Rallye Lecture : Evolution de la gestion des livres

Techniques mises en œuvre :

- ✓ Utilisation du Framework codeIgniter
- ✓ Utilisation du design pattern MVC
- ✓ Persistance des données au travers de classes DAO
- ✓ Gestion d'une pagination multiple
- ✓ Upload d'un fichier sur un site web

Objectif : faire évoluer la gestion des livres :

- Gérer une pagination dans la liste des livres. Actuellement la liste des livres est trop longue, il y a une centaine de livres dans la base, consulter les livres pour un utilisateur peut devenir assez difficile. Dépassé un certain nombre de livres cela peut même être pénalisant pour les performances du site en transmettant sur internet un grand volume de données. Pour optimiser l'affichage nous allons gérer une pagination : limiter la liste des occurrences visibles lors de l'affichage d'une page.
- Améliorer l'ergonomie de la saisie des livres : Ajouter l'upload de fichiers pour les images des couvertures. Lorsque l'on crée ou modifie un livre, l'utilisateur a la possibilité d'associer une couverture à un livre. Nous allons lui donner la possibilité de sélection un fichier jpeg ou tout autre fichier image puis d'uploader ce fichier sur le site afin qu'il soit disponible lors des consultations des visiteurs.

1. Pré requis : Modifier le lien de navigation dans la vue LivreIndex

Pensez à Modifier la vue : LivreIndex.php sinon les modifications liées à la pagination crée un bug

```
<div class="navigation">
    <a href="<?php echo base_url(); ?>">Home</a>
    <a href="<?php echo base_url('livre/add');?>">Ajouter</a>
```

2. Modification de la gestion des livres ajout de la pagination

Typiquement nous voulons obtenir le résultat suivant :

uri différente pour chaque page affichée

numéro du premier livre affiché ici le 110ème dans la table (indice de base 0)

pagination de 10 livres

déplacement au début et à la fin de la pagination

liste des pages

CodeIgniter nous fournit un ensemble d'outils pour répondre assez facilement et rapidement à ce problème. Voyons ce que nous devons mettre en œuvre.

Evolution du modèle du Livre

Pour paginer il faut récupérer les livres à partir d'une position ici 110 (c'est un paramètre $start) et nous récupérons les 10 livres suivants (c'est un paramètre que l'on nommera $count).

Il y a déjà une méthode nommée get_all_livres qui fait à peu près ce que nous voulons mais malheureusement elle retourne tous les livres.
Il faudrait créer une surcharge de cette méthode. Mais il n'existe pas de notion de surcharge en php. Il va falloir détourner le problème en gérant nous même la surcharge :

Modifions la méthode get_all_livre prenant en compte deux paramètres, ajoutons ces deux paramètres :

- n° d'enregistrement de début à retourner ($start) et
- nombre d'enregistrements à retourner à partir de l'enregistrement de départ ($count).

Faites évoluer la signature de la méthode get_all_livres en prévoyant c'est deux paramètres. Mais pour que cette méthode puisse continuer à fonctionner sans paramètres, l'astuce consiste à définir des valeurs par défaut à chacun des paramètres. Ainsi s'ils ne sont pas alimentés, ils prendront la valeur par défaut et comme la valeur par défaut que nous prévoyons est NULL les paramètres ne sont pas obligatoire (on a presque réussi à créer une surcharge) :

```
function get_all_livres($start=NULL,$count=NULL) {

}
```

Maintenant dans le corps de la méthode il faut prévoir tous les cas de paramètres possibles : ici il n'y en a que deux :

⇒ Premier cas : $start et $count ne sont pas définis : c'est ce que fait actuellement la méthode et on utilise le query suivant :

```
$this->db->get('livre') qui retourne tous les livres
```

⇒ Deuxième cas : $start et $count sont définis (on utilise la fonction php isset pour tester) dans ce cas on utilise le query suivant :

```
$this->db->get('livre',$count,$start ) qui retourne $count livres à partir du $start livre…
```

Dans tous les cas on retourne le jeu d'enregistrement résultat du query.

A noter que le deuxième cas génère une requete sql particulière utilisant la clause LIMIT. Exemple :

```
$query = $this->db->get('mytable',10,20);
// génère la requete sql suivante et affiche
// les 10 premières occurrences en partant de la 20ème
SELECT * FROM mytable LIMIT 20,10
```

Attention aux deux paramètres qui s'inversent entre la méthode php et la requête sql !!!

Ajout d'un comptage dans des enregistrements de la table.
La bibliothèque pagination que nous allons mettre en œuvre un peu plus loin a besoin de connaitre le nombre d'enregistrements à paginer, il lui faut donc obtenir le comptage des occurrences de la table Livre

2 solutions possibles :
1ère solution. Plus optimisée car nécessitant moins d'accès à la base de données :

- Faire évoluer le modèle Livre en intégrant une donnée membre supplémentaire nommée count qui permet de connaitre le nombre d'éléments récupérés dans un query.
- Créer un accesseur publique get_count sur cette donnée membre pour qu'elle soit accessible aux autres classes extérieures.
- Lors de l'appel à get_all_livres penser à renseigner la donnée membre count avec le nombre de livres contenus dans la table.

2ème Solution moins sujette au bug mais nécessitant une plus grande sollicitation de la base :
Créer une méthode publique get_count qui exécute une requête sur la table et retourne directement le comptage des enregistrements de la table.

Quelque soit la solution la méthode get_count reflétera donc exactement le nombre d'occurrences contenues lors de la manipulation d'un modèle de livres. Elle est nécessaire pour savoir sur combien d'éléments on doit paginer et déterminer ainsi le nombre de pages.Exemple : elle sera utile pour calculer la pagination : 132 livres dans la table 10 livres par page cela donne 14 pages.

Evolution du contrôleur
Il s'agit ici de faire évoluer la méthode index de manière à se qu'elle tienne compte de la pagination.
CodeIgniter propose une bibliothèque et une classe qui va nous permettre de mettre en œuvre cette pagination :

Pensez à placer dans le constructeur du contrôleur Livre le chargement de la librairie "pagination".

Puis il faut préparer la pagination dans la méthode index :

`$config['base_url']=site_url().'/Livre/index/page';`	ajoute la page de pagination dans l'uri
`$page=$this->uri->segment(4,0);`	récupère le numéro de page ou le rang du premier livre
`$config['total_rows']=$this->livreModel->get_count();`	récupère le nombre de livre total et l'initialise dans la pagination
`$config['per_page']=10;`	nombre de livres par page
`$config['uri_segment']=4;`	en quel position se trouve le numéro de page dans l'uri
`$this->pagination->initialize($config);`	le paramètres sont initialisés dans l'objet pagination
`$data['livres']= $this->livreModel->get_all_livres($page,$config['per_page']);`	on récupère les 'per_page' livres suivants à partir de la position $page
`$links=$this->pagination->create_links();`	create_links() permet de générer la liste des liens href vers les différentes pages possibles. Contiendra une chaine de caractère qui une fois affichée devrait donner un rendu qui pourra resssembler à ceci : <<début 1 2 3 4 5 6 fin>> $links devra être passé en paramètre à la vue, la vue devra afficher cette chaine, c'est elle qui permettra de passer les paramètres nécessaires à l'affichage de la prochaine page.

⇒ On passe en paramètre à la vue la variable $links.

Autre solution pour les paramètres de pagination :

On peut placer les paramètres de config de pagination dans un fichier de configuration. Voici comment procéder :
- ✓ Créer un fichier texte nommé pagination.php,
- ✓ Déclarer un tableau $config dans ce fichier avec toutes les variables vues précédemment,
- ✓ Sauvegarder ce fichier dans application/config/pagination.php, il sera automatique chargé au démarrage de l'application. Dans ce cas vous n'avez pas à initialiser les paramètres $config et à utiliser $this->pagination->initialize(). La logique du reste du traitement ne change pas.

Evolution de la vue
On ajoute en bas de page l'affichage de la variable $links. C'est tout.

Enfin vous avez tout un ensemble de paramètres permettant de personnaliser la pagination.

$config['uri_segment'] = 3	Quelle section de l'uri contient le numéro de page. Ici la troisième section.
$config['num_links'] = 2	Nombre de positions numérique affichées pour les numéro de pages.
$config['use_page_numbers'] = TRUE	index de départ du premier objet de la pagination. Si vous préférez voir un numéro de page dans l'uri, il faut mettre cette propriété à TRUE (c'est codeigniter qui calcul les numéro de page).
$config['page_query_string'] = TRUE	FALSE : les uri des pages sont construits de la manière suivante : http://example.com/index.php/test/page/20 TRUE : les uri des pages sont construits de la manière suivante : http://example.com/index.php?c=test&m=page&per_page=20
$config['full_tag_open'] = ' '	Vous pouvez entourer le lien de pagination avec des balises html. Une ouvrante et une fermante.
$config['full_tag_close'] = ' '	
$config['first_link'] = 'First'	Texte du lien partie gauche ou FALSE si vous voulez qu'il n'apparaisse pas.
$config['first_url'] = ''	Une url alternative pour la première page
$config['last_link'] = 'Last'	Texte du lien partie droite ou FALSE si vous voulez qu'il n'apparaisse pas.
$config['next_link'] = '&gt'	Texte du lien du suivant
$config['prev_link'] = '&lt	Texte du lien du précédent
… d'autres dans la doc	toutes les balises peuvent être modifiées.

3. Ajout de l'upload d'un fichier

Nous allons développer un système d'upload de fichiers pour faciliter l'ajout des jaquettes des livres gérés par notre base de données. Dans la base nous gérons le nom du fichier image et le fichier est physiquement stocké dans le répertoire /img.

Modification de la vue

Nous allons commencer par modifier la vue : LivreAdd.

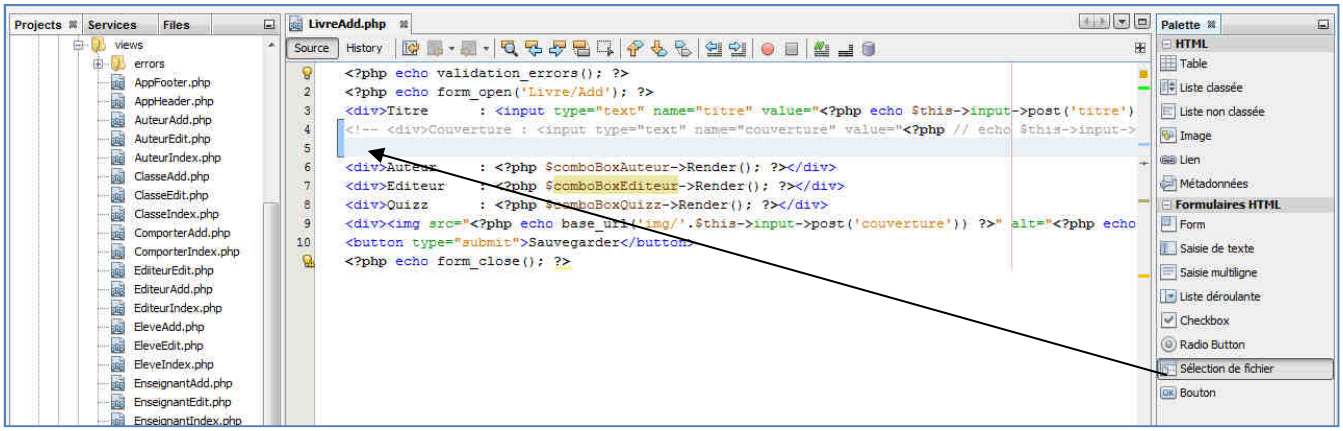
Ouvrez le source,

Dans netBean lancez le menu : Fenêtre/Outils IDE/ Palette,

Une palette de composant html standards et de composants liés au formulaires apparaissent.

Commenter la ligne n°4 du code. Nous allons remplacer la zone de saisie de text par une sélection de fichier.

Sélectionner le composant : "Sélection fichier".



Effectuez un drag and drop vers la ligne laissée vide dans le code.

Une boite de dialogue doit s'ouvrir complétez la.

changer la ligne

```
form_open('Livre/Add');
```

en

```
form_open_multipart('Livre/Add');
```

nécessaire pour que l'on puisse transférer des données binaires (l'image) de la balise de type "file".

Contrôleur

Le constructeur est modifié comme suit : on définit les caractéristiques de l'upload et on charge l'objet upload dans le contrôleur :

```
function __construct() {
        // code partiel
        $config['upload_path']='./images/';
        $config['allowed_types']='gif|jpg|png';
        $config['overwrite']=TRUE;
        $config['max_size']='2000';
        $config['max_width']='1024';
        $config['max_height']='768';
        $this->load->library('upload',$config);
```

la méthode add du contrôleur Livre va faire appel à une nouvelle méthode upload_image dont voici le code :

```
    function upload_image() {
            if (!$this->upload->do_upload('couverture')) {
                    $error=TRUE
            }
            else {
                    $error=FALSE;
            }
            return $error;
    }
```

Enfin modifiez la méthode add()

```
function add() {
        //  code partiel
        // appelez la méthode upload image
        // récupérer dans la donnée membre $this->upload grace à
        // l'accesseur file_name le nom du fichier image
        // puis transmettez le nom du fichier image au tableau de paramètres
        // qui est transmis à la méthode add_livre de livreModel qui va effectuer l'insert en base de
        // données.
}
```

Poursuite du ppe

- Vous pouvez mettre en place la pagination sur d'autres écrans qui le nécessitent.
- Au moment de la création d'un nouveau Livre, il serait souhaitable que l'utilisateur lorsqu'il a choisi une image puisse la voir apparaitre à l'écran.
- A vous de gérer la modification d'un livre pour gérer l'upload.

Brouillions

Questionnement possible :
dans la liste des livres remplacer le supprimer individuel par une case à cocher + bouton supprimer.
changer count en GetCount
Menu le gérer dans json et chargement au démarrage. ou en fichier parametre de CI
liste des livres decoder les libelles des auteurs et des editeurs
edit : gerer la modification et l'upload de l'image
Add : après avoir sélectionné l'image, l'afficher à l'utilisateur avant la validation

La solution à ce problème vous est donnée, il faut juste la coder et la comprendre.

Modification du Modèle de Menu.php
Cette classe permet de retourner en fonctions du groupe auquel appartient l'utilisateur qui se logue (Administrateur, Elève, Enseignant ou Visiteur) de retourner les options de menu qui correspondent à ses droits. Ainsi chacun reçoit un menu qui lui est adapté.
Ces options de menu sont stockées directement dans un tableau dans la Classe Menu. Chaque option de menu correspond à une ligne de tableau. Chaque ligne de tableau est structurée comme suit :

- Contrôleur à instancier,
- méthode à exécuter,
- et libellé du menu apparaissant dans la vue.

Problème : gérer la pagination ajoute un niveau supplémentaire dans la définition de certaines uri gérées dans le tableau.
Le menu livre ne possède plus trois informations mais nécessite un niveau supplémentaire pour y accéder :

Livre/Index/Page/110
Page est un niveau supplémentaire à gérer
En 4ème paramètre nous verrons plus tard que l'on ajoutera le numéro de l'occurrence de départ à afficher.

```
class Menu extends CI_Model {
    function __construct() {
        parent::__construct();
    }

    function GetOptions($group) {
        switch ($group) {
            // en fonction du groupe on retourne le menu correspondant
            // structure {'Controleur','méthode','libellé)
            case 'Admin':
                return array(
                    array('Auteur','Index','Auteur'),
                    array('Editeur','Index','Editeur'),
                    array('Livre','Index','Page','Livre'),
                    array('','','#'),
                    array('Enseignant','Index','Enseignant'),
…
```

Le menu ne prend pas en compte ce poste supplémentaire Lors de l'affichage de la vue. Solution il faut faire évoluer la vue Menu.php de la manière suivante :

```
<!-- navigation -->
<div class="navigation">
    <a href="<?php echo base_url(); ?>">home</a>
    <?php foreach ($options as $parametres) : ?>
        <?php
        $count=count($parametres);
        $i=0;
        $uri = '';
        foreach ($parametres as $parametre) {
            if ($i<$count-1) {
                $uri=$uri."{$parametre}/";
            }
```

```
            else {
                $option=$parametre;
            }
            $i++;
        }?>
        <a href="<?php echo $uri; ?>"><?php echo $option; ?></a>
    <?php endforeach; ?>
</div>
```

Au lieu d'utiliser un tableau de 3 postes. On boucle sur tous les postes du tableau que l'on considère chacun comme étant un élément de l'Uri. Le dernier élément étant considéré comme étant le libellé du menu à afficher.
Le contrôleur du Menu n'est pas impacté par ces modifications. Mais vous pouvez jeter un coup d'œil et voir comment il fonctionne.

Contrôleur Menu.php

```
<?php

/** @property Menu $Menu
 *  @property CI_Session $session
 */
class Home extends CI_Controller {
    function __construct() {
        parent::__construct();
        $this->load->model('Menu');
    }

    public function Index() {
        // par défaut toute personne accédant au site fait partie du groupe visiteur
        // on teste si la variable de session existe
        if ($this->session->groupe) {
            $groupe=$this->session->groupe;
        }
        else {
            $groupe='Visiteur';
            $this->session->groupe=$groupe;
        }
        // on charge l'entete et la page d'accueil
        $data['title']='rallye lecture';
        $this->load->view('AppHeader',$data);
        // le menu est conditionné par le groupe.
        $data['options']=$this->Menu->GetOptions($groupe);
        $this->load->view('menu',$data);
        $this->load->view('AppFooter');
    }
}
```