpRequest() ;
  var url="http ://localhost/html/giac_text.php" ;
  req.open('post',url, false) ;
  req.setRequestHeader('Content-Type','application/x-www-form-urlencoded') ;
  var id=document.getElementById("in") ;
  var s=id.value ;
  req.send("efface=on") ;
  return ;
 }
</script>
```

```
</head>
<body>
  <h1>Essai php giac en mode MathML</h1>
  <p>(Mozilla seulement)</p>
  <form action="javascript :giac()">
   Instruction : <input type='text' id='in' size='50' />
   <input type="submit" value="OK" />
   <input type="button" value="Effacer l'historique" onclick="javascript :effaceHisto()'
 </form>
 <p>
  <math mode="display" id="resultat" xmlns="http ://www.w3.org/1998/Math/MathML">
   </math>
 </p>
</body>
</html>
```

5 Communiquer avec une application Actionscript

Ces techniques pourraient notamment être mises en oeuvre pour développer des applications pédagogiques avec le logiciel propriétaire « Adobe Flash » (http ://www.adobe.com) dans le cadre du projet « Mathenpoche » (http ://mathenpoche.sesamath.net)

5.1 Le script php

Il est légèrement différent du précédent, car il doit retourner une réponse au format UTF8. code source de giac_flash.php :

```
<?php
// mise en route du mécanisme des sessions php
session_start();
// limite le temps de calcul à 1 seconde sur le serveur
set_time_limit(1);
// chargement du module giac par php
dl("phpgiac.so");
// récupération de l'archive
$archive=$_SESSION["archive"];
giac_unarchive_session($archive);
// requête de l'utilisateur
$input=$_POST["mavariable"];
// on sollicite giac
$retour=giac_eval_txt("regrouper(".$input.")");
// on retourne le résultat
print "mavariable=".utf8_encode(urlencode($retour));
// on archive la session
$archive=giac_archive_session();
$_SESSION["archive"]=$archive;
?>
```

5.2 Le programme client en Actionscript

On donnera juste deux points clés : comment communiquer avec le serveur, puis comment afficher le résultat. On communique avec le serveur grâce à un objet de type LoadVars

```
var envoi = new LoadVars();
var reception= new LoadVars();
```

```
// récupération de l'entrée de l'utilisateur supposée être das le champ _root.input1
envoi.mavariable =_root.input1.text ;
// envoi au serveur et réception de la réponse
var sl=envoi.sendAndLoad("giac_flash.php", reception, "POST") ;
```

Les données retournées seront disponibles lorsque l'objet reception émettra le signal « onLoad » qui activera les instructions suivantes

```
reception.onLoad = function() {
var s=ReceptionLoadVars.mavariable ;
// la variable s contient maintenant la chaîne de caractères retournée par le serveur
// Il ne reste plus qu'à l'afficher quelque part
}
```

Le résultat peut être affiché directement en mode texte, mais on peut souhaiter une typographie adaptée aux mathématiques. C'est possible grâce à l'application « TEXTEMATHS » (http ://emmanuel.ostenne.free.fr/textemath), mais celle-ci nécessite un format d'entrée spécifique. On pourra utiliser la classe « moulinette » fournie avec ce document, pour convertir une expression GIAC en une expression TEXTEMATHS. Voici un exemple de code

```
var mclip=_root.createEmptyMovieClip("mclip",_root.getNextHighestDepth()) ;
mclip._x=500 ;
mclip._y=150 ;
var texteMath = new TexteMath(mclip,"","£",30,0x000000,50,true) ;
var mouline=new moulinette() ;
var s=mouline.parse(reception.mavariable) ;
texteMath.formule=s ;
texteMath.ecriture() ;
```

6 Communiquer avec CasenPoche

Casenpoche (http ://casenpoche.sesamath.net) est un projet de tableur pédagogique en ligne. Dans cette section, on se propose de déléguer tous les calculs du tableur à GIAC, par un mécanisme proche du précédent.

Comme Casenpoche intègre la partie « cliente » de la communication avec GIAC, on donnera juste ici le code source du script php avec lequel il communique :

```
<?php
// limite le temps de calcul à 1 seconde sur le serveur
set_time_limit(1) ;
// chargement du module giac par php
dl("phpgiac.so") ;
// récupération de la feuille expédiée par Casenpoche
$input=$_POST["mavariable"] ;
// on sollicite giac
$retour=giac_eval_cep("regrouper(".$input.")") ;
// renvoi de la feuille calculée
print "mavariable=".utf8_encode(urlencode($retour)) ;
?>
```

7 GNU Free Documentation License

Le texte complet est disponible ici :
http ://www.gnu.org/licenses/fdl.html#TOC1