Architecture applicative de l'application Web « GSB-AppliFrais »

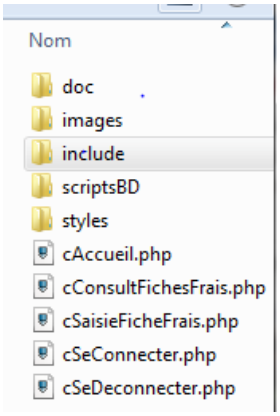
Principes d'organisation de l'application PHP Gsb-AppliFrais

Les principes d'organisation de l'application s'inspirent des travaux réalisés autour du contexte "Festival version allégée" : http://reseaucerta.org/cotecours/pub.php?num=390.

Le code PHP a été organisé de façon à respecter les contraintes suivantes :
- Les traitements ont été structurés selon les principes suivants :
  - la gestion des données est isolée dans un fichier à inclure : ce fichier comporte les fonctions de manipulation des données et les fonctions de connexion ;
  - la cinématique des cas d'utilisation et la gestion de l'affichage sont prises en charge dans les mêmes fichiers ;
  - la gestion des erreurs est décrite dans une bibliothèque de fonctions ad hoc.
- Séparer la présentation des informations de leur description : avoir recours à une feuille de style CSS pour la mise en forme afin que le langage HTML ne soit utilisé que pour décrire les informations.

Le langage HTML respectera la norme **XHTML1.0 version stricte**. Les pages seront validées à l'aide du validateur du W3C : http://validator.w3.org/check.
Les règles de style respecteront la norme **CSS2**. Elles seront validées auprès du validateur http://jigsaw.w3.org/css-validator/validator.

Organisation du stockage des fichiers de l'application



Le répertoire `styles` contient la(les) feuille(s) de style.
Le répertoire `images`, utilisé uniquement pour le logo, est prévu pour contenir les images figurant dans les différentes pages de l'application.
Le fichier `cAccueil.php` est le fichier de démarrage de l'application.
Les fichiers d'inclusion :

- `_debut.inc.html` contient l'entête de la page ;
- `_sommaire.inc.php` contient le menu de la page ;
- `_pied.inc.html` contient le pied de page ;
- `_init.inc.php` procède aux initialisations de variables et à l'inclusion des fichiers bibliothèques de fonctions ;
- `_fin.inc.php` procède à la libération des ressources ;
- `_utilitairesEtGestionErreurs.lib.php` est une bibliothèque de fonctions utilitaires et de gestion des erreurs ;
- `_bdGestionDonnees.lib.php` contient les fonctions de connexion et les fonctions de manipulation de la base de données.
- `_gestionSession.lib.php` contient les fonctions de gestion d'une session tq le démarrage d'une session, la vérification d'une connexion utilisateur, la consultation/modification des variables de session.

Règles de nommage côté scripts PHP

Outre les règles énoncées dans le document de référence GSB-STDDEVWEB, viennent s'ajouter les règles de nommage suivantes :

Item	Règle de nommage[1]
Variables	
jeu d'enregistrements	*$idJeu* suivi du rôle
ligne du jeu d'enregistrements	tableau *$lg* suivi du rôle
chaîne contenant une requête SQL	*$req* ou *$requete*
autre variable	pas de règle (le nom choisi doit toujours être porteur de sens par rapport au rôle de la variable)
Fonctions	
fonction retournant une requête SQL	*obtenirReq* suivi du rôle (exemple : obtenirReqEltsForfaitFicheFrais est le nom de la fonction qui retourne la requête permettant d'obtenir les données sur une fiche de frais)
fonction retournant une ligne ou une valeur	*obtenir* suivi du rôle (exemple : obtenirDetailFicheFrais est le nom de la fonction qui retourne la ligne correspondant à la fiche de frais demandée)
fonction d'ajout ou modification ou suppression	verbe *ajouter* ou *modifier* ou *supprimer* suivi du nom de la table. Si la fonction prend en charge plusieurs actions, on accole les noms des actions (exemple : ajouterFicheFrais)
fonction de vérification	son nom sera formé de *estUn* ou *verifier* ou *existe* suivi du rôle
Fichiers	
fichier contrôleur + affichage vue	Tout fichier contrôlant la cinématique des cas d'utilisation et la gestion de l'affichage commence par la lettre c (c comme contrôleur)

[1] Tous les noms respectent la règle « Camel » qui est notamment utilisée pour la programmation en langage Java.

Règles de nommage côté base de données

Concernant le schéma de la base de données les règles d'écriture suivantes ont été appliquées :

- pas de blanc ni de caractère accentué dans les noms de table ou d'attribut ;
- chaque nom de table commence par une majuscule et est suivi de minuscules. Si elle est composée de deux mots, ils sont collés et distingués par une majuscule ;
- chaque nom d'attribut est écrit en minuscule. S'il est composé de deux mots, ils sont collés et distingués par une majuscule. Le nom choisi pour l'attribut représente le rôle de son domaine dans la table ;
- une clef étrangère porte un nom significatif de son rôle dans la table.

Structure de chaque page de l'application

Toutes les pages contrôleur d'un cas ou sous-cas d'utilisation sont construites selon cette structure.

```
<?

  $repInclude = "./include/";
  require($repInclude . "_init.inc.php");



  require($repInclude . "_entete.inc.html");
  require($repInclude . "_sommaire.inc.php");
?>
<!-- Division pour le contenu principal -->
    <div id="contenu">
      <h2>Mes fiches de frais</h2>
...
    </div>
<?php
  require($repInclude . "_pied.inc.html");



  require($repInclude . "_fin.inc.php");
?>
```

Le fichier _init.inc.php (voir ci-dessous) contient toutes les initialisations de variables (identifiant de connexion au serveur MySql, tableau des erreurs).

L'en-tête (titre et barre de menus) est affiché grâce à l'exécution du code contenu dans les fichiers _entete.inc.html et _sommaire.inc.php. Dans ce fichier se trouve également la déclaration de la feuille de style `styles.css`.

Le pied de page est affiché grâce à l'exécution du code contenu dans le fichier_pied.inc.html.

Le fichier _fin.inc.php libère les ressources (identifiant de connexion au serveur MySql).

Fichier _init.inc.php

```
<?php
  require("_bdGestionDonnees.lib.php");


  require("_utilitairesEtGestionErreurs.lib.php");


  // initialement, aucune erreur ...
  $tabErreurs = array();


  // établissement d'une connexion avec le serveur de données
  // puis sélection de la BD qui contient les données des anciens
  $idConnexion=connecterServeurBD();

  if (!$idConnexion) {
     ajouterErreur($tabErreurs, "Echec de la connexion au serveur MySql");
  }

  elseif (!activerBD($idConnexion)) {

     ajouterErreur($tabErreurs, "La base de données gsb_frais est inexistante ou non
accessible");
  }
?>
```

Fonctions pour la gestion des données.

Fonctions utilitaires et de gestion des erreurs.

Création d'un tableau vide destiné à recevoir les messages d'erreur.

Appel de la fonction de connexion au serveur MySql.

Appel de la fonction de sélection de la base de données gsb_frais.

Détail des choix pour la gestion des données et la gestion des erreurs

Gestion des données (fichier `_bdGestionDonnees.lib.php`)

- Les interrogations de la base retournant une seule ligne sont entièrement prises en charge dans une fonction déportée ; cette fonction retourne alors le résultat dans un tableau (si plusieurs colonnes ont été demandées) ou dans une variable élémentaire.

Exemples :
obtenirDetailUtilisateur($idCnx, $unId) : retourne un tableau contenant les données de l'utilisateur d'id $unId.
obtenirDernierMoisSaisi($idCnx, $unIdVisiteur) : retourne une chaîne correspondant au mois (forme AAAAMM) de la dernière fiche de frais du visiteur d'id $unIdVisiteur.

- Les interrogations de la base pouvant retourner plus d'un enregistrement sont traitées ainsi :
  - constitution de la requête dans une fonction,
  - exécution de la requête et traitement du jeu d'enregistrements dans le code de la page appelante.

Exemple : obtenirReqLibellesFraisForfait
Appel à la fonction pour constituer la requête :

```
$req=obtenirReqEltsForfaitFicheFrais();
 // obtenirReqEltsForfaitFicheFrais est la fonction qui constitue le texte
 // de la requête permettant d'obtenir la liste des éléments forfaitisés
```

Exécution de la requête :

```
$idJeuEltsFraisForfait = mysql_query($req,$idConnexion);
```

Traitement du jeu d'enregistrements :

```
$lgEltForfait = mysql_fetch_assoc($idJeuFraisForfait);
while ( is_array($lgEltForfait) ) {
  ...
    $lgEltForfait = mysql_fetch_assoc($idJeuFraisForfait);
}
mysql_free_result($idJeuFraisForfait);
```

- Les mises à jour au sens large (modification, insertion, suppression) sont entièrement réalisées dans des fonctions.

Exemple : ajouterLigneHorsForfait(…)

Gestion des erreurs (fichier `_utilitairesEtGestionErreurs.lib.php`)

Principes de la bibliothèque de fonctions de gestion des erreurs

Par convention dans cette application, lorsqu'une erreur est détectée, un message d'erreur approprié est construit et fourni au système de gestion des erreurs. Ceci est simplifié par l'utilisation de la fonction *ajouterErreur*.

```
function ajouterErreur(&$tabErr, $msg) {
   $tabErr[count($tabErr)]=$msg;
}
```

$tabErr est le paramètre formel correspondant au tableau destiné à recevoir les différents messages d'erreur. Il est ici passé par référence car la fonction *ajouterErreur* doit modifier le contenu du tableau en y ajoutant un message dans le tableau.

Une fonction *nbErreurs* a été écrite pour retourner le nombre d'erreurs ; cela permet de tester le nombre d'erreurs avant d'appeler la fonction d'affichage des erreurs.

```
function nbErreurs($tabErr) {
      return count($tabErr);
}
```

La fonction d'affichage des erreurs parcourt le tableau des erreurs et les affiche les unes sous les autres.

```
function afficherErreurs($tabErr) {
     echo '<div class="erreur">';
     echo '<ul>';
     foreach($tabErr as $erreur)  {
        echo "<li>$erreur</li>";
     }
     echo '</ul>';
     echo '</div>';
}
```

Principes d'utilisation des fonctions de gestion d'erreurs

Nous illustrons ces principes grâce aux contrôles effectués sur le formulaire de modification d'une fiche de frais.

```
      // l'utilisateur valide les éléments forfaitisés
      // vérification des quantités des éléments forfaitisés
      $ok = verifierEntiersPositifs($tabQteEltsForfait);
      if (!$ok) {
          ajouterErreur($tabErreurs, "Chaque quantité doit être renseignée et
numérique positive.");
      }
      else { // mise à jour des quantités des éléments forfaitisés
          modifierEltsForfait($idConnexion, $mois,
obtenirIdUserConnecte(),$tabQteEltsForfait);
      }


     // si besoin, affichage des erreurs
          if ( $etape == "validerSaisie" ) {
              if ( nbErreurs($tabErreurs) > 0 ) {
                echo toStringErreurs($tabErreurs);
              }
```