Réseaux 1

TP 11 - Programmation Web avec PHP (2)

Objectif : utiliser les fonctions PHP d'accès à une base de données Oracle

Note : vous placerez vos fichiers PHP sur nyx, dans un répertoire `~/web/rx1/TD11`

1. Tests de connexion PHP/Oracle

Nous commençons par créer les tables de démonstration de oracle. Le schéma relationnel est le suivant :
EMP (**empno**, ename, job, mgr, hiredate, sal, comm, ***deptno***)
DEPT (**deptno**, dname, loc)
Les clés primaires de EMP et DEPT sont respectivement `empno` et `deptno`.
Le champ `deptno` dans EMP est une clé étrangère qui fait référence à la clé primaire `deptno` de DEPT.

Le script `demobld.sql` permettant de créer ces deux tables est donné en annexe1. Vous trouvez le fichier dans : `Support Cours/Réseaux1`

Créer les tables de démonstration

1.1. Lancez le client **SQL*PLUS** sur votre PC dans Démarrer/ Programmes / Oracle. Entrez votre nom d'utilisateur oracle, votre mot de passe et la chaîne hôte « IUTINFO »

1.2. Depuis SQL*PLUS, entrez la commande (on suppose que `demobld.sql` est copié dans `d:\temp`) :

```
SQL> start d:\temp\demobld.sql
```

1.3. Vérifiez les tables sous SQL*PLUS, tapez les commandes suivantes :

```
SQL> describe dept
SQL> describe emp

SQL> select * from dept;
SQL> select * from emp;
```

Test de connexion

1.4. Créez le script `test_connexion.php` permettant de se connecter à l'instance oracle « IUTINFO »

```
<?php
   $user="et";                  // changer par votre login oracle
   $pass="ora";                 // changer par votre mot de passe oracle
   $db="iutinfo.unice.fr";      // instance oracle iutinfo sur nyx

   $connexion = OCILogon($user,$pass,$db);
   if ( !$connexion ) {
        echo "Impossible de se connecter";
        exit;
   };
   echo "Connexion réussie";
?>
```

Tests de requêtes SQL

1.5. Créez le script `test_select.php` permettant de lister les noms et les commissions des employés triés par noms.

```
<?php
   require ("test_connexion.php");

   $query = "SELECT ename, comm FROM emp ORDER BY ename ";
   $stmt = OCIParse($connexion, $query);
   OCIExecute($stmt);

   echo "<pre>";
   $nbrows=0;
   while ( OCIFetchInto($stmt,&$data,OCI_ASSOC) ) {
        echo "$data[ENAME] \t $data[COMM] \n";
        $nbrows++;
   }
   echo "$nbrows lignes sélectionnées";
   echo "</pre>";
?>
```

1.6. Créez le script `test_update.php` permettant d'augmenter de 20% la commission des employés.

```
<?php
   require ("test_connexion.php");

   $query ="UPDATE emp SET comm=comm*1.2 ";
   $stmt = OCIParse($connexion, $query);
   $res = OCIExecute($stmt, OCI_COMMIT_ON_SUCCESS);

   if ($res) {
        echo OCIRowCount($stmt) . " lignes modifiées";
   } else {
        echo "<hr>Erreur fatale : opération abandonnée<hr>";
   };
?>
```

1.7. Créez le script `test_insert.php` permettant d'ajouter une ligne dans la table EMP. Testez les cas d'erreurs suivants : l'employé est déjà présent, le no de département n'existe pas.

```
<?php
   require ("test_connexion.php");

   $query = "INSERT INTO emp (empno, ename, deptno) VALUES (8000, 'TOTO', 10) ";
   $stmt = OCIParse($connexion, $query);

   if ($res) {
        echo OCIRowCount($stmt) . " ligne ajoutée";
   } else {
        echo "<hr>Erreur fatale : opération abandonnée<hr>";
   };
?>
```

2. Session utilisateur

Nous allons créer les fonctions PHP permettant de contrôler l'accès de certaines pages d'un site à des utilisateurs autorisés (login/password). On suppose que les login / password sont stockés dans une table oracle.

Le principe est simple : on place en début de chaque page sécurisée une fonction qui contrôle l'accès.
Cette fonction bloque l'accès et envoie un formulaire d'identification si l'utilisateur ne s'est pas identifié.
La page est alors rappelée par le formulaire d'identification, avec envoi du login / password.

Lorsque l'utilisateur s'est correctement identifié, on enregistre son login dans une variable de session.
On pourra alors simplement tester si l'utilisateur s'est identifié en regardant si la variable de session login est définie.

Table des utilisateurs

2.1. Créez la table oracle pour stocker les login / passwd des utilisateurs autorisés, et insérez des lignes pour les tests.

Vous regrouperez les différentes fonctions dans un module ***`util.php`***

Fonction pour envoyer le formulaire d'identification

2.2. Créez la fonction FormIdentification ($nomScript) qui envoie le formulaire d'identification. Cette fonction prend en argument le nom du script appelé par le formulaire.

Fonction pour vérifier l'identification par login / password

2.3. Créez la fonction VerifLogin ($login, $pwd) qui recherche la combinaison login/passwd dans la table des utilisateurs. Cette fonction renvoie TRUE si la combinaison est trouvée, ou sinon FALSE.

Fonction pour contrôler l'accès d'une page

2.4. Créez la fonction ControleAcces() qui est appelée au début de chaque page sécurisée.

On examine les cas suivants :

1. si la variable de session login est définie (utilisateur déjà logué), alors retour sans rien faire
2. si les données de formulaire d'identification sont reçues, et le login/password est bon, alors enregistrer le login dans une variable de session, puis retour dans le script
3. si 1. et 2. non satisfaits, alors envoyer le formulaire d'identification en passant en paramètre le nom du script de la page, puis sortie du script

Pages de test

2.5. Créez une page d'accueil sécurisée, placez dans cette page un lien vers une deuxième page sécurisée que vous créez. Testez.

3. Application à rendre

Le but de l'application est de créer une interface très simple, en prenant l'exemple de l'annuaire du personnel du département informatique.

Base de données

Définissez la table pour stocker les noms, prénoms et numéros de poste téléphonique du personnel.

- Les numéros de poste sont de la forme 99.99
- La clé primaire est composée du couple (nom, prénom)

Fonctionnalités

On souhaite pouvoir :

1. Rechercher une entrée dans la table annuaire à partir des premières lettres du nom et/ou du prénom
2. Ajouter une nouvelle entrée dans l'annuaire
3. Modifier un numéro de poste
4. Supprimer une entrée de l'annuaire

Les opérations de mises à jour sont réservées à l'administrateur.

On pourra se loguer en tant qu'administrateur avec le logon : « admin/oracle »
(impérativement afin que l'on puisse tester votre application !)

Vous utiliserez une feuille de style pour la présentation de l'ensemble des pages.

Cette application sera intégrée dans vos pages de TP.

4. Annexe 1 – Tables de démonstration

Script SQL permettant de construire les tables de démonstration (dans Support Cours / Réseaux1 / demobld.sql)

```
SET TERMOUT ON

PROMPT Building demonstration tables.  Please wait.
SET TERMOUT OFF

DROP TABLE EMP    ;
DROP TABLE DEPT   ;

CREATE TABLE DEPT
   (
   DEPTNO         NUMBER(2) PRIMARY KEY             ,
   DNAME          VARCHAR2(14)                      ,
   LOC            VARCHAR2(13)
   )    ;

CREATE TABLE EMP
   (
   EMPNO          NUMBER(4)      PRIMARY KEY        ,
   ENAME          VARCHAR2(10)                      ,
   JOB            VARCHAR2(9)                       ,
   MGR            NUMBER(4)                         ,
   HIREDATE       DATE                              ,
   SAL            NUMBER(7, 2)                      ,
   COMM           NUMBER(7, 2)                      ,
   DEPTNO         NUMBER(2) REFERENCES DEPT
   )    ;


INSERT INTO DEPT VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT VALUES (20, 'RESEARCH',   'DALLAS');
INSERT INTO DEPT VALUES (30, 'SALES',      'CHICAGO');
INSERT INTO DEPT VALUES (40, 'OPERATIONS', 'BOSTON');


INSERT INTO EMP VALUES
(7369, 'SMITH','CLERK',7902, TO_DATE('17-DEC-1980', 'DD-MON-YYYY'),800, NULL,20);
INSERT INTO EMP VALUES
(7499, 'ALLEN','SALESMAN',7698, TO_DATE('20-FEB-1981','DD-MON-YYYY'),1600, 300,30);
INSERT INTO EMP VALUES
(7521, 'WARD', 'SALESMAN',7698, TO_DATE('22-FEB-1981', 'DD-MON-YYYY'), 1250,  500, 30);
INSERT INTO EMP VALUES
(7566, 'JONES','MANAGER',7839,TO_DATE('2-APR-1981', 'DD-MON-YYYY'),2975, NULL, 20);
INSERT INTO EMP VALUES
(7654, 'MARTIN','SALESMAN',7698,TO_DATE('28-SEP-1981', 'DD-MON-YYYY'), 1250, 1400, 30);
INSERT INTO EMP VALUES
(7698, 'BLAKE', 'MANAGER',7839,TO_DATE('1-MAY-1981', 'DD-MON-YYYY'), 2850, NULL, 30);
INSERT INTO EMP VALUES
(7782, 'CLARK',  'MANAGER',7839,TO_DATE('9-JUN-1981', 'DD-MON-YYYY'), 2450, NULL, 10);
INSERT INTO EMP VALUES
(7788, 'SCOTT', 'ANALYST',7566,TO_DATE('09-DEC-1982', 'DD-MON-YYYY'), 3000, NULL, 20);
INSERT INTO EMP VALUES
(7839, 'KING','PRESIDENT', NULL,TO_DATE('17-NOV-1981', 'DD-MON-YYYY'), 5000, NULL, 10);
INSERT INTO EMP VALUES
(7844, 'TURNER','SALESMAN',7698,TO_DATE('8-SEP-1981', 'DD-MON-YYYY'), 1500, 0, 30);
INSERT INTO EMP VALUES
(7876, 'ADAMS', 'CLERK', 7788,TO_DATE('12-JAN-1983', 'DD-MON-YYYY'), 1100, NULL, 20);
INSERT INTO EMP VALUES
(7900, 'JAMES', 'CLERK', 7698,TO_DATE('3-DEC-1981', 'DD-MON-YYYY'),   950, NULL, 30);
INSERT INTO EMP VALUES
(7902, 'FORD', 'ANALYST',7566,TO_DATE('3-DEC-1981', 'DD-MON-YYYY'),  3000, NULL, 20);
INSERT INTO EMP VALUES
(7934, 'MILLER','CLERK',7782,TO_DATE('23-JAN-1982', 'DD-MON-YYYY'), 1300, NULL, 10);

COMMIT;
SET TERMOUT ON

PROMPT Demonstration table build is complete.
```

5. Annexe 2 – Fonctions OCI (*Oracle Call Interface*) de PHP

Etablit une connexion à un serveur Oracle
resource ocilogon (string **username**, string **password**, string **db**)

ocilogon() retourne un identifiant de connexion, nécessaire à la plus part des fonctions OCI. Si l'option ORACLE_SID n'est pas précisée, PHP utilisera la variable d'environnement ORACLE_SID pour déterminer le serveur de connexion.

Les connexions sont partagées, à l'intérieur d'une même page avec ocilogon(). Cela signifie que COMMIT et ROLLBACK s'appliquent à toutes les transactions commencées à l'intérieur d'une même page, même si vous avez créé de multiples connexions.

Déconnexion d'un serveur Oracle
int ocilogoff (resource **connection**)

ocilogoff() ferme la connexion Oracle.

Analyse une requête
int ociparse (ocifreedesc **conn**, strint **query**)

ociparse() analyse la requête **query** sur la connexion **conn**, et retourne TRUE si la requête **query** est valide, et FALSE, si ce n'est pas le cas. **query** peut être n'importe quelle requête SQL.

Exécute une commande
int ociexecute (resource **statement**, int **mode**)

ociexecute() éxécute une commande déjà préparée (voir ociparse()). L'option **mode** vous permet de spécifier le mode d'exécution (par défaut, il est à OCI_COMMIT_ON_SUCCESS). Si vous ne voulez pas que la commande soit automatiquement validée, utilisez le mode OCI_DEFAULT.

Retourne le nombre de lignes affectées
int ocirowcount (resource **statement**)

ocirowcount() retourne le nombre de lignes affectées par une commande de modification. Cette fonction ne vous indiquera pas le nombre de lignes retournées par un SELECT : il faut que les lignes aient été modifiées.

Retourne la valeur d'une colonne dans une ligne lue
mixed ociresult (resource **statement**, mixed **column**)

ociresult() retourne les données de la colonne **column** dans la ligne courante (voir ocifetch()). ocifetch()retournera tout les types, sauf les types abstraits (ROWIDs, LOBs et FILEs).

Modifie la prochaine ligne dans le pointeur interne de résultat.
int ocifetch (resource **statement**)

ocifetch() place la prochaine ligne (d'une commande SELECT) dans le pointeur interne de résultat.

Retourne la ligne suivante dans un tableau
int ocifetchinto (resource **stmt**, array &**result**, int **mode**)

ocifetchinto() retourne la ligne suivante (pour une commande SELECT) dans le tableau **result**. ocifetchinto() écrasera le contenu de **result**. Par défaut, **result** sera un tableau à index numérique, commençant à 1, et qui contiendra toute les colonnes qui ne sont pas NULL.

L'option **mode** vous permet de modifier le comportement par défaut de la fonction. Vous pouvez passer plusieurs modes simplement en les additionnant (i.e. OCI_ASSOC+OCI_RETURN_NULLS). Les modes valides sont :

- OCI_ASSOC Retourne un tableau associatif.
- OCI_NUM Retourne un tableau à index numérique (DEFAULT, valeur par défaut)
- OCI_RETURN_NULLS Retourne les colonnes vides.

- OCI_RETURN_LOBS Retourne la valeur des objets LOB plutôt que leur descripteur.

Retourne toutes les lignes d'un résultat
int ocifetchstatement (resource **stmt**, array &**variable**)

ocifetchstatement() retourne toutes les lignes d'un résultat dans le tableau variable. ocifetchstatement() retourne le nombre de lignes retournées.

Teste si la valeur d'une colonne est NULL
int ocicolumnisnull (resource **stmt**, mixed **column**)

ocicolumnisnull() retourne TRUE si la colonne **col** du résultat **stmt** est NULL. Vous pouvez utiliser le numéro de colonne (l'indexation des colonnes commence à 1) ou le nom de la colonne, pour le paramètre **col**.

Valide les transactions en cours
int ocicommit (resource **connection**)

ocicommit() valide toutes les transactions en cours sur la connexion Oracle **connection**.

Annule les transactions en cours
int ocirollback (resource **connection**)

ocirollback() annule les transactions en cours sur la connexion Oracle **connection**.

Libère toutes les ressources occupées par une commande.
int ocifreestatement (resource **stmt**)

ocifreestatement() retourne TRUE en cas de succès, et FALSE en cas d'échec.

Retourne la dernière erreur de stmt|conn|global.
array ocierror (int **stmt)conn**|

ocierror() retourne la dernière erreur trouvée. Si l'option **stmt|conn** n'est pas fournie, la dernière erreur rencontrée est retournée. Si aucune erreur n'est trouvée, ocierror() retourne FALSE.