

Architecture applicative de l'application Web

« GSB-AppliFrais-MVC »

ARCHITECTURE APPLICATIVE DE L'APPLICATION WEB « GSB-APPLIFRAIS-MVC »	1
Remarques préalables.....	1
Un développement guidé par les cas d'utilisation.....	1
Fonctionnement de l'application.....	6
Utilisation d'une classe d'accès aux données.....	8

Remarques préalables

Nous ne présentons pas ici les avantages de la structuration du code relevant de l'architecture Modèle-Vue-Contrôleur ; de nombreux documents se penchent sur la question. Des frameworks (Zend, Symfony, PhpCake) fournissent les classes mettant en œuvre cette technologie. Nous avons fait le choix ici de faire le travail « à la main » Nous nous bornerons à préciser certains choix faits. Ces choix se sont inspirés de deux travaux :

- Un document de **Serge Tahé** (2004-2008) qui présente une implémentation détaillée de l'architecture MVC pour php (en mode procédural -sans objet-) : <http://tahe.developpez.com/web/php/mvc/>
- Le travail d'**Olivier Cappuozzo** autour du contexte *festival* : <http://www.reseaucerta.org/cotecours/cotecours.php?num=363>

Un développement guidé par les cas d'utilisation

C'est le propre de l'architecture MVC ; le système (l'application) doit répondre aux sollicitations de l'utilisateur. Les cas d'utilisation sont les moyens textuels de décrire ces sollicitations et les réponses.

Prenons l'exemple du cas d'utilisation suivant :

PROJET : Application web de gestion des frais	<h2 style="text-align: center;">Description cas d'utilisation</h2>
Nom cas d'utilisation : Renseigner fiche de frais	
Acteur déclencheur : Visiteur médical	
Pré conditions : Visiteur médical authentifié	
Post conditions : néant	
<p>Scénario nominal :</p> <ol style="list-style-type: none"> <i>L'utilisateur demande à saisir un ou plusieurs frais pour le mois courant.</i> Le système retourne les frais actuellement saisis – éléments forfaitisés et hors forfait - pour le mois courant. <i>L'utilisateur modifie une ou des valeurs des frais au forfait et demande la validation.</i> Le système enregistre cette ou ces modifications et retourne ces valeurs à jour. <i>L'utilisateur ajoute un nouveau frais hors forfait en renseignant les différents champs – date d'engagement, libellé, montant - et valide.</i> Le système enregistre la ligne de frais hors forfait. 	
<p>Exceptions :</p> <p>2.a- C'est la première saisie pour le mois courant. Si ce n'est pas encore fait, le système clôt la fiche du mois précédent et crée une nouvelle fiche de frais avec des valeurs initialisées à 0. Retour à 3.</p> <p>4.a. Une valeur modifiée n'est pas numérique : le système indique 'Valeur numérique attendue '. Retour à 3.</p> <p>6.a Un des champs n'est pas renseigné : le système indique : 'Le champ date (ou libellé ou montant) doit être renseigné'.</p> <p>6.b La date d'engagement des frais hors forfait est invalide : le système indique 'La date d'engagement doit être valide'. Retour à 5.</p> <p>6.c La date d'engagement des frais hors forfait date de plus d'un an. Le système indique 'La date d'engagement doit se situer dans l'année écoulée'. Retour à 5.</p> <p><i>7. L'utilisateur sélectionne un frais hors forfait pour suppression.</i></p> <p>8. Le système enregistre cette suppression après une demande de confirmation.</p>	
<p>Contraintes :</p>	

L'utilisateur sollicite à 4 reprises le système (points 1, 3, 5 et 7 en *italique gras*). Le contrôleur (fichier spécifique) doit donc répondre à ces 4 sollicitations :

```

$action = $_REQUEST['action'];
switch($action){
    case 'saisirFrais':{
    case 'validerMajFraisForfait':{
    case 'validerCreationFrais':{
    case 'supprimerFrais':{
}

```

Remarque : le code des cases a été plié ici pour se concentrer sur l'essentiel.

Pour chacune des sollicitations, le système réagit et agit en conséquence, par exemple pour la demande de saisie des frais :

```

7  $action = $_REQUEST['action'];
8  switch($action){
9      case 'saisirFrais':{
10         if($pdo->estPremierFraisMois($idVisiteur,$mois)){
11             $pdo->creeNouvellesLignesFrais($idVisiteur,$mois);
12         }
13         break;
14     }
15     case 'validerMajFraisForfait':{
26     case 'validerCreationFrais':{
39     case 'supprimerFrais':{
44 }
45 $lesFraisHorsForfait = $pdo->getLesFraisHorsForfait($idVisiteur,$mois);
46 $lesFraisForfait= $pdo->getLesFraisForfait($idVisiteur,$mois);
47 include("vues/v_listeFraisForfait.php");
48 include("vues/v_listeFraisHorsForfait.php");
49
50 ?>

```

Le système teste (ligne 10) si c'est la première fois que l'utilisateur accède à cette demande de saisie de frais –cf extension 2.a- et va chercher en base (lignes 45-46) les données concernant les frais forfaitisés et non forfaitisés afin d'afficher les deux vues demandées (lignes 47-48). Ici, ces affichages sont communs aux autres cases.

Les deux vues affichées sont ici :

Renseigner ma fiche de frais du mois 9-2011

Eléments forfaitisés

Forfait Etape

Frais Kilométrique

Nuitée Hôtel

Repas Restaurant

Vue
Frais
Forfait

Descriptif des éléments hors forfait

Date	Libellé>	Montant	
25/08/2011	Invitation collaborateur	75.00	Supprimer ce frais

Nouvel élément hors forfait

Date (jj/mm/aaaa):

Libellé

Montant :

Vue
Frais
Hors
forfait

Nous avons fait le choix de présenter deux vues distinctes –nous aurions pu bien sûr mettre ce code dans un seul fichier- pour éventuellement réutiliser une de ces vue dans un autre cas d'utilisation.

Dans cette architecture, l'affichage des vues est provoqué par un ordre **include** (ou require) *nomVue*.

Pour respecter l'indépendance des couches (vue, modèle), le modèle (fichier php) retourne des tableaux :

```

public function getLesFraisForfait($idVisiteur,$mois)
{
    $req = "select fraisforfait.id as idfrais, fraisforfait.libelle as libelle,
    lignefraisforfait.quantite as quantite from lignefraisforfait,
    fraisforfait where fraisforfait.id = lignefraisforfait.idfraisforfait
    and lignefraisforfait.idvisiteur ='$idVisiteur' and lignefraisforfait.mois='$mois'
    order by lignefraisforfait.idfraisforfait";
    $res = PdoGsb::$monPdo->query($req);
    $lesLignes = $res->fetchAll();
    return $lesLignes;
}
    
```

La vue construit le code HTML à partir du tableau retourné :

```
<fieldset>
  <legend>Éléments forfaitisés
</legend>
<?php
    foreach ($lesFraisForfait as $unFrais)
    {
        $idFrais = $unFrais['idfrais'];
        $libelle = $unFrais['libelle'];
        $quantite = $unFrais['quantite'];
    }
    <p>
        <label for="idFrais"><?php echo $libelle ?></label>
        <input type="text" id="idFrais" name="lesFrais[<?php echo $idFrais?>]" size="10" maxlength="5" value="<?php echo $quantite?>" />
    </p>
<?php
}
~
```

Fonctionnement de l'application

C'est la page index qui sert d'aiguilleur principal et oriente vers un contrôleur de cas d'utilisation :

```
require_once("include/fct.inc.php");
require_once("include/class.pdgsb.inc.php");
include("vues/v_entete.php");
session_start();
$pdo = PdoGsb::getPdoGsb();
$estConnecte = estConnecte();
if(!isset($_REQUEST['uc']) || !$estConnecte){
    $_REQUEST['uc'] = 'connexion';
}
$uc = $_REQUEST['uc'];
switch($uc){
    case 'connexion':{
        include("controleurs/c_connexion.php");break;
    }
    case 'gererFrais' :{
        include("controleurs/c_gererFrais.php");break;
    }
    case 'etatFrais' :{
        include("controleurs/c_etatFrais.php");break;
    }
}
include("vues/v_pied.php");
?>
```

Ceci est à associer à ce que l'utilisateur sélectionne dans le sommaire :

```

</div>
<ul id="menuList">
  <li>
    <?php echo $_SESSION['prenom']. " ".$_SESSION['nom'] ?>
  </li>
  <li class="smenu">
    <a href="index.php?uc=gererFrais&action=saisirFrais" title="Saisie fiche de frais">Saisie fiche de frais</a>
  </li>
  <li class="smenu">
    <a href="index.php?uc=etatFrais&action=selectionnerMois" title="Consultation de mes fiches de frais">Mes fiches de frais</a>
  </li>
  <li class="smenu">
    <a href="index.php?uc=connexion&action=deconnexion" title="Se déconnecter">Déconnexion</a>
  </li>
</ul>

```

Louis
Villechalane

Saisie fiche de
frais

Mes fiches de
frais

Déconnexion

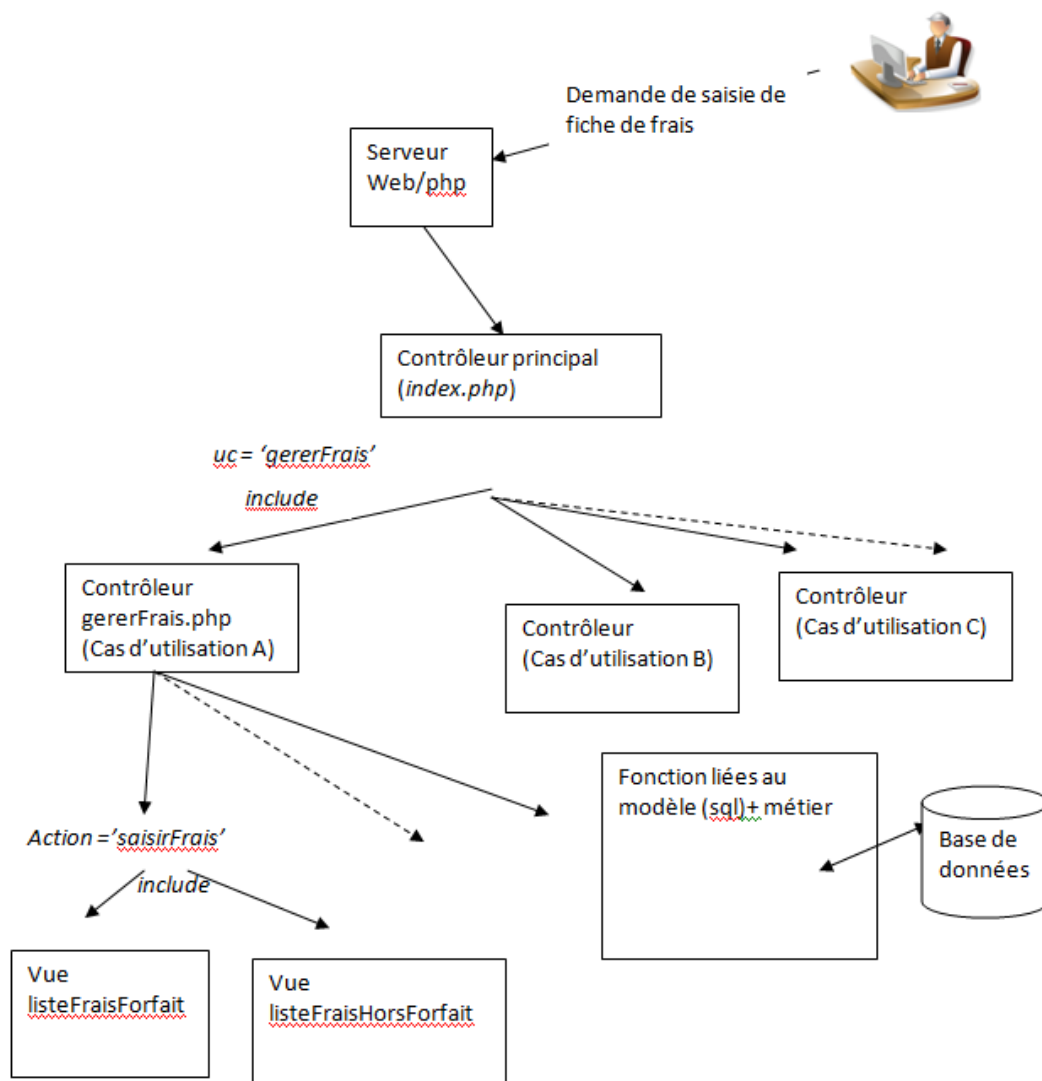
Renseigne

Eléments for

For

For

On peut résumer cette cinématique par un schéma :



Ainsi chaque page reçue est construite à partir de l'index comme une succession de fichiers *include* selon le cas d'utilisation. L'action demandée entraîne un traitement, à partir de la base de données et des règles métier (responsabilité de la couche Modèle) et expose les vues associées.

L'arborescence du site reflète cette architecture :

controleurs	16/08/2011 08:41	Dossier de fichiers
images	11/08/2011 16:29	Dossier de fichiers
include	16/08/2011 08:42	Dossier de fichiers
styles	11/08/2011 16:29	Dossier de fichiers
vues	16/08/2011 08:36	Dossier de fichiers
index	17/08/2011 23:46	Fichier PHP

Le répertoire *include* contient les fichiers utiles au modèle : accès à la base, fonctions métier, gestion des erreurs.

Utilisation d'une bibliothèque d'accès aux données

Php contient en standard une classe PDO, d'accès aux données ; elle a l'avantage d'être générique-indépendante du SGBD- et propose des services performants qui évitent la plupart du temps de faire soi-même les boucles de parcours des jeux d'enregistrements :

```
/**
 * Retourne tous les id de la table FraisForfait
 * @return un tableau associatif
 */
public function getLesIdFrais(){
    $req = "select fraisforfait.id as idfrais from fraisforfait order by fraisforfait.id";
    $res = PdoGsb::$monPdo->query($req);
    $lesLignes = $res->fetchAll();
    return $lesLignes;
}
/**
```

Cette classe n'est pas associée au modèle MVC, mais elle allège grandement la gestion des données.

Dans l'application GSB, la classe PDO est encapsulée dans une classe PdoGsb.