Docu-Pro

Contexte

Etablissement

Une grande surface spécialisée dans différents secteurs (notion de Monde).

Développement

Il s'agit de créer un gestionnaire de documentation adapté à une entreprise, qui permettra aux salariés de gérer/créer puis consulter/rechercher dans une documentation partagée.

Outils

- Développement sous Eclipse.
- Utilisation de doctrine, DAO, JsUtils.
- Utilisation du framework CodeIgniter.

Langages

- PHP
- MySQL
- jquery

Fonctionnalité : Recherche quelconque

• Définition de la fonctionnalité :

La fonctionnalité « rechercher » permettra donc à un membre de saisir une valeur puis de choisir le type de recherche. Recherche de type document, theme, utilisateur, groupe, domaine, monde ou bien de tout type.

• Contenu

1/ La valeur recherchée et le choix du type seront récupérés en paramètre dans une fonction :

function SearchBar ($valeurRecherchée, $type=utilisateur)

La fonction sera appelée via le click d’un bouton représentant une image de loupe, logo habituellement associé à la recherche et/ou via un keyPress de la touche Entrée.

2/ Ensuite ces valeurs seront retransmises vers une deuxième fonction : cette dernière contiendra donc plusieurs requêtes SQL selon le type du style :

SELECT id, tnom FROM utilisateur WHERE nom like '%valeurrecherhée%'

3/ Enfin, il faudra afficher pour chaque résultat un lien permettant à l’utilisateur d’afficher le contenu d’un monde par exemple, ou le contenu d’un document. Il sera donc appelé au click, une troisième fonction qui

contiendra l'id de l'élement cliqué ainsi que son type!

Exemples : si on clique sur le monde informatique ayant pour id=2, cette troisième fonction récupère donc l'integer 2 et le type=monde et exécute une requête du style :

If (type==monde){ SELECT * FROM domaines WHERE idmonde =2 ; }

Controleur

→ Un seul contrôleur « CRecherche ».

Voici les fonctions :

Nom	Description
index()	Charge les vues
_construct()	Charge la librairie pour les sessions
search()	Récupère la valeur recherchée ainsi que les cases cochées par l'utilisateur puis retourne ces informations vers resultatSearch()
resultatSearch()	Effectue les requêtes SQL selon les cases cochées et la valeur recherchée. Tout type de résultat est géré (valeur recherchée nulle, aucun résultat après la recherche,etc)
loadViewExecSearch()	Fonction exécutée au click d'un résultat : si on clique sur un monde, cette fonction se chargera d'afficher les domaines contenus dans ce dernier. Si on clique sur un domaine, sera alors affichés les thèmes présents dans ce domaine, etc

Vues

Nom	Description
vHeader	Affichage de l'en tête de la page
vMenu	Affichage du menu à gauche
vRecherche	Affichage de la zone de saisie pour la recherche, les cases a cochées et une zone consacrées aux résultats
vResultatRecherche	Affichage des résultats dans la zone prévue dans la vue vRecherche
vExecSearch	Affichage des informations concernant l'élément cliqué toujours dans la zone prévue dans la vue vRecherche
v_footer	Affichage du pied-de-page

Base de donnée



! Remarque :

- il faut supprimer les ORM pour les models Utilisateur, Partie et Version.
- Des modifications sont nécessaires sur la base de donnés : dans la table partie, il faut rajouter un champ document_id, qui est donc en relation avec la table document. Pensez à ajouter ce dernier champ au model Partie sur eclipse !
- Dans la fonction loadViewExecSearch, le click sur un utilisateur n'a pas encore été traité. Des modifications de la base de données sont nécessaires pour cela.



Code

Fonction search()

```
        public function search(){
        // on récupere la valeur recherchée
        $searched=$_POST['searchBar'];
        // stock l'array des casés cochées en session (le temps d'une requête)
        $caseChecked=$_POST['Recherche'];
        $this->session->set_flashdata('caseChecked', $caseChecked);
        // get avec la valeur recherchée par l'utilisateur vers resultatSearch
$this->jsutils->get("CRecherche/resultatSearch/".$searched,"#SearchResultContainer"
);
        echo $this->jsutils->compile();
    }
```

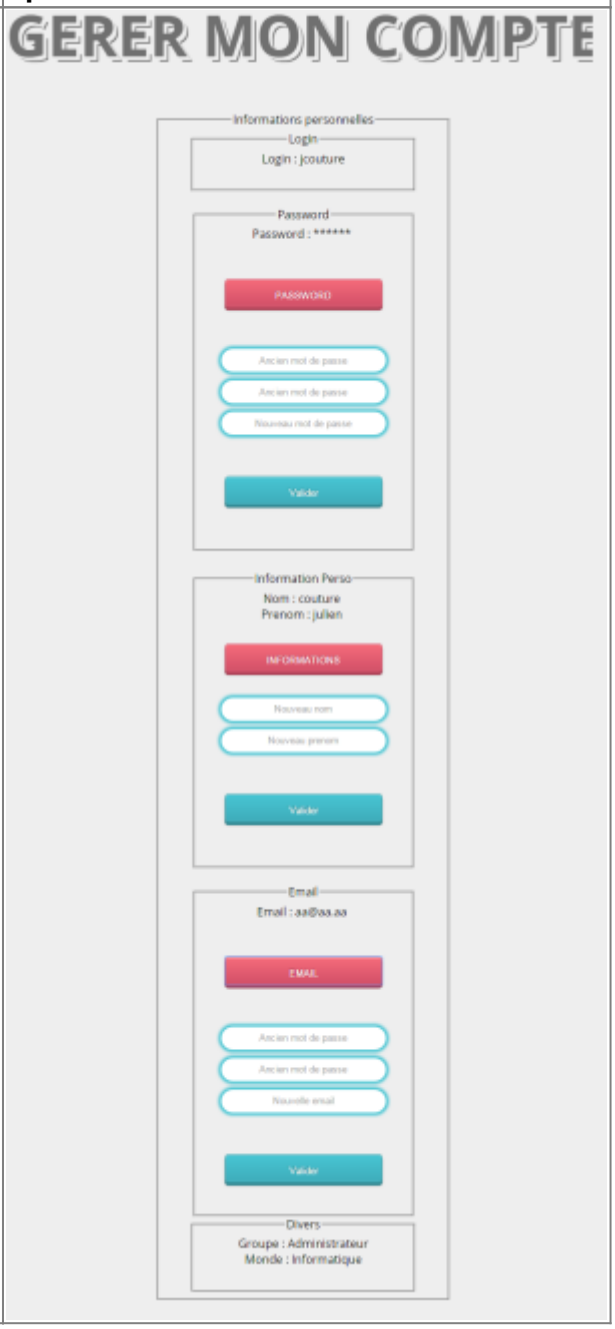
Utilisation de DAO

```
    /**
     * @brief Modification du mot de passe
     * @details    saisi de l'ancien mot de passe + nouveau mot de passe
     * @details Vérifie si l'un des champs n'est pas vide
     */
    public function _modifierMDP(){
        $Passe1 = $_POST['AncienPassword'];
        $passe2 = $_POST['AncienPassword2'];
        $newPasse = $_POST['NewPassword'];
        $user = $this->_chargerUtilisateur();
        if($Passe1 != null || $passe2 != null || $newPasse != null){
            if($this->_verifierMDP($Passe1,$passe2)){
                $user->setPassword($newPasse);
                DAO\update($user);
                echo "Modification de votre mot de passe réussi";
            }
        }
        else{
            echo "veuillez saisir les champs mots de passe";
        }
    }
```

Exemple DQL (doctrine)

```
<?php
//Création de la requête d'update
$query = $this->doctrine->em->createQuery("UPDATE utilisateur u SET u.nom ='%$nom%'
WHERE u.id=1");
//Exécution de la requête.
$search = $contenu->execute();
```

Design

Avant	Aprés
	

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/etudiants/2014/jeremy.habit/docuwiki?rev=1419863215

Last update: **2019/08/31 14:31**