Inf340 Systèmes d’information

Troisième site

Objectifs

Objectif :

- Reprendre le site précédent en utilisant le framework CodeIgniter.
- Appréhender les limites de ses modèles

- Le site permet de gérer un carnet d'adresse composé d'une liste de noms et de numéros de téléphone, un nom peut posséder plusieurs numéros de téléphones.

Bilan du premier site

Nous avions un site :

- Sans sécurité
- Ne dispose pas d’une communauté d’utilisateurs

- Facilement déployable
- Simple à maintenir

Les framework

- Un framework est en français un cadre d’application.
- Un framework est un ensemble d'outils et de composants logiciels organisés conformément à un plan d'architecture et des design patterns (Wikipedia).
- En PHP : Zend Framework, Symfony, CakePHP, CodeIgniter, …

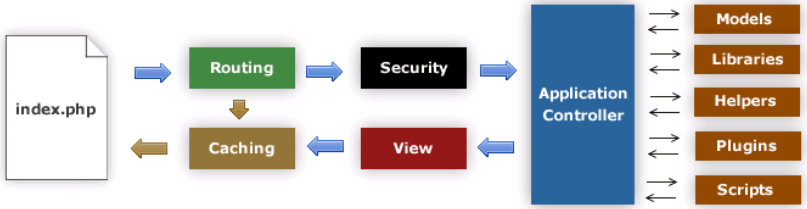
CodeIgniter

Un framework léger avec :

- MVC
- ActiveRecord
- Gestion de la sécurité (Injection SQL + XSS)
- Fortement orienté objet (librairies) mais aussi des bibliothèques de fonctions (helpers)
- Gestion de caches

CodeIgniter

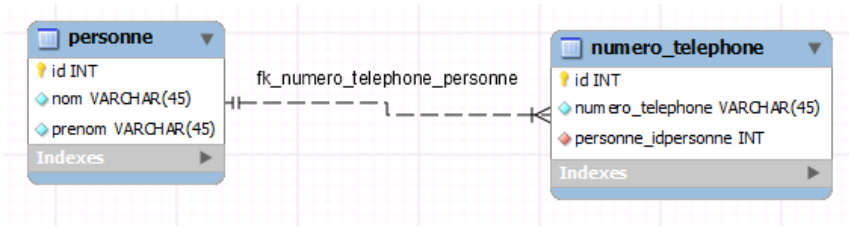
- Index.php est le contrôleur principal
- Routing : examine la requête HTTP et choisi l'action a déclancher
- Security : filtre les entrées
- Application Controller : les contrôleurs de pages
- View : les vues
- Caching : le cache



Modèle

Basé sur le design pattern ActiveRecord : une classe étends la classe CI_Model et est utilisée pour manipuler la table de même nom. Chaque table a une clef primaire nommée id.

```
class Personne_model extends CI_Model{
private  $nom;
private  $prenom;

 function __construct()
   {
     parent::__construct();

   }

 function setNom($nom){
 $this->nom = $nom;
 }


 function  setPrenom($prenom){
 $this->prenom = $prenom;
 }

}
```



Modèle

Dans cette démo, nous avons utilisé une bibliothèque fonctionnelle, un helper, pour accéder au modèle.

```
function getAllPersonne() {
$CI = & get_instance();

$query = $CI->db->get('personne');
return $query->result();
}

function createPersonne($nom, $prenom) {
$CI = & get_instance();

$CI->db->set('nom',$nom);
$CI->db->set('prenom',$prenom);
$CI->db->insert('personne');
}
```

Contrôleur

Le contrôleur est une classe qui hérite de CI_Controller : il charge les bibliothèques et les librairies et en fonction des URL sollicite le modèle et affiche les vues

```
class Personne extends CI_Controller{
public function __construct(){
parent::__construct();
$this->load->library('input');
$this->load->helper('url');
$this->load->helper('my_personne_helper');
$this->load->helper('my_numero_telephone_helper');
}

public function index()
{
$data['personnes']= getAllPersonne();
$this->load->view('templates/header');
$this->load->view('accueil_view',$data);
$this->load->view('templates/footer');
}

public function create(){
$nom = $this->input->post('nom');
$prenom = $this->input->post('prenom');
createPersonne($nom, $prenom);
redirect('personne');
}
 …
```

Vue

Les vues reçoivent les données depuis le contrôleur et les affiches

```
<?php foreach ($personnes as $personne):?>
<tr>

<td> <?php echo $personne->id; ?></td>
<td> <?php echo $personne->nom; ?></td>
<td> <?php echo $personne->prenom; ?></td>
<td> <a href="<?php echo site_url('personne/update/'.$personne->id);
    ?>"> modifier </a></td>
<td> <a href="<?php echo site_url('personne/delete/'.$personne-
    >id);?>"> supprimer </a></td>
</tr>
<?php endforeach;?>
```

Action read

- Pas de lecture sur GET et POST
- Sollicitation du modèle
- Affichage d’une vue

Action read

URL : http://site/personne

```
public function index()
{
$data['personnes']= getAllPersonne();
$this->load->view('templates/header');
$this->load->view('accueil_view',$data);
$this->load->view('templates/footer');
}
```

Action create

- Lit en POST les données d’un formulaire
- Sollicite le modèle
- Effectue une redirection pour produire un rafraichissement.

Action create

URL: http://site/personne/create

```
public function create(){
$nom = $this->input->post('nom');
$prenom = $this->input->post('prenom');
createPersonne($nom, $prenom);
redirect('personne');
}
```

Action delete

- lit en GET les données d’une ancre
- Sollicite le modèle
- Effectue une redirection

Action delete

URL : http://site/personne/delete/id

```
public function delete($idpersoone){
deletePersonne($idpersonne);
redirect('personne');
}
```

La modification

- La modification est en deux phases
  - L'affichage du formulaire de modification : l'action update
  - Le traitement de l'information du formulaire de modification : l'action updateOk

L’action update

- Lire en GET depuis une ancre l’identifiant de l’enregistrement à modifier.
- Solliciter le modèle pour récupérer l’enregistrement depuis l’identifiant
- Afficher la vue de modification

L’action update

URL : http://site/personne/update/id

```
$data['personne']= getPersonneById($idpersonne);
$data['numeros'] =
    getNumeroTelephoneByIdPersonne($data['personne']-
    >id);

$this->load->view('templates/header');
$this->load->view('consulter_modifier_view',$data);
$this->load->view('templates/footer');
```

L’action updateOk

- Attend en POST depuis un formulaire
- Sollicite le modèle
- Actualise le modèle

L’action updateOk

URL : http://site/personne/updateOk

```
public function updateOk()
{
$id = $this->input->post('idpersonne');
$nom = $this->input->post('nom');
$prenom = $this->input->post('prenom');
updatePersonne($id, $nom, $prenom);
redirect('personne');
}
```

Le modèle

Un classe qui est le mapping d'une table est des bibliothèques de fonctions.

Exemple récupérer tous les enregistrements :

```
function getAllPersonne() {
$CI = & get_instance();

$query = $CI->db->get('personne');
return $query->result();
}
```

Le modèle

```
function createPersonne($nom, $prenom) {
$CI = & get_instance();

$CI->db->set('nom',$nom);
$CI->db->set('prenom',$prenom);
$CI->db->insert('personne');
}
```

Les vues

- Pour le moment pas de langage de templates bien que CodeIgniter en possède un.
- Elles sont invoquées par les contrôleurs
- Elles utilisent les données du modèle fournies par le contrôleur.

- Mécanisme particulier pour récupérer les données depuis le contrôleur, si le contrôleur passe $data[‘un_nom’]=valeur alors dans la vue on utilise $un_nom pour accéder à la valeur.

accueil_view.php

```
<?php foreach ($personnes as $personne):?>
<tr>

<td> <?php echo $personne->id; ?></td>
<td> <?php echo $personne->nom; ?></td>
<td> <?php echo $personne->prenom; ?></td>
<td> <a href="<?php echo site_url('personne/update/'.$personne->id); ?>"> modifier
    </a></td>
<td> <a href="<?php echo site_url('personne/delete/'.$personne->id);?>"> supprimer
    </a></td>
</tr>
<?php endforeach;?>
```

Avec dans le contrôleur :

```
$data['personnes']= getAllPersonne();
$this->load->view('accueil_view',$data);
```

Conclusion

- Un gain en temps de développement et en sécurité
- La possibilité de disposer du travail d'une communauté
- Un modèle objet mais restrictif qui impose des contraintes à la base

=>

Utiliser un autre modèle : Doctrine

La suite

- Présentation de l'ORM doctrine
- Intégration de l'ORM doctrine dans notre site