Formation à l'utilisation d'un modèle d'application MVC

Création d'une application simple avec PrototypePHP

I. Installation

A. Pré-requis

Pour fonctionner, ce projet nécessite :

- un serveur web (apache par défaut dans cette documentation)
- php version 5
- un serveur MySQL et la possibilité soit de créer une base de données, soit d'exécuter un script de création de tables dans une base de données
- phpMyAdmin, pour gérer la base de données MySQL
- la version 0.4.1 ou ultérieure de prototypePHP (http://prototypephp.sourceforge.net)
- un éditeur PHP (recommandé : Eclipse PHP).

B. Installation

Importer le projet proto.0.4.2 depuis Sourceforge (http://prototypephp.sourceforge.net) dans **/var/www/protoform**

Dans phpmyadmin (pas pour les élèves de l'IUT, la base de données est déjà créée) :

- création de l'utilisateur **protoform**, mot de passe **protoform**, avec création d'une base de données de même nom.
- Gérer les droits pour donner les droits d'accès à l'utilisateur sur localhost (ou sur le serveur hébergeant la base de données)
- Pour créer un projet quelconque, import du fichier **protoform/install/proto.sql**, pour remplir la base de données. Dans le cadre de ce document, importer le fichier **protoform.sql**, fourni par ailleurs, qui contient les tables **personnel** et **civilite**.

Dans Eclipse :

- création d'un projet PHP dans le dossier **www/html/protoform**
- modifier dans fichier **param/param.inc.php** tous les champs ayant trait à la base de données (BDD), et l'emplacement de l'application
- modifier le fichier **param/gacl.ini.php**, et modifier également tout ce qui concerne la base de données (db_, à l'exclusion de db_table_prefixe).

Dans l'explorateur de fichiers, ou en mode console :

- rendre modifiable par tous **templates_c** :

```
chmod -R 777 templates_c
chmod -R 777 plugins/phpgacl/templates_c
```

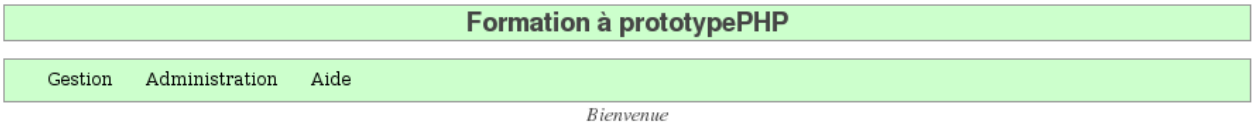
- vérifier les droits d'accès au compte Apache :

```
chown -R apache protoform (ou www-data avec une distribution debian ou ubuntu).
```


quinton.eric@gmail.com

Dans le navigateur :

- lancer l'application. Si tout ce passe bien, l'application initiale démarre.
- Ouvrir le fichier **locales/fr.php**, et modifier la variable **$LANG["message"][1]** pour mettre le nom de l'application.
- Cliquer sur Gestion ou Administration. Se connecter avec le login **admin**, mot de passe :**password**. Aller dans le module Administration / liste des comptes : la liste des logins existants est affichée.



La documentation d'utilisation de **ProtoypePHP** est disponible dans les rubriques du menu Aide.

II. Structure de la base de données

A. Structure de la table civilite

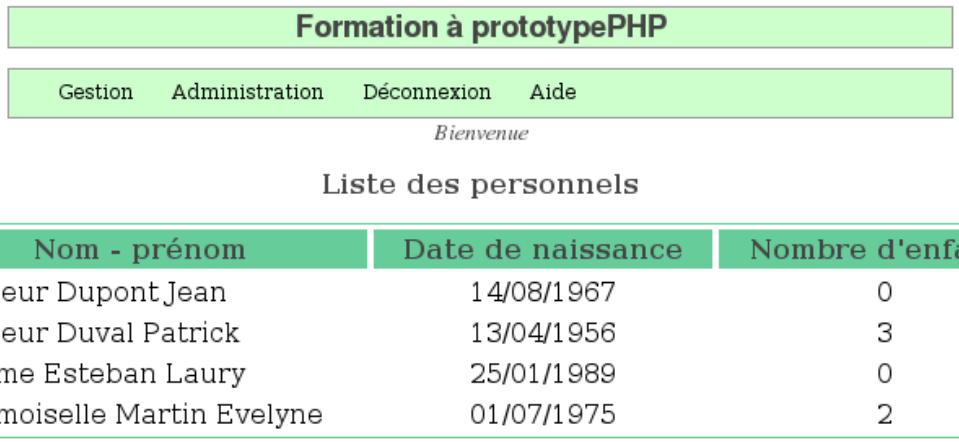
Champ	*Type*	*Null*	*Défaut*	*Commentaires*
id	int(11)	Oui	NULL	
libelle	varchar(32)	Oui	NULL	
ordreTri	smallint(6)	Oui	NULL	

B. Structure de la table personnel

Champ	*Type*	*Null*	*Défaut*	*Commentaires*
id	int(11)	Oui	NULL	
nom	varchar(32)	Oui	NULL	
prenom	varchar(32)	Oui	NULL	
dateNaissance	date	Oui	NULL	
nbreEnfants	smallint(6)	Oui	0	
civilite	smallint(6)	Oui	NULL	

III. Exercice 1 – afficher la liste des personnes

Dans un premier temps, nous allons afficher la liste des personnels présents dans la table Personnel. Nous devons arriver à un écran ressemblant à ceci :



A. Créer la classe Personnel

dans le dossier **gestion**, créer le fichier php **personnel.class.php**

Les classes vont être des classes héritées de ObjetBDD. Pour connaître le détail de la classe ObjetBDD, on consultera utilement la documentation sur http://objetbdd.sourceforge.net. La classe peut également être consultée directement dans *plugins/objetBDD/ObjetBDD.php*.

Créer la classe **Personnel**, héritée de ObjetBDD :

- définir le constructeur de la classe, qui doit supporter deux variables :
  - $link, qui contiendra l'intance ADODB ;
  - $param, un tableau optionnel, qui sera utilisé ultérieurement dans l'application pour modifier dynamiquement le comportement de la classe.
- le constructeur va permettre de décrire les informations suivantes :
  - le nom de la table (personnel)
  - la clé (id)
  - le fait que la table est gérée par une clé automatique (id_auto=1)
  - la liste des colonnes de la table, en définissant pour chaque colonne son type et si la valeur doit être renseignée ou non (champ obligatoire).
  - Inégrer dans le tableau $param la clé fullDescription = 1
  - une fois tous les paramètres définis, ne pas oublier d'appeler le constructeur de la classe parente, en y incluant l'instance de base de données et le tableau $param.

B. Créer la page permettant de lire la liste des personnels

Créer la page **gestion/personnelListe.php**. Cette page doit réaliser les opérations suivantes :

- intégrer le fichier contenant la définition de la classe Personnel ;
- créer une instance de la classe Personnel (avec, en paramètre, $bdd – instance ADODB, et $ObjetBDDParam, qui contient les paramètres permettant de modifier dynamiquement le fonctionnement d'ObjetBDD) ;
- récupérer dans la variable $listePersonnel le résultat de la fonction getListe() ;
- assigner à la classe Smarty la variable $listePersonnel (dans la variable « personnel ») ;
- assigner à la classe Smarty, dans la variable « corps », le nom du gabarit qui sera utilisé pour l'affichage : gestion/personnelListe.htm.

C. Créer le gabarit SMARTY

Créer un tableau HTML, qui comprendra 3 colonnes :

- nom – prénom
- date de naissance
- nombre d'enfants.

Une fois l'entête dessinée, créer une section SMARTY, dont le nom sera **tableau**, qui bouclera sur la variable **personnel** affectée précédemment. Pour chaque occurrence :

- créer une nouvelle ligne
- afficher les variables adéquates dans chaque cellule (par exemple, {$personnel[tableau].nom}
- ne pas oublier de fermer la boucle avec {/section}.

D. Créer les libellés qui seront affichés dans le menu

Ouvrir le fichier locales/fr.php, et rajouter les lignes suivantes :

```
$LANG["menu"][40] = "Liste des personnels";
$LANG["menu"][41] = "Liste de l'ensemble des personnels";
```

Ces informations seront utilisées pour afficher le menu.

E. Décrire le module

La page **personnelListe.php** ne va pas être appelée directement. Nous allons déclarer un module qui permettra d'exécuter le code correspondant.

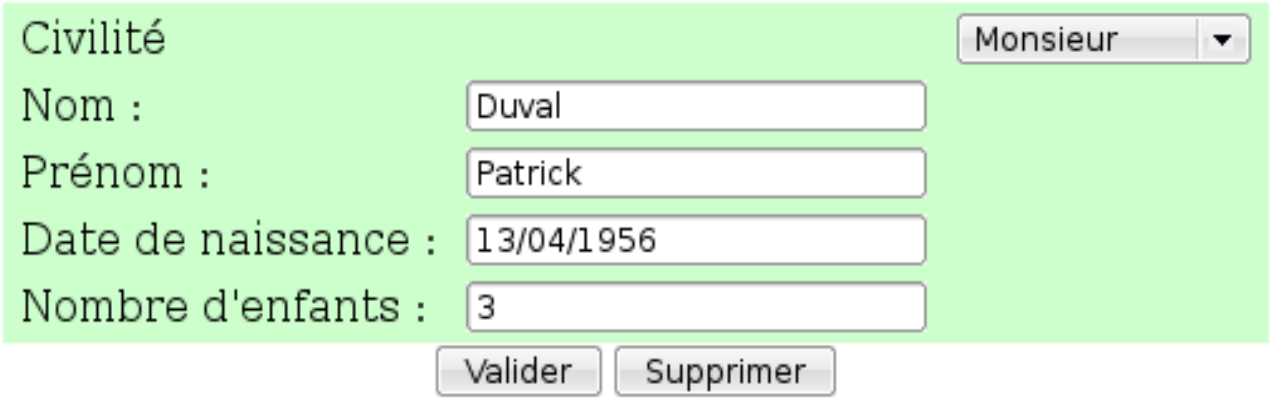
- ouvrir le fichier **navigation/actions.xml**.
- Rajouter un module, nommé **personnelliste**, avec les attributs suivants :
  - action : gestion/personnelListe.php
  - menulevel : 1 (il s'agit d'un sous-menu)
  - menuorder : 0 (c'est le premier item du sous-menu)
  - menuparent : 1 (ce sous-menu est rattaché au menu principal n° 1)
  - menuvalue : 40 (c'est le libellé qui sera affiché, celui saisi précédemment dans le fichier fr.php)
  - menutitle : 41 (c'est le libellé qui s'affichera si on laisse la souris en position sur l'item du menu).
- dans le module **gestion**, supprimer l'attribut **droits**.

F. Tester l'affichage de la liste des personnels

Lancer l'application. Dans le menu Gestion, nous avons maintenant un item : Liste des personnels. En cliquant sur cet item, la liste des personnels s'affiche.

IV. Exercice 2 : gérer la saisie d'une fiche Personnel

Nous allons maintenant rajouter la saisie d'une fiche Personnel à notre application. Nous devons arriver à un écran ressemblant à ceci :



Cet écran implique une sélection via une liste déroulante à partir de la liste des civilités. Nous avons donc besoin :

- de créer une classe Civilite pour afficher le contenu de la table ;

- de modifier la classe Personnel pour qu'elle soit capable de gérer la mise en fichier ;
- de créer une page PHP qui nous permettra d'enregistrer nos modifications ;
- de créer un gabarit SMARTY qui nous permettra de saisir les informations ;
- de rajouter des liens adéquats sur notre page « liste des personnels » pour pouvoir rajouter un nouvel enregistrement ou en modifier un.

A. Créer la classe Civilite

Dans notre fichier **gestion/personnel.class.php**, rajouter une nouvelle classe Civilite, héritée de ObjetBDD, selon le même principe que pour la classe Personnel.

B. Modifier la classe Personnel

Définir la fonction **ecrire($liste)**, $liste étant un tableau comprenant la liste des informations à écrire en base de données. En principe, $liste est alimenté par le super-tableau $_REQUEST, qui comprend toutes les variables passées par le navigateur.

Pour que l'écriture s'effectue, nous devons donc nous limiter aux champs qui nous intéressent, à savoir le nom des colonnes utilisées dans la table. Il faut donc :

- créer un tableau (par exemple, $tableau) ;
- affecter les données à écrire à ce tableau, par exemple :

```
$tableau["nom"] = $liste["nom"];
```

Cette opération est à effectuer pour toutes les colonnes, sans oublier la colonne contenant la clé (id).

Une fois le tableau renseigné, il faut appeler la fonction ecrire($tableau) de l'objet parent ObjetBDD, et renvoyer le code fourni par cette fonction.

C. Créer la page PHP permettant l'affichage des informations

Créer la page **gestion/personnelSaisie.php** :

- intégrer le fichier contenant la définition des classes Personnel et Civilite ;
- créer une instance de la classe Personnel (avec, en paramètre, $bdd – instance ADODB, et $ObjetBDDParam, qui contient les paramètres permettant de modifier dynamiquement le fonctionnement d'ObjetBDD) ;
- créer une instance de la classe Civilite (mêmes paramètres que pour l'instance de Personnel) ;
- récupérer dans la variable $detailPersonnel le résultat de la fonction personnel->lire($id). Attention : la variable $id est récupérée depuis le navigateur (tableau $_REQUEST) ;
- récupérer la liste des civilités dans la variable $listeCivilite par la fonction civilite->getListe() ;
- assigner à la classe Smarty les variables $detailPersonnel et $listeCivilite ;
- assigner à la classe Smarty, dans la variable « corps », le nom du gabarit qui sera utilisé pour la saisie : gestion/personnelSaisie.htm.

D. Créer la page PHP permettant l'enregistrement des informations

Nous allons créer la page **gestion/personnelModif.php**. Cette page va tester 3 cas différents, identifiés dans une variable $_REQUEST["action"] :

- nous avons une demande de modification ($action=='M') ;
- nous avons une demande de modification ($action=='S') ;
- nous n'effectuons rien du tout ($action prend une autre valeur, en principe : X).

Notre page va donc réaliser les opérations suivantes :

- intégrer le fichier contenant la définition des classes Personnel et Civilite ;

- créer une instance de la classe Personnel (avec, en paramètre, $bdd – instance ADODB, et $ObjetBDDParam, qui contient les paramètres permettant de modifier dynamiquement le fonctionnement d'ObjetBDD) ;
- Selon la valeur de $action, soit déclencher la modification ($personnel->ecrire($_REQUEST), soit déclencher la suppression de la fiche ($personnel->supprimer($_REQUEST['id'] ;
- en fonction du code de retour, afficher un message. Cette opération est réalisée en affectant un libellé à la variable $message. Les messages peuvent être récupérés depuis le fichier de traduction (locales/fr.php), dans le sous-tableau ['message']. Ainsi, pour indiquer que la mise en fichier s'est bien déroulée, nous réaliserons l'affectation suivante :

```
$message=$LANG['message'][5];
```

- enfin, nous allons indiquer le code de retour utilisé pour enchaîner sur le module d'affichage de la liste des personnels. Ainsi, si l'enregistrement s'est bien passé, nous indiquons :

```
$module_coderetour = 1;
```

Cette information sera détaillée dans la déclaration du module de gestion de l'enregistrement.

E. Créer le gabarit SMARTY

Nous allons créer le gabarit **templates/gestion/personnelModif.htm**, qui va comprendre un formulaire. Ce formulaire aura comme action :

```
<form method="post" action="index.php">
```

et les champs cachés suivants vont être rajoutés :

```
<input type="hidden" name="action" value="M">
<input type="hidden" name="id" value="{$detailPersonnel.id}">
<input type="hidden" name="module" value="personnelmodif">
```

Le premier champ correspond à l'action qui va être réalisée quand nous appuierons sur le bouton [Valider]. $id correspond à la clé de l'enregistrement, et $module au nom du module qui devra être appelé (c'est le module qui appellera la page personnelModif.php).

Nous allons créer également une boucle SMARTY pour afficher la liste des civilités. La saisie de la civilité va être réalisée par une liste déroulante, gérée par une balise HTML <select>. Voici le code correspondant :

```
<select name="civilite">
        {section name=lst loop=$listeCivilite}
        {strip}
        <option value="{$listeCivilite[lst].id}"
        {if $listeCivilite[lst].id == $detailPersonnel.civilite} selected{/if}
        >
        {$listeCivilite[lst].libelle}
        </option>
        {/strip}
        {/section}
</select>
```

Le test en milieu de script permet de positionner le curseur sur l'enregistrement courant. A noter l'usage de la balise SMARTY **{strip}** qui permet d'afficher le code sur plusieurs lignes.

- insérer ensuite les champs de formulaires adéquats pour pouvoir saisir les autres informations de l'enregistrement.

- Terminer en rajoutant deux boutons : l'un pour valider le formulaire, l'autre pour supprimer l'enregistrement. La suppression de l'enregistrement ne sera déclenché qu'après une demande de confirmation, qui va être gérée en Javascript.
  - Nous commençons par charger un script disponible dans l'application, qui va permettre d'afficher une boite de dialogue :

```
<script language="javascript" SRC="javascript/fonctions.js"></script>
```

  - puis nous déclenchons l'appel de ce script sur le bouton [Supprimer] :

```
<input type="submit" name="suppr" value="Supprimer"
onClick="javascript:setAction(this.form, this.form.action,'S')"/>
```

Ce script va positionner la variable **action** à **S** si la suppression est confirmée, sinon elle prendra la valeur **X**. C'est cette valeur qui sera analysée par le script **personnelModif.php** pour savoir quoi faire.

F. Déclarer les modules personnelsaisie et personnelmodif

Dans le fichier **navigation/actions.xml**, rajouter deux nouvelles actions :

- personnelsaisie :
  - action : gestion/personnelSaisie.php
- personnelmodif :
  - action : gestion/personnelModif.php
  - retourok : personnelliste
  - retourko : personnelsaisie
  - retournull : personnelsaisie.

Ainsi, si le code de retour est 1 ou 0, c'est la liste des personnels qui sera affichée, sinon nous retournerons sur l'écran de saisie.

G. Rajouter les liens dans la liste des personnels

A partir de la liste des personnels, nous avons maintenant besoin de pouvoir appeler la fiche de saisie. Nous allons créer deux liens :

- le premier va permettre de créer un nouvel enregistrement. Il va être inséré avant le tableau. Il peut prendre la forme suivante :

```
<a href="index.php?module=personnelmodif&id=0">Nouveau personnel</a>
```

L'identifiant est passé à la valeur 0, ce qui correspond, par convention, à la création d'un nouvel enregistrement (il n'y a pas d'enregistrement dont la clé vaut 0).

Il est alors facile de rajouter un lien sur chaque item du tableau, pour passer en mode modification, par exemple en cliquant sur le nom – prénom :

```
<a href="index.php?module=personnelmodif&id={$personnel[tableau].id}">
...affichage du nom et du prénom...
</a>
```

V. Exercice 3 : gérer les droits d'accès

Jusqu'à présent, notre application est accessible à n'importe qui, sans avoir besoin de s'identifier. Nous allons donc maintenant attribuer les droits et les gérer, grâce à phpGACL.

A. Corriger un bug dans l'application

- éditer le fichier **plugins/phpgacl/admin/index.php**, et ne conserver que la ligne :

```
header('Location: acl_admin.php');
```

B. Modifier les droits dans PHPGACL pour créer un profil lecture

Nous allons utiliser PHPGACL pour définir les droits d'accès. Pour cela, nous allons créer deux logins, **consult** et **gestion**.

- Se connecter à l'application avec le login **admin** (mot de passe : password).
- Afficher la liste des comptes, et créer les comptes **consult** et **gestion** ;

Remarque : dans le cadre de travaux pratiques en groupe, il est conseillé que chaque étudiant crée ses propres logins, en les préfixant de ses initiales ou de son nom, pour ne pas interférer avec les tâches de ses voisins.
De même, il est possible de créer, dans l'onglet **ARO group admin**, une nouvelle entrée en plus de application, qui prendra comme nom le nom de l'étudiant, toujours pour éviter que les droits se mélangent.

- Aller dans le menu Administration > gestion des droits.

Dans l'onglet **[ACL Admin]** :

- cliquer sur login, puis, dans la zone **Access Request Objects**, cliquer sur **Edit**
- rajouter les login **consult** et **gestion**.

Dans l'onglet **[ARO group admin]** :

- sur la ligne Application, cliquer sur **Assign ARO**
- rajouter le login **consult**.

Le login **gestion** est, par défaut, déjà dans le groupe **gestion**. Si l'item **Application** a été dupliqué dans le cadre d'un travail de groupe – cf. remarque ci-dessus – il faudra, dans ce cas de figure :

- dans l'onglet **[ARO group admin]** :
  - si ce n'est déjà fait, créer une nouvelle application, par exemple **appli1** (bouton **[Add]**, en bas d'écran) ;
  - sur la ligne correspondant à cette nouvelle application, cliquer sur **Add child**, et rajouter un item : **appli1_service1** ;
  - sur la ligne **appli1_service1**, cliquer sur Assign ARO, et sélectionner le login **gestion** ;
- dans l'onglet **[ACL admin]** :
  - cliquer sur **proto**
  - **Access Control Objects**, cliquer sur **Edit**
  - Rajouter **gestion**
  - revenir sur l'onglet
  - cliquer sur **proto**, puis sur **gestion**, puis **>>** pour le sélectionner
  - dans la zone **Groups**, cliquer sur **appli1_service1**, puis bouton **[Submit]**

Nous avons ainsi attribué les droits **gestion** au groupe **appli1_service1**, qui contient le login **gestion**. Nous allons faire la même opération pour donner les droits de consultation à notre login **consult** :

De nouveau, dans l'onglet **[ACL Admin]** :

- cliquer sur **proto**
- **Access Control Objects**, cliquer sur **Edit**
- Rajouter **consult**
- revenir sur l'onglet
- cliquer sur **proto**, puis sur **consult**, puis **>>** pour le sélectionner
- dans la zone **Groups**, cliquer sur **application**, puis bouton **[Submit]**.

Nous avons donné les droits **consult** au groupe **application**, c'est à dire à toutes les personnes dont le login a été attribué dans celui-ci ou dans ses sous-branches.

C. Modifier l'application pour tester la consultation

Editer le fichier **navigation/actions.xml** :
- dans **personnelliste**, rajouter l'attribut **droits**, avec comme valeur : **consult**.
- Dans **personnelmodif**, rajouter l'attribut **droits**, avec comme valeur : **gestion**.

Se connecter à l'application avec le login **consult**, et vérifier que l'on peut bien consulter la liste des personnels, mais que l'on ne peut pas accéder à la fiche de modification.

Se connecter alors avec le login **gestion**, et vérifier que l'on peut, maintenant, créer, modifier et supprimer un enregistrement.

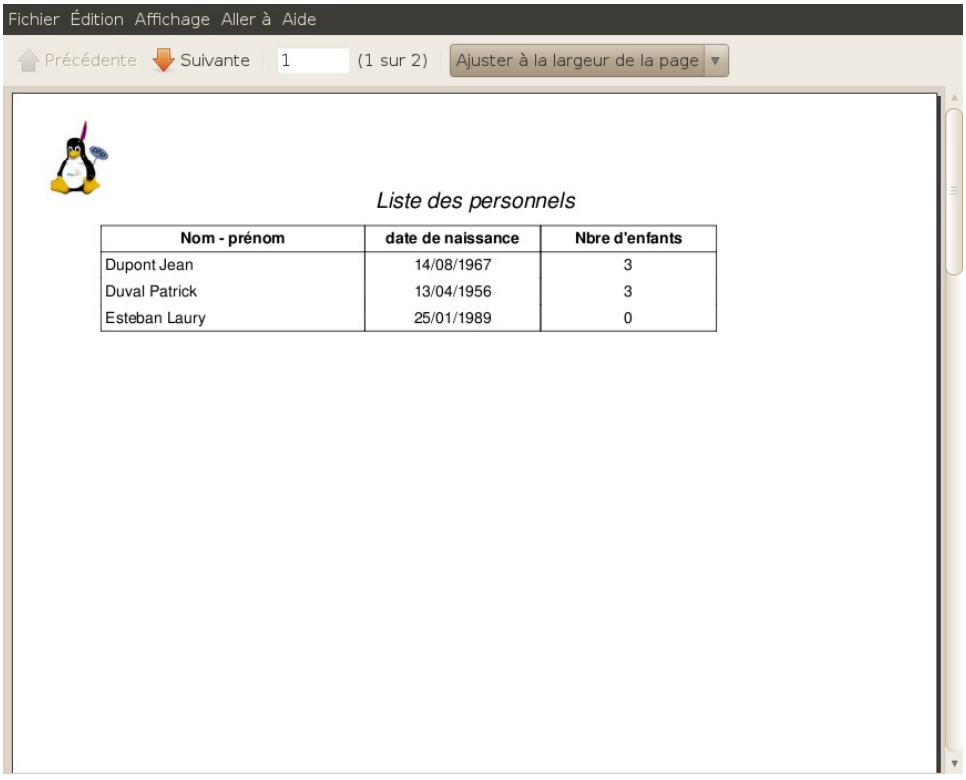
VI. Exercice 4 : rajouter une fonction permettant l'affichage de la liste des personnels au format PDF

A. Objectif

Rajouter une fonction permettant la génération d'un fichier PDF contenant la liste des personnels en utilisant la classe FPDF.

La liste des personnels sera affichée sous forme de tableau. Le programme gérera le nombre de lignes affichées dans chaque page de manière à pouvoir créer autant de pages que nécessaire. L'entête du tableau devra donc être générée à chaque changement de page.

L'entête contiendra une icône, et le titre du document. Le pied de page devra contenir le numéro de la page.



B. Préparer une classe héritée de fpdf

Cette classe va permettre de surcharger la classe d'origine, en préparant notamment l'entête et le pied de page.

Créer le fichier **gestion/pdf.class.php**.

```
<?php
/**
 * @author Eric Quinton
 * 04/05/2011
 */
```

Hériter la classe **fpdf**

```
/**
 * class PDF
 * mise en forme par defaut des documents PDF generes
 *
 */
class PDF extends FPDF {
```

Surcharge de la fonction **Header(),** qui sera appelée systématiquement lors de chaque nouvelle page

```
	/**
	 * Mise en place de l'entete
	 * (non-PHPdoc)
	 * @see FPDF::Header()
	 */
	function Header() {
		// positionnement de l'icone
		$this->image($this->icone,5,5,20,20,'JPEG');
		// Definition de la police
		$this->SetFont("arial", "I",14);
		// Calcul de la largeur du titre et positionnement
		$w=$this->GetStringWidth($this->title)+6;
		$this->SetX((210-$w)/2);
		// Ecriture du titre dans une cellule
		$this->Cell($w,9,$this->title,0,0,'C',false);
		// Insertion d'un saut de ligne
		$this->Ln(10);
	}
```

Surcharge de la fonction **Footer(),** pour rajouter le numéro de page en pied de page

```
	/**
	 * Definition du pied de page
	 * (non-PHPdoc)
	 * @see FPDF::Footer()
	 */
	function Footer() {
		// Positionnement a 15mm du bas
		$this->SetY(-15);
		// definition de la police
		$this->SetFont("arial","",8);
		// Recuperation du nombre de pages total
		$this->AliasNbPages('nbpages');
		// mise en place de la numerotation
```

```
            $this->Cell(0,10,$this->PageNo().'/nbpages',0,0,'C');
        }
```

Rajouter une fonction permettant d'indiquer l'icône qui sera utilisée dans l'entête

```
        /**
         *
         * Definition de l'icone utilisee
         * @param string $icone
         */
        function setIcone($icone){
            $this->icone = $icone;
        }
```

Rajouter une fonction qui permettra de générer l'entête du tableau sur chaque page

```
        /**
         * Création de l'entete du tableau
         * @param int $hauteur : hauteur de chaque ligne
         * @param array $largeur : liste des largeurs des colonnes
         * @param arrray $titre : liste des intitules des colonnes
         */
        function SetEnteteTableau($hauteur,$largeur,$titre) {
            // Definition du tableau - largeur des colonnes
            // Preparation de l'entete du tableau
            $this->SetFont('arial','B',10);
            for ($i=0;$i< count($largeur);$i++) {
                $this->Cell($largeur[$i],$hauteur,$titre[$i],1,0,'C');
            }
            // saut de ligne
            $this->ln();
        }
}
?>
```

C. Création du module de génération du document

Créer le fichier **gestion/personnelPDF.php**, et intégrer les classes nécessaires :

```
<?php
/**
 * Script permettant de generer un tableau de la liste
 * des personnels, au format PDF
 * @author Eric Quinton
 * 04/05/2011
 */
include_once("plugins/fpdf16/fpdf.php");
include_once("gestion/pdf.class.php");
include_once("gestion/protoform.class.php");
```

Définir les paramètres généraux du document PDF

```
// Definition de la classe et taille du document
$pdf = new PDF('P','mm','A4');
```

```
// definition de l'icone
$pdf->setIcone("images/tux-lamp.jpg");
$pdf->SetFont('Arial','',10);
$pdf->SetMargins(20,20,20);
// Definition des proprietes du PDF (carte de visite)
$pdf->SetAuthor($_SESSION["login"]);
$pdf->SetSubject("Liste des personnels");
$pdf->SetKeywords("personnels formation prototypePHP");
// Definition du titre
$pdf->SetTitle("Liste des personnels");
// Definition du tableau - largeur des colonnes
$largeur = array(60,40,40);
$largeurTotale=140+$pdf->lMargin;
$titreTableau = array("Nom - prénom", "date de naissance", "Nbre d'enfants");
$hauteur = 6;
$nbLigne = 0;
$nbPage = 1;
$nbreLigneParPage=2;
```

Récupérer la liste des personnels, et l'afficher :

```
// Traitement de chaque item de la liste des personnels
$personnel = new Personnel($bdd,$ObjetBDDParam);
$listePersonnel = $personnel->getListe();
foreach ($listePersonnel as $key=>$value) {
	// Generation de l'entete du tableau
	if ($nbLigne==0) {
		if ($nbPage > 1) {
			// Fermeture du tableau sur la page precedente
			$pdf->Line($pdf->lMargin,$pdf->GetY(),$largeurTotale,$pdf->GetY());
		}
		// Creation d'une nouvelle page
		$pdf->AddPage();
		$nbPage++;

		$pdf->SetEnteteTableau($hauteur, $largeur, $titreTableau);
		$pdf->SetFont('arial','',10);
	}
	// Affichage des informations
	$pdf->Cell($largeur[0],$hauteur,$value["nom"].' '.$value["prenom"],"LR");
	$pdf->Cell($largeur[1],$hauteur,$value["dateNaissance"],"LR",0,'C');
	$pdf->Cell($largeur[2],$hauteur,$value["nbreEnfants"],"LR",0,'C');
	$pdf->Ln();
	$nbLigne++;
	if ($nbLigne>$nbreLigneParPage) {
		$nbLigne=0;
	}
}
```

Exporter le document PDF vers le navigateur :

```
// Terminaison du tableau
$pdf->Line($pdf->lMargin,$pdf->GetY(),$largeurTotale,$pdf->GetY());
// Envoi du PDF au navigateur
$pdf->Output("ListePersonnel.pdf","I");
?>
```

D. Intégrer le module dans l'application

1. Créer les libellés pour générer le menu

Editer le fichier locales/fr.php, et rajouter les deux lignes suivantes :

```
$LANG["menu"][42] = "Liste PDF des personnels";
$LANG["menu"][43] = "Génération de la liste de l'ensemble des personnels au format PDF";
```

2. Rajouter le module dans le fichier actions.xml

Editer le fichier navigation/actions.xml, et rajouter l'élément personnelpdf :

- action : gestion/personnelPDF.php
- droits : gestion
- menulevell : 1
- menuorder : 1
- menuparent : 1
- menutitle : 43
- menuvalue : 42

VII. Exercice 5 : modifier la feuille de style pour la rendre plus proche des standards du web

- recopier la feuille **proto.css** en **protonew.css**
- modifier les entrées adéquates pour :
  - supprimer les marges gauche et droite
  - remettre les liens hypertexte en mode classique (souligné, couleur standard (bleu/violet) ou noir/gris). Ne pas intervenir dans le module « menu » ;
  - supprimer les indications de police et de taille, si nécessaire.
- Modifier le fichier **param.inc.php**, et intégrer la nouvelle feuille de style.

Table des matières