Projet d'Authentification Unique

Document rédigé par
Philippe Bickham
Guy Robertson

Professeur Superviseur :
Stéphane Gagnon

Présenté à
Michal Iglewski
Professeur Coordonnateur

Dans le cadre du cours
Projet de synthèse
INF4173

Université du Québec en Outaouais
Lundi 18 Décembre 2006.

Table des matières:

Introduction

Les portails GagnonTech & IndusProd sont la propriété de l'entreprise Innovations Intracubator Inc (http://www.intracubator.com). Il est dirigé par le Dr. Stéphane Gagnon, un nouveau professeur au DSA à l'UQO qui nous a présenté ce projet.

Le projet d'authentification unique est l'idée de créer un portail web qui permet d'accéder a une multitude d'applications et de services web par l'entremise d'un seul nom d'usager et mot de passe.

De nombreuses applications commerciales et non-commerciales existent déjà pour réaliser ce genre de projet mais en grande partie ils utilisent le protocole LDAP (Lightweight directory acces protocol) qui permet de chercher de l'information sur le serveur. Parmi les applications que nous devions intégrer, la majorité ne supportaient pas le protocole LDAP. Ce qui nous motivait donc a explorer les nouvelles possibilités de connexion a distance.

Toutefois le sujet du projet tout aspect commercial écarté suscitait notre intérêt car nous n'avions pas eu jusqu'à cette session, l'opportunité de travailler sur une plateforme de réseau et de développer des applications serveur dans le cadre universitaire.

Objectif du projet

Le projet de départ était d'intégrer sous un seul domaine web 16 applications avec un seul et unique accès par la plateforme de Joomla. Toutes les applications que nous devions intégrer ainsi que la plateforme Joomla étaient des applications 'Open Source' ce qui nous permettait d'accéder a tout le code source, sans quoi nous n'aurions sans doute pas pu réaliser notre projet.

Les Langages utilisés

L'installation de xampp permettait de développer un environnement de serveur local qui comprend :

- Un serveur apache pour soit des machines Linux soit des machines Microsoft Windows
- Une base de données MySQL
- La possibilité d'exécuter du code PHP et Perl

Le logiciel Maguma Open Studio nous donnait la possibilité d'editer le code source des applications, essentiellement PHP

Les phases du projet

Nous avons divisé notre travail en 6 phases :

- **Conceptuelle/Analyse :** Évaluation des solutions open source retenues selon la grille des besoins et fonctionnalités désirées. Exploration des logiciels a intégrer pour trouver les méthodes relatives aux fonctionnalités que nous devions implémenter.
- **Architecture/ Design :** Identification des points d'intégration possibles et design d'une architecture pour la solution pouvant servir de point d'accès commun (single-sign-on) entre toutes les applications retenues. Design des processus et spécification de l'environnement de gestion intégré de ce point d'accès.
- **Développement :** Programmation de la plateforme d'intégration, des données, des profils d'usagers, des sessions, des processus et des transactions intra- et inter-applications.
- **Test :** Consultation avec les usagers/collaborateurs pour compléter les premiers tests betas.

Les deux dernières phases de rollout et maintenance n'ont pu être complétés due a la prolongation de la phase de programmation

Les applications à intégrer

Applications liées aux fonctions « conférences »:

Logiciels liées a la gestion de conférences permettant des fonctionnalités comme préparer une conférence, réserver une place, publier des articles et bibliographies, imprimer des plans de salle, faire des sondages etc.

- Commence

- Open Conf. System
- Open Journal System
- phpScheduleIT
- phpSurveyor
- Aigaion

Applications liées aux fonctions « éducation »:
Logiciels liées au domaine de l'enseignement permettant la gestion via des comptes enseignants/étudiant de cours, projets et recherches et offrant des services de Blog, forums, messagerie instantanée, dictionnaires et encyclopédies etc. Semblable au Webcity que nous connaissons déjà a l'UQO.

- elgg
- MediaWiki
- Moodle

Applications liées aux fonctions « Intranet » :
Ces Logiciels varient un peu plus dans leurs services que les précédents, ils servent a gérer des projets, relations clients - entreprises, comptes, ressources humaines, paniers d'achats électroniques et bien d'autres.

- dotProject
- syntese
- eGroupWare
- Orange HRM
- Sugar CRM
- webERP
- Zen Cart

Applications liées aux fonctions « Portail » :
Ces logiciels avaient pour rôle de gérer notre intranet, les modules nous donnaient plus de fonctionnalités nous permettant d'exploiter le domaine comme portail.

- Joomla
- Les modules comprofiler et Community Builder

Authentification

Comme nous n'étions pas les seuls à travailler sur ce projet, a notre première réunion avec le Dr. Stéphane Gagnon nous nous sommes repartis les différentes fonctionnalités que nous allions implémenter. Notre équipe s'est vue assigné les fonctions d'authentification et de gestion de session avec des cookies. Pendant que l'autre équipe s'occuperai de la gestion des usagers, ajout mise à jour etc.

Puisque les analyses des besoins et fonctionnelle de ces 2 services/portails sont presque complétées, nous offrons aux étudiants de l'UQO de participer à titre d'analystes-programmeurs à la phase d'intégration et d'implantation du système. Nous utiliserons strictement des solutions/packages open sources bien connues, roulant sur la plateforme Linux Apache MySQL PHP (LAMP), et hébergés sur des serveurs partagés virtuels.

Les objectifs de ces 2 projets étudiants sont les mêmes et peuvent être accomplis par une équipe conjointe (objectifs exécutés en parallèles à travers plusieurs itérations):

Conceptuelle/Analyse

- **(Aigaion)webinterface**

On avait pas de site de démonstration pour ce logiciel. Alors nous avons accédé le code source directement. C'est loin d'être un programme très volumineux. Nous avons trouve des le répertoire principal les scripts de login.

\3-integration\conf\webinterface

_login.php

ligne 51-52

setcookie() \\FALSE
90-91
setcookie("loginname",$R['login'],(30*24*60*60)+time());
setcookie("password",$R['password'],(30*24*60*60)+time());

_global.php

- **CommenceV1_21**

Cette application n'avait pas de démonstration non plus, et l'application était plus volumineuse que la précédente. Grace à l'index présent dans le répertoire principal nous avons pu trouver des fichiers de session dans le répertoire includes.

\3-integration\conf\CommenceV1_2a\includes

+ main_fns.php
253-check_privilege_type
311-generate_password
596-register
2633-Session
2653-add_session

+ user_authen_fns.php
6-authentication
125-login // check username and password with db
166-suLogin // check username only with db, used for su function

+ data_validation_fns.php
675-isValidPassword

\3-integration\conf\CommenceV1_2a

+ index.php
29-Session_start();

\\Apres met le username dans une variable session
31-$_SESSION["valid_user"]

50-case 1
52-view_papers
57-case 2
64-reviewer_home
69-case 3
71-admin_home

- **Comprofiler**

Comprofiler n'était pas une application en soi mais l'origine du plugin qui nous permettait de faire notre portail. Nous avons parcouru l'ajout à titre d'information. L'essentiel des modifications que nous avons apportées provenaient du module cblogin dont nous vous parlerons plus loin.

+admin.comprofiler.php
998-function editUser($uid='0', $option='users') {
1042-function saveUser($option) {
1064- if ($row->password == '') {
$pwd = mosMakePassword();
$row->password = md5($pwd);

1848-function makePass(){

+ Snoopy.class.php
70-71- var $user = ""; // user for http authentication
var $pass = ""; // password for http authentication

- **dotproject**

Site de démonstration:
http://www.dotproject.net/modules.php?op=modload&name=News&file=article&sid=7
Les portions de code pertinentes se trouvaient dans les répertoires ticketsmith et classes.

\intra\dotproject\includes

+ session.php
161-201 function dpSessionStart
192 session_set_cookie_params
193 session_start

\intra\dotproject\modules\ticketsmith

+ login.php
+ common.inc.php
492-
/* register login stuff */
//session_register("login_id");
//session_register("login_name");

\intra\dotproject\classes

+permissions.class.php
57
```
 function checkLogin($login) {
   // Simple ARO<->ACO check, no AXO's required.
   return $this->acl_check("system", "login", "user", $login);
 }
```

- **egroupware**

Site de démonstration: http://www.egroupware.org/demo
Cette application nous a donne le plus de difficultés comme c'était l'application la plus volumineuse tout groupe confondu. Et nous n'avons pas trouves de déclaration de fonctions spécifiques aux cookies. Il était très difficile de parcourir tous les répertoires alors nous nous sommes essentiellement tenus au principal. Le dernier commentaire fait référence a la quantité totale de fichiers de langue que nous avons trouves. Il semblerait que cette application ait été traduite dans beaucoup de langues. L'essentiel de ces fichiers peuvent donc être survoles.

3-integration\intra\egroupware

+login.php
94-function check_logoutcode($code)
117-		$GLOBALS['egw']->session->phpgw_setcookie('sessionid');
		$GLOBALS['egw']->session->phpgw_setcookie('kp3');
		$GLOBALS['egw']->session->phpgw_setcookie('domain');
246-	$GLOBALS['sessionid'] = $GLOBALS['egw']->session->create($login,$passwd,$passwd_type,'u');

+ logout.php
58-	$GLOBALS['egw']->session->phpgw_setcookie('eGW_remember');
	$GLOBALS['egw']->session->phpgw_setcookie('sessionid');
	$GLOBALS['egw']->session->phpgw_setcookie('kp3');
	$GLOBALS['egw']->session->phpgw_setcookie('domain');

\\3256 fichiers phpgw-????? essentiellement lang

- **elgg**

Site de démonstration : http://elgg.net/
L'adresse de démo est l'adresse principale. Nous avons pu tester le login sur cette page. Toutefois malgré la complexité du programme il nous a été relativement facile de trouver les fonctions génératrices de cookie dans le répertoire login.

class userlib.php

ligne 19-21
```
// Persistent login cookie DEFs
   define('AUTH_COOKIE', 'elggperm');
   define('AUTH_COOKIE_LENGTH', 31556926); // 1YR in seconds
```

\3-integration\edu\0.651\login

+ index.php
15-if (logged_on) {
27-authenticate_account
```
if ($ok) {
	$messages[] = gettext("You have been logged on.");
	define('redirect_url', $redirect_url);
	$_SESSION['messages'] = $messages;
	header("Location: " . redirect_url);
```

exit;

\3-integration\edu\0.651\auth\internal

+ lib.php
3-authenticate_user_login

- **mediawiki-1.7.1**

Site de démonstration:
http://www.opensourcecms.com/index.php?option=content&task=view&id=415
\3-integration\edu\mediawiki-1.7.1\includes
Idem que précédemment les fonctions se trouvent essentiellement dans le répertoire includes

+ user.php
ligne 642 et 660
function setupSession - function loadfromsession()
998-setPassword
1028-setNewpassword
1414-setCookies
1438-logout
1503-addToDatabase
1676-checkPassword
1722-editToken \\session
1909-getGroupPermissions
1926-getGroupName
1941-getGroupMember

+ Setup.php
ligne 129
session.auto_start

\3-integration\edu\mediawiki-1.7.1\includes

+ AuthPlugin.php
155-setPasswor
190-addUser
219-initUser

- **moodle**

  Application que nous avons réussi a intégrer.
  Site de démonstration: http://demo.moodle.org/

A la première approche de cette application nous avions en tête essentiellement les fonctions cookie, les trouver et les prendre en note. Nous ignorions alors la complexité du programme come nous n'avions pas eu encore à faire marcher quoi que ce soit. Certaines fonctions aux login pointent sur le répertoire /auth/cas, mais le gros des fonctions et le début de l'arborescence d'appels de fonctions de login se trouvent dans moodlelib.php dans le répertoire moodle/lib. Les fonctions d'encryptage de mots de passe aussi.

\3-integration\edu\moodle\login

+ confirm.php
ligne 42
set_moodle_cookie($USER->username);

+ index.php
ligne 137
set_moodle_cookie($USER->username);

\3-integration\edu\moodle\lib

+ cookies.js
    set_login_session_preferences();

\3-integration\edu\moodle\lib

+ session-test.php
9-session_start

- **OCS**

  Site de démonstration: http://pkp.sfu.ca/ocs/demo/

Cette application est parmi les moins volumineuses. On en a vite fait le tour. Les fonctionnalités de cookie sont dans le répertoire includes

\3-integration\conf\ocs

+ submit.php

42-if(!$login) { getusercookie(); }

50-51 if(validate($username,$password,$dbtable[papers],"",true)) {
setusercookie($username, $password, $save);

\3-integration\conf\ocs\include

+cookie.inc.php
31-function getusercookie() {
48-function setusercookie($username, &$password, $save = false) {
67-function deleteusercookie() {

+validate.inc.php
34-function validate($username, $password, $table, $where = "", $md5 = false) {
111-function paper_access($items, $username, $password) {

- **OJS**

Site de démonstration:
http://pkp.sfu.ca/ojs/demo/present/index.php/demojournal
Application de taille moyenne. Nous n'avons rien trouve sur des cookies. Il est possible que cette application ne se sert que de variables de session. Nous avons pris en note les premiers fichiers php auxquels la session faisait appel. Mais comme moodle nous a appris méfiez vous des premières apparences.

\ojs-2.1.1\classes\session

+ Session.inc.php

+ SessionDAO.inc.php

+ SessionManager.inc.php

\ojs-2.1.1\classes\user

+ User.inc.php

+ UserDAO.inc.php

+ UserSettingsDAO.inc.php

\ojs-2.1.1\classes\core

+ Request.inc.php
458 - getUserVar

511 - cleanUserVar

- **orangehrm**

Site de démonstration: http://www.orangehrm.com
Encore une petite application. L'essentiel se trouve dans le répertoire principal.

\intra\orangehrmb

+index.php
27- session_start
82 - setcookie

+login.php
22-session_start
32-$login = new Login();

\intra\orangehrm/lib/models/eimadmin/Login.php

29-function Login()

- **phpScheduleIt**

Site de démonstration: http://www.php.brickhost.com/demo.php
Encore une très petite application. Les fonctions sont a première vue mieux cibles que dans les autres. Probablement parmi les plus faciles à intégrer.

\phpScheduleIt\lib

+ Auth.class.php
82 - doLogin

index.php fait reference a ce fichier

\phpScheduleIt

+ index.php
29 - doLogin (avec un utilisateur)
32 - doLogin (avec seulement le cookie) <-- important ?

setcookie == fonction predefinie en PHP 3 PHP 4

- **phpsurveyor**

Site de démonstration:
http://www.phpsurveyor.org/index.php?option=com_content&task=view&id=15&Itemid=29
Idem que précédemment. Leur équivalent de login cette fois est la fonction de session.

\3-integration\conf\phpsurveyor-1_00\admin
+ admin.php

ligne 45 et 46 returnglobal \\user & pass

\3-integration\conf\phpsurveyor-1_00

+ common.php
ligne 935
returnglobal()

\3-integration\conf\phpsurveyor-1_00\admin
+ sessioncontrol.php
40-session_start

- **SugarSuite**

Site de demonstration:http://www.sugarcrm.com/crm/demo/sugar-suite.html
Application assez extensive. Beaucoup de fonctions relatives au login, mais assez facile a parcourir malgré tout. Attention au javascript.

\3-integration\intra\SugarSuite-Full-4.5.0b

+ acceptDecline.php
38-session_start();
45- session_destroy();
sugar_cleanup();

\3-integration\intra\SugarSuite-Full-4.5.0b\include

+ logging.php
55- class SimpleLog
{

function SimpleLog()

\3-integration\intra\SugarSuite-Full-4.5.0b\include\javascript

+ cookie.js
25-function Get_Cookie(name) {
40-function Set_Cookie(name, value, expires, path, domain, secure)
65-function Delete_Cookie(name,path,domain) {
77-function get_sub_cookies(cookie){
95-function subs_to_cookie(cookies){

\3-integration\intra\SugarSuite-Full-4.5.0b\modules\Users
+ User.php
339-function authenticate_user($password) {
388-function load_user($user_password) {

\3-integration\intra\SugarSuite-Full-4.5.0b\modules\Users\authentication
+ AuthenticationController.php
75-function login($username, $password, $PARAMS = array ()) {
116- function sessionAuthenticate() { \\good one just no params

\3-integration\intra\SugarSuite-Full-4.5.0b\modules\Users\authentication\SugarAuthenticate
+ SugarAuthenticate.php
56-function loginAuthenticate($username, $password){
88-function postLoginAuthenticate(){
159-function sessionAuthenticate(){

+ SugarAuthenticateUser.php
41-function authenticateUser($name, $password) {

\3-integration\intra\SugarSuite-Full-4.5.0b\modules\Users\authentication\LDAPAuthenticate
\3-integration\intra\SugarSuite-Full-4.5.0b\modules\Users\authentication\EmailAuthenticate

- **webERP**

Site de démonstration: http://www.weberp.org/weberp
Le site de démonstration est très simple. Et l'application très claire. Définitivement parmi les plus faciles. Comprend des fonctionnalités de crypt. L'essentiel est dans le répertoire includes.

\intra\webERP\includes

+ Login.php

+ session.inc
14-session_start();
231-function CryptPass($Password) {

- **zen-cart**

Site de démonstration:
http://www.opensourcecms.com/index.php?option=com_content&task=view&id=300
Le site de démonstration est bien. Vous remarquerez que l'application est liée a d'autres parmi la liste que nous avons étudié comme egroupeware. Étant donne ca nature nous avons probablement manques certaines fonctionnalités relatives au user.

\3-integration\intra\zen-cart-v1.3.5-full-fileset-09042006\includes\functions
+ sessions.php
26-function _sess_open($save_path, $session_name) {

\\followed by all session functions

\3-integration\intra\zen-cart-v1.3.5-full-fileset-09042006\includes\modules\pages\login
+ jscript_form_check.php
function check_password(field_name_1, field_name_2, field_size, message_1, message_2) {

Développement

Durant cette phase nous avions la tâche, de construire une base de données qui contiendrait toute l'information nécessaire pour la réalisation du projet. Monsieur Stéphane Gagnon nous a proposé 4 types d'architectures que nous devions analyser pour savoir laquelle serait la meilleure solution.

Voici les 4 solutions proposées :

- **Objets Externes** :

  Consiste a inclure les fonctions des différentes applications à l'intérieur de notre portail(Joomla).

- **SQL** :

  Nous prenons les différents scripts SQL des applications et les intégrons dans Joomla. Ceci serait très bénéfique pour l'ajout, la suppression et aussi la modification des usagers du portail.

- **Nouvelle objet** :

  Cette solution serait de passer de façon itérative au travers de chaque application, et d'inclure les scripts nécessaires pour faire fonctionner application.

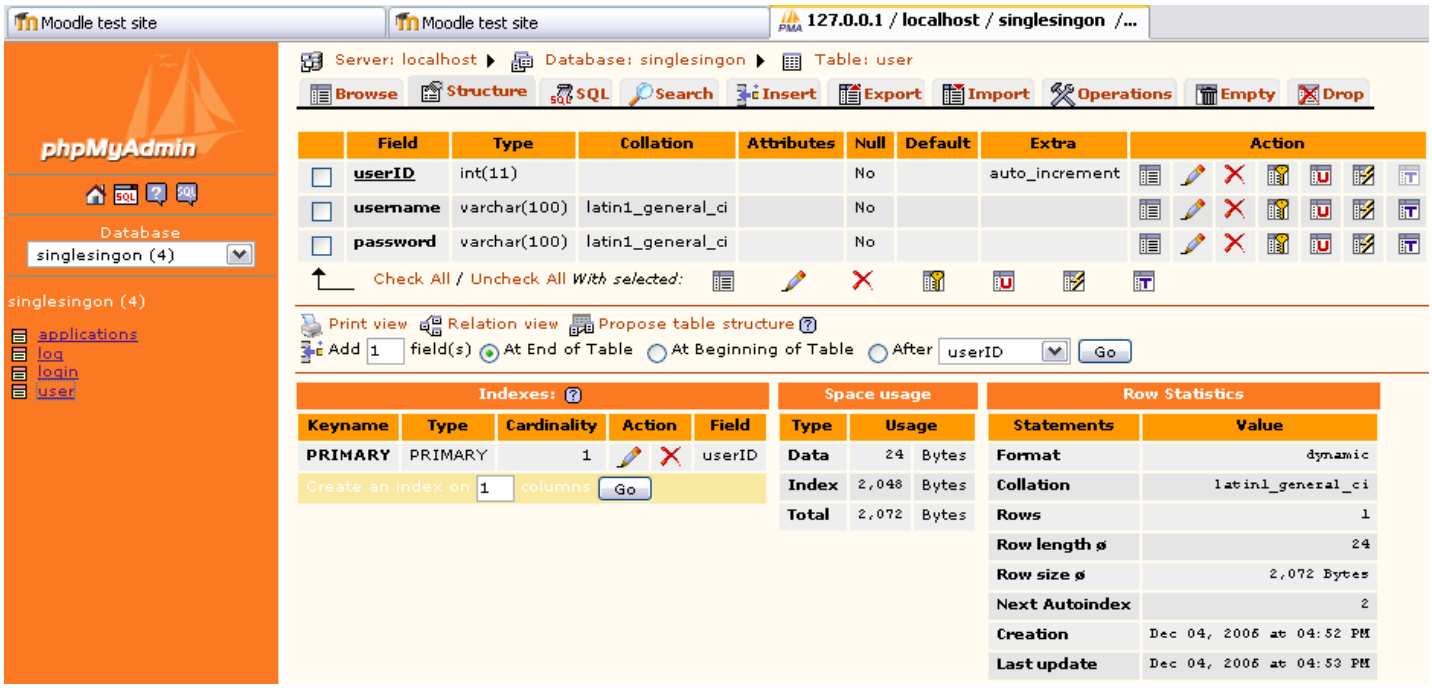
- **Composant** :

  Inclure toutes les applications à intégrer dans Joomla, qui permettra d'inclure toutes leurs fonctions dans ce portail.

Base de données :

La réalisation de ce projet nous a amené à construire trois tables qui seront nécessaire au bon fonctionnement du projet.

- **Table des usagers**, qui va comprendre tout les usagers de Joomla, ce qui nous a facilité la tache lors de la création de nos différentes fonctions.

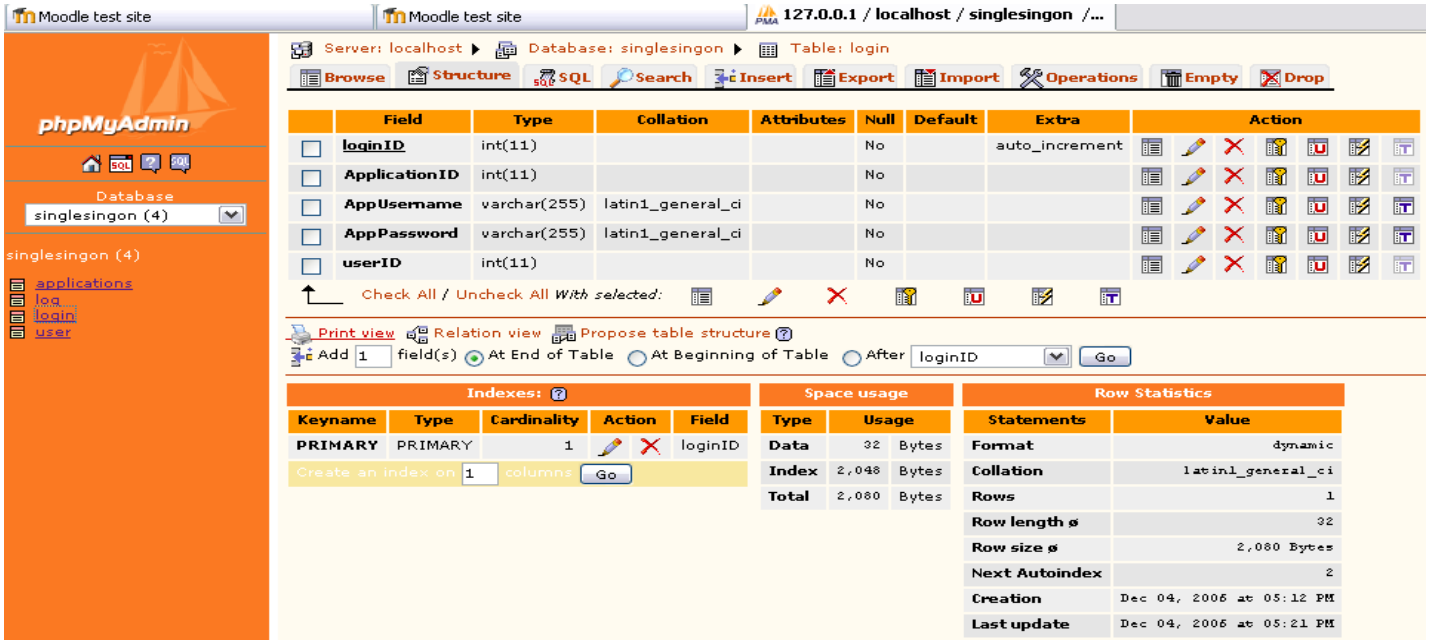


- Nous avons champ « UserID » qui va être u numéros automatique, mais aussi la clé de ces champs

- Champ « username », qui va nom de l'utilisateur dans Joomla.

- Champ « Password », qui va mot de passe de l'utilisateur dans Joomla.

2- **Table des login**, qui va regrouper toutes les informations sur l'application, sur usager et la réalisation du script pour authentification unique.



- Champ « loginID », va être numéros automatique et aussi la clé de cette table

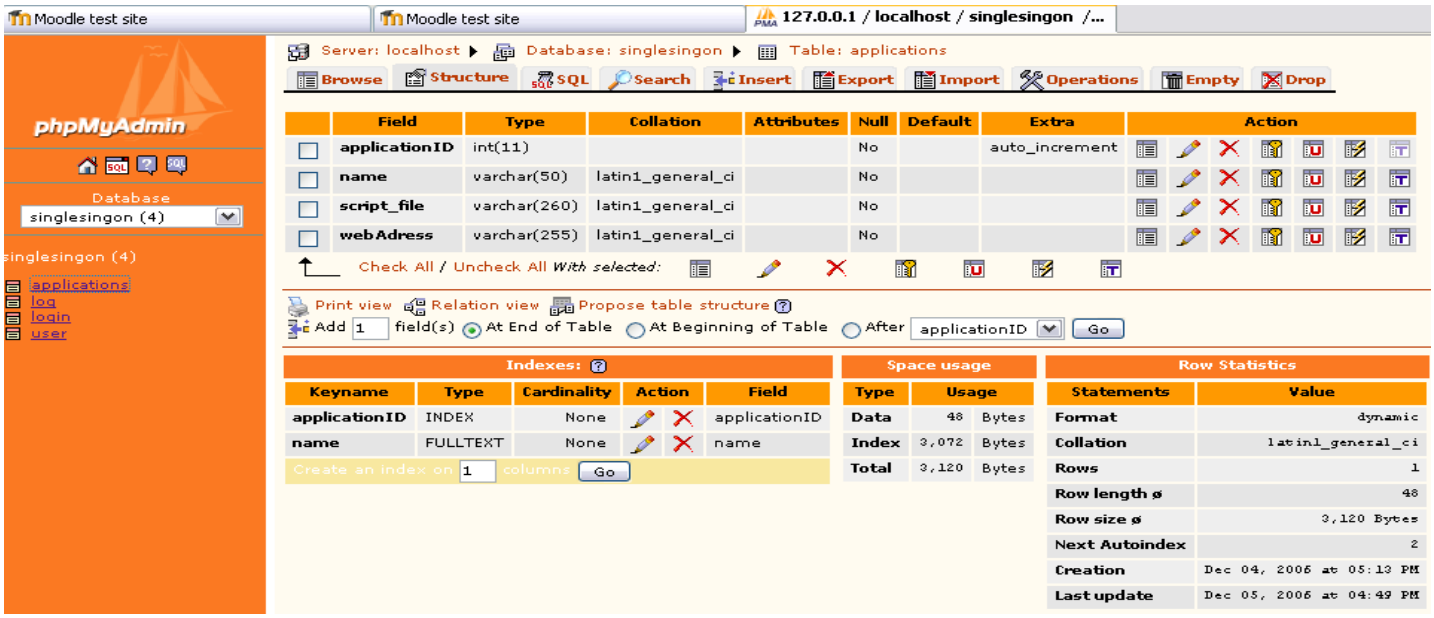
- Champ « ApplicationID », va être une clé étrangère qui va relier cette table avec la table Applications.

- Champ « AppUserName », le nom d'usager pour l'application, associer avec le bon usager Joomla.

- Champ « AppPassword », le mot de passe pour l'application, associer avec le bon usager Joomla.

- Champ « UserID », numéros identifiant utilisateur Joomla dans la base de données.

- **Table des applications,** toutes les 16 applications qui sont a intégrer dans ce projet vont se retrouver avec d'autre informations nécessaire pour le fonctionnement de nos différents scripts.



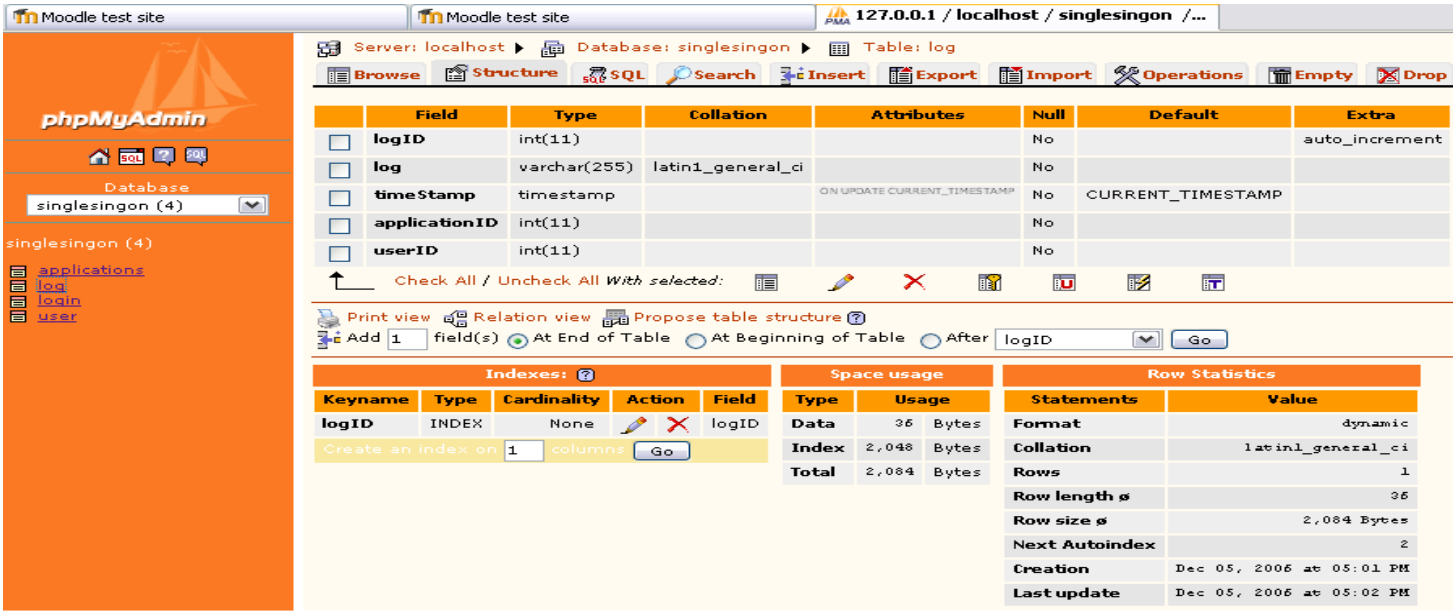
- Champs « applicationID », numéros automatiques qui servent comme clé dans cette table.

- Champs « name », va contenir le nom des applications.

- Champs « script_file », contient le nom du fichier a inclure pour par la suite appeler le bon script pour la bonne application.

- Champs « web Address », adresse Web de l'application qui va nous permette avec un click de souris d'accéder a l'application.

- **Table des logs**, cette table va réunir toute l'information sur les événements qui sont réalisé par nos scripts, par exemple le « login » ou le « logout », sur différentes applications.



- Champs « logID », un numéro automatique, qui va servir comme clé.

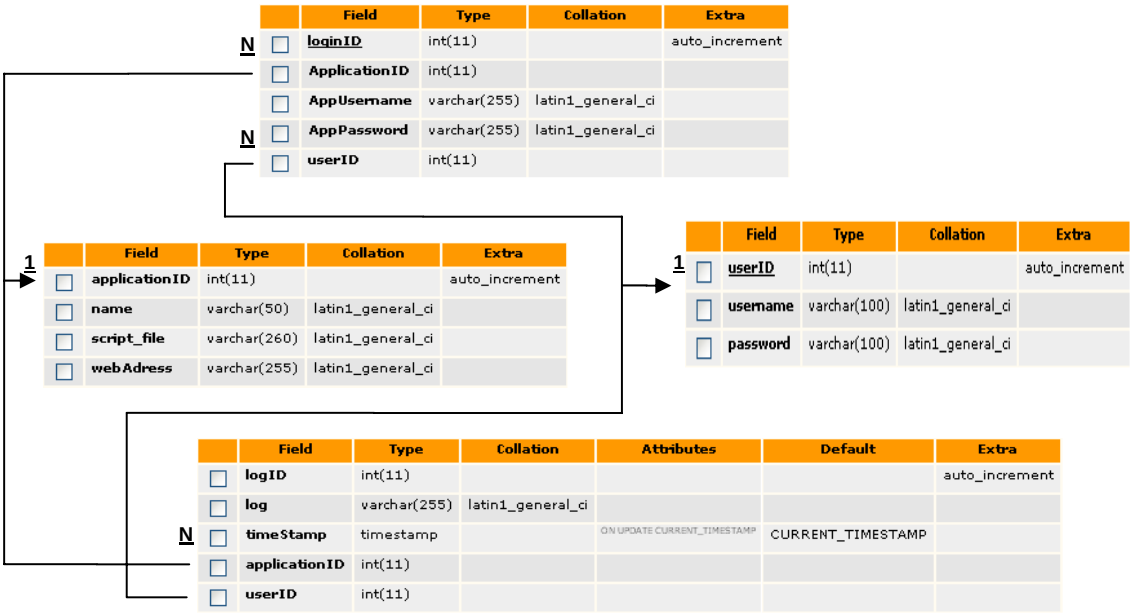
- Champs « log », information sur quel événements qui vient d'être réalisé.

- Champs « timeStamp », la date et l'heure de quand cet événement a eu lieu.

- Champs « applicationID », clé étrangère qui pointe sur application pour laquelle événement c'est produit.

- Champs «userID », clé étrangère qui pointe sur l'usager qui vient de réaliser cette événement.

Les relations entre les tables :



Notre table « login », va contenir toute l'information requis pour réaliser notre authentification à chaque application. Table « login », va pointer vers la table application par le champ « ApplicationID », cette relation est de un a plusieurs, car un enregistrement de « login peut pointer seulement sur une application. Cela va de même de pour la table « User », qui vont utiliser le champ « UserID » pour établir cette relation.

Pour ce qui est de la table « Log », ces enregistrements vont réunir toute les informations sur les événements qui vont être réalisé par nos scripts, donc une relation équivalente que celle de « login », va être nécessaire pour capter toute l'information désiré.

La réalisation des scripts :

Pour atteindre notre but, nous avons opté pour la solution numéros 3. Donc nous allons passer de façon itérative au travers de chaque application pour sortir l'information requis pour l'authentification.
Donc si l'usager a un compte dans cette application nous allons prendre le fichier des scripts inclure dans joomla et par la suite appeler la fonction que nous avons créé pour cette application.

Voici un example, pour Moodle:

```
$usernameSSO = 'root';
                        $passwordSSO = "";
                        $databaseSSO = 'singlesingon';
                        $connectionSSO =
mysql_connect(localhost,$usernameSSO,$passwordSSO);
$conn = mysql_select_db($databaseSSO);

// Script Sql pour chercher info sur usager
$querySSO = "SELECT * FROM variable INNER JOIN  applications ON
variable.variableID = applications.variableID WHERE variable.username = 'admin' and
variable.password = 'ndWBidDu'" ;

$resultSSO = mysql_query($querySSO) or die (mysql_error());
                        while ($wrow = mysql_fetch_row($resultSSO)){
                                if ($wrow[4] = 'moodle')
                                {
                                        include($wrow[8]);
                                        singlesingon();
                                }

                        }
mysql_close($connectionSSO);
```

Nous allons chercher l'information sur l'usager et nous passons de façon itérative sur chaque application, si l'usager a un compte dans cette application, nous faisons un « include », de cette page PHP, pour après appeler une fonction de cette page. Voila authentification est effectué.

Conclusion

Pour conclure, le but premier de ce projet était de réussir a intégrer 16 applications dans un portail. Donc l'usager devait rentrer seulement un nom d'usager et un mot de passe et par la suite avoir accès a chaque application, avec seulement un click de souris.

Le plus gros ennuie que nous avons encourues durant le projet était de bien comprendre le code de chaque application. Pour cette raison que monsieur Stephane Gagnon nous a demandé de nous concentrer seulement sur une application, qui est Moodle.

Les choses acquises durant ce projet sont, de toujours bien documenter son code, surtout lorsque ce code est open source, parce que nous savons jamais qui va venir travailler sur son application. C'est pour cette raison que le code doit être très lisible et structure. De plus, comment créer un environnement de travail, l'installation de serveur web, installation d'une base de données et aussi installer un intranet.

Pour ce qui est du futur du projet, la prochaine équipe devra prendre notre projet, garder la même structure de données que nous avons choisies, car nous croyons que c'est la bonne solution à prendre. Puis, tenter d'intégrer les 15 applications restantes a joomla.

Bibliographie

L'environment

- **XAMPP**: http://www.apachefriends.org/en/xampp.html

Le portail (Intranet)

- **Joomla**: http://www.joomla.org

Les applications

- **Applications liées aux conférences**
  - Commence: http://iaprcommence.sourceforge.net/
  - Open Conf. System: http://pkp.sfu.ca/?q=ocs
  - Open Journal System: http://pkp.sfu.ca/?q=ojs
  - phpScheduleIT: http://www.php.brickhost.com/
  - phpSurveyor: http://www.phpsurveyor.org/
  - Aigaion: http://sourceforge.net/projects/aigaion/

- **Applications liées a l'éducation**
  - Elgg: http://elgg.net/
  - MediaWiki: http://www.mediawiki.org
  - Moodle: http://www.moodle.org/

- **Applications liées a l'Intranet**
  - DotProject: http://www.dotproject.net/
  - eGroupWare: http://www.egroupware.org/
  - Orange HRM: http://sourceforge.net/projects/orangehrm/
  - Sugar CRM http://www.sugarcrm.com/
  - webERP: http://www.weberp.org/
  - Zen Cart http://www.zen-cart.com/

Langages utilises

- PHP:
  - W3Schools: http://www.w3schools.com/php/
  - PHP Freaks: http://www.phpfreaks.com/
  - PHP Help: http://www.phphelp.com/

- **SQL**:
  - PHP Freaks: www.phpfreaks.com/quickcode/SQL-script-import- function/661.php

Méthodes utilisées

- **RUP:**
  http://glef.uqo.ca/rup/

Liens importants pour notre recherche sur les applications

xrefMoodle: http://xref.moodle.org/nav.html?index.html

LDAP: http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol
http://www.gracion.com/server/whatldap.html
http://www.kingsmountain.com/ldapRoadmap.shtml
http://www.ldapman.org/articles/intro_to_ldap.html

SetCookie: http://ca3.php.net/setcookie

Session: http://ca.php.net/session

Cookie: http://www.w3schools.com/php/php_cookies.asp