M3.23.4 SI

Les composants métiers

Jean-François Berdjugin
IUT1 Grenoble, dept. SRC 2009

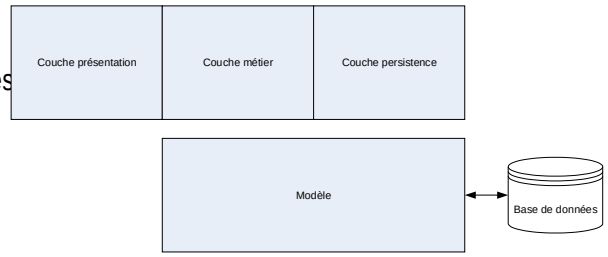
Introduction

- Nous avons des tables, nous avons des objets comment lier les deux en capitalisant notre travail : en utilisant des objets ou des composants métier.

Objets métiers : Ensemble d'objets conçus pour représenter les processus et les connaissances d'un métier en particulier.

Introduction

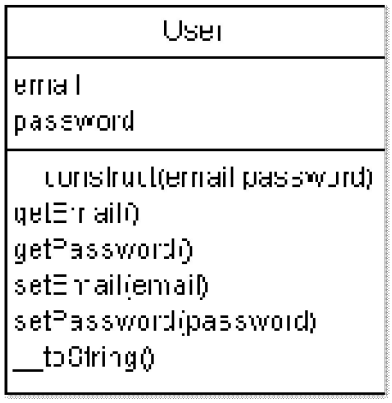
- Présentation : IHM
- Métier : cœur de l'application, lieu ou réside les objets traités par l'application
- Persistance : couche d'accès aux données. Cette couche permet une indépendance de la logique **m**étier et du stockage des données associées.



Exemple

- Nous avons un système d'information contenant des utilisateurs et des articles.
- Nous allons les représenter sous forme objets => la classe User et la classe Article.
- Nous allons fournir les moyens de leur persistance => la classe GestionBDUser, la classe GestionBDArticle.
- Nous pourrions fournir des traitements pour les utilisateurs et les articles

La classe User



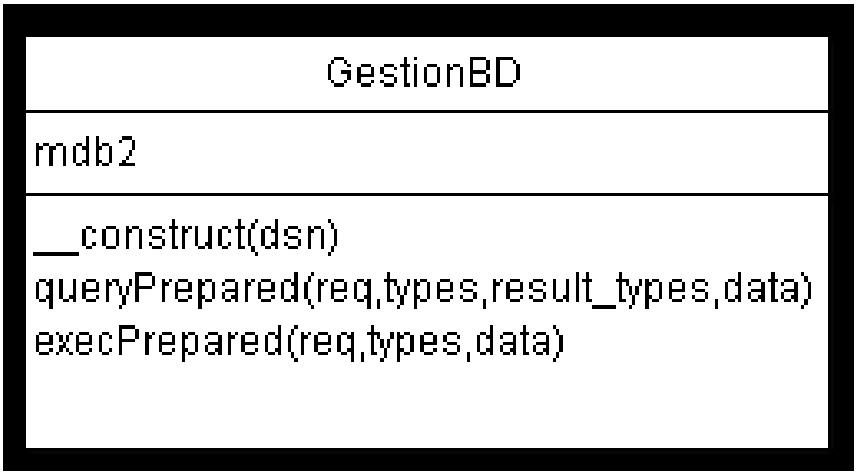
La classe User

```
class User {
private $email;
private $password;

function __construct($email, $password='')
{
$this->email = (string) $email; //un transtypage
      est utilisé pour avoir une chaine
$this->password = (string) $password;
}

function getEmail()
{
return $this->email;
}

function getPassword()
{
return $this->password;
}

function setEmail($email)
{
$this->email = (string) $email;
}

function setPassword($password)
{
$this->password = (string) $password;
}

function __toString()
{
return "email : $this->email password :
     $this->password";
}
}
```

La classe GestionBD

- Nous avons deux types de requêtes :
  - Les requêtes de sélection et
  - Les requêtes de mise à jour
- Nous souhaitons être indépendant du SGBD nous avons choisi MDB2
- Nous souhaitons aussi a moins cout pouvoir changer de couche d'abstraction

La classe GestionBD



La classe GestionBD

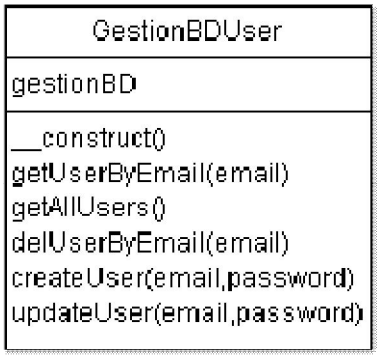
```
class GestionBD {

protected $mdb2 = null;

public function __construct($dsn) {

      $options = array(
         'debug' => 2,
         'result_buffering' => true,
      );
      $mdb2 = & MDB2::singleton($dsn,$options);

      if (MDB2::isError($mdb2)) {
         die($mdb2->getDebugInfo() & $mdb2->getMessage());
      }
      $this->mdb2 = $mdb2;
   }
```

```
public function queryPrepared($req,$types,$result_types,$data) {
$statement = $this->mdb2->prepare($req,$types, $result_types);

      $resultset = $statement->execute($data);

      if (MDB2::isError($resultset)) {
      die($resultset->getMessage());
      }
      return $resultset;
   }

public function execPrepared($req,$types,$data) {
$statement = $this->mdb2->prepare($req,$types,
         MDB2_PREPARE_MANIP);
      $affectedRows = $statement->execute($data);
      //echo $this->mdb2->last_query;
      if (MDB2::isError($affectedRows)) {
         die($affectedRows->getMessage());
      }
      return $affectedRows;
   }

}
```

La classe GestionBDUser

- GestionBD est généraliste et ne répond pas directement à nos besoins : getUserByEmail, getAllUsers, ...
- => GestionBDUser

La classe GestionBDUser



La classe GestionBDUser

```
class GestionBDUser {
private $gestionBD;

function __construct()
{
$this->gestionBD = new GestionBD(DSN);
}

function getUserByEmail($email)
{
$req= 'select email, password from User where email=:email';
$types = array('text');
$result_types = array('text', 'text');
$data = array('email' => $email);

$resultSet = $this->gestionBD->queryPrepared($req,$types,
        $result_types, $data);

$user = null;
if ($resultSet->numRows()==1) {
$row = $resultSet->fetchRow(MDB2_FETCHMODE_ASSOC);
$user =  new User($row['email'],$row['password']);
}

return $user;
}
```

```
function getAllUsers()
{
$req= 'select email, password from user';
$types=array();
$result_types = array('text', 'text');
$data = array();

$resultSet = $this->gestionBD->queryPrepared($req,$types,
        $result_types, $data);

$users = array();
$rows = $resultSet->fetchAll(MDB2_FETCHMODE_ASSOC);
foreach ($rows as $row)
{
$user =  new User($row['email'],$row['password']);
array_push($users, $user);
}

return $users;
}
```

La classe GestionBDUser

```
function delUserByEmail($email)
{
$req= 'delete from User where email=:email';
$types = array('text');
$data = array('email' => $email);
$res = $this ->gestionBD ->execPrepared($req, $types,
        $data);
return $res;
}

function createUser($email, $password)
{
$req = 'insert into User (email,password) values (:email,
        :password)';
$types = array('text','text');
$data = array('email' => $email, 'password' =>
        $password);
$res = $this ->gestionBD ->execPrepared($req, $types,
        $data);
return $res;
}
```

```
function updateUser($email, $password)
{
$req = 'update User set email=:email,
        password=:password where email=:email';
$types = array('text','text');
$data = array('email' => $email, 'password' =>
        $password);
$res = $this ->gestionBD ->execPrepared($req, $types,
        $data);
return $res;
}
}
```

Exemple d'utilisation

```
gestionBDUser = new GestionBDUser();

$gestionBDUser->createUser('monemail','monpassword');

$user = $gestionBDUser->getUserByEmail('monemail');
echo $user;

$users = $gestionBDUser->getAllUsers();
//print_r($users);

$gestionBDUser->updateUser('monemail','elo');
$user = $gestionBDUser->getUserByEmail('monemail');
echo $user;


$gestionBDUser->delUserByEmail('monemail');
```

Conclusion

- Nous pouvons lier nos objets à la base de données.
- Nous sommes ‘indépendant’ du SGBD
- Nous pouvons facilement changer la couche d’abstraction