

# DokuMission

## Contexte

### Développement

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

### Outils de développement

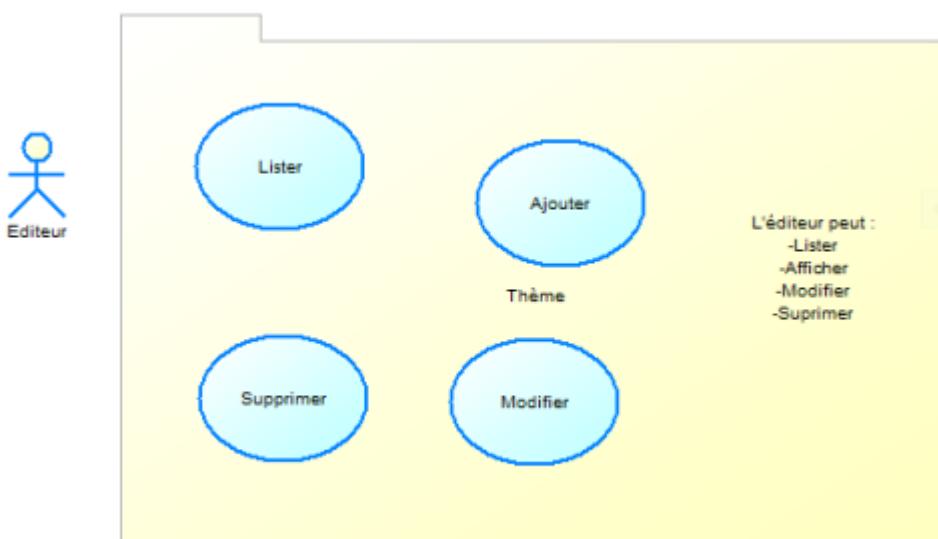
- PHP/Mysql
- Doctrine pour le mappage relationnel/Objet
- CodeIgniter pour la mise en place MVC

## Travail fourni (Thème)

### Document

Libelle	Document
Scénario	descriptif_textuel.pdf
Base de donnée	docu_4_.sql
Application	dokumission.zip

### Diagramme de cas d'utilisation des thèmes



## Modèle MVC

## Modèle

Pour pouvoir travailler sur les thèmes, il a fallu agir sur 4 classes métier:

1. Domaine
2. Thème
3. Document
4. Utilisateur

- Un utilisateur est associé à des documents s'il en possèdent.
- Un utilisateur peut travailler dans un ou plusieurs domaines.
- Les documents sont identifier dans des thèmes.
- Un thème appartient à un domaine.
- Un thème peut posséder un thème parent.
- Un thème appartient à un utilisateur.

## Vue

Pour la gestion des thèmes 3 vues sont utilisées:

- v\_theme (Vue général, liste les thèmes triés + formulaire d'ajout et de modification)

**Gestion des thèmes**

Domaine :	Domaine: Développement ▾
<b>Analyse</b>	Thème parent: Aucun ▾
Analyse fonctionnelle	<b>Ajouter:</b> <input type="text"/>
<a href="#">Modifier</a> <a href="#">Supprimer</a> - 2 Document(s)	<b>Ajouter</b> <input type="button"/>
Comédie	
<a href="#">Modifier</a> <a href="#">Supprimer</a> - 1 Document(s)	
<b>Danse</b>	
Classique	
<a href="#">Modifier</a> <a href="#">Supprimer</a> - 0 Document(s)	

- v\_confirmdeletetheme (La vue s'affiche si aucun document n'est affecté)

Suppression

Confirmer Suppression de **Classique** ?

Oui  Non

**Envoyer**

- v\_formdeletetheme (Avant suppression, on demande s'il on souhaite garder les documents)

Suppression:

Thème : Comédie

Document associé :

- Les films comiques

Souhaitez-vous également supprimer les documents ?

Oui  Non

## Contrôleur

La gestion des thèmes est effectué dans Gtheme. Le contrôleur possède plusieurs méthode pour optimiser la mise en place des fonctionnalités.

- index (Méthode appelée par défaut "appel les vue supplémentaire tel que le menu, footer...")
- refresh (Appelée par l'index "initialise les outils nécessaire au premier affichage de la page (Paramètre de la BDD/fonction JS)")
- addParentThem/updateParentThem (Ces méthode ajoute/modifie l'ensemble des thèmes qui existe pour le domaine choisi pour les ajouter dans les listes déroulante)
- add (Ajoute un nouveau thème)
- theme\_modif (Est appelé quand on veut modifier un thème, celle-ci pré-remplit le domaine, thème parent)
- update (met à jour les informations modifiées)
- deleteForm (Cette méthode vérifie si le thème possède des documents, suivant le résultat la vue affichée ne sera pas la même)
- checkConfirmDelete (Appelée par **deleteForm** s'il n'y a pas de document)
- delete (Supprime le theme)
- saveDocs (Verifi si l'utilisateur a décidé de garder ou non les documents )
- updateDocs (Dans la table document, met l'identifiant de thème à nul)
- deleteDocs (Supprime le document)

## Librairie

Appelé dans le constructeur : `$this->load->library('Modelutils');` Utilisation de la librairie **ModelUtils**, dans cette classe 2 modèles ont étaient créées.

- cleanPost (nettoie les variables)
- ifempty (Recoit un tableau pour vérifier chaque contenu, si vide retourne false)

```
/**
 * Récupère un array puis vérifie si une variable est vide
 * @param $params
 * @return boolean
 */
public function ifempty($params=array()){
    $checked=true;
    foreach ($params as $param){
        if($checked==true){
            if(empty($param)){
                $checked=false;
            }
        }
    }
    return $checked;
}
```

## Requête doctrine utilisées

Les méthodes doctrine sont appelées de la manière suivante : `$this->doctrine->em->`

Pour un **Select** :

1. Appel de `createQuery` (création de la requête)
2. `getResult` (Récupération multiple)
3. `getSingleResult` (Récupérer une seule valeur)

Pour un **Update**:

1. Appel de `createQuery`
2. `execute` (Exécute la requête de mise à jour)

Pour une **insertion**:

1. `Persist` (Prépare l'instance à être insérée)
2. `flush` (Ajoute l'instance dans la base)

## Jsutils

Les fonctions jsutils sont appelées pour rendre le contenu dynamique. Pour pouvoir l'utiliser il a été ajouté dans le constructeur `$this->load->library('jsUtils');`

**\$library\_src** doit-être ajouté la vue pour charger la librairie jquery, **\$script\_foot** script compilé pour effectuer les actions

Les méthodes jsutils sont appelées de la façon suivante : `$this->jsutils->`

Liste des méthodes utilisées :

1. `postFormAndBindTo` Poste le formulaire
2. `getAndBindTo` Effectue une action suite à un événement (les paramètres de l'id Html sont envoyés automatiquement)
3. `click` Lors d'un clic sur un élément on appelle une autre méthode
4. `show` Affiche l'élément mis en paramètre
5. `hide` Cache l'élément mis en paramètre
6. `get` Effectue une redirection
7. `doSomethingOn` Peut servir à ajouter ou supprimer des éléments dans un contenu existant
8. `compile` Ajoute la portion de code écrite vers la vue

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

Permanent link:  
<http://slamwiki2.kobject.net/etudiants/2014/beaugrand/dokumission?rev=1418681109>

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

