

## **Analyse pour un CMS en symfony**

- Dénombrer les applications symfony nécessaires à l'élaboration d'un CMS
- Dénombrer les parties génériques d'un CMS

### ***Gestion d'utilisateurs***

- Décrire le modèle utilisateur
- Décrire le modèle groupe
- Dénombrer les entités nécessaires à l'implémentation en symfony de la gestion de groupes d'utilisateurs

### ***Gestion des permissions***

- Sachant qu'elles doivent être associées à un utilisateur et / ou un groupe, dénombrer les logiques possibles d'héritages des permissions
- Dénombrer les entités nécessaires à l'implémentation symfony d'un tel système de permissions
- Ecrire la méthode getCredentials de l'objet utilisateur
  - Discuter la complexité induite par le fonctionnement de symfony (création d'un objet ⇔ requête à la base de données) au niveau du système de permissions appliqué au système de gestion de groupes d'utilisateurs

### ***Gestion d'une arborescence de contenus***

L'arborescence doit pouvoir présenter des parties privées, c'est à dire qui ne sont visibles qu'à partir du back office et avec les bonnes permissions. La notion de confidentialité doit pouvoir être héritée par tous les objets contenus dans une partie de l'arborescence privée

- Gestion de dossiers
  - Décrire le modèle des dossiers (contenants)
- Gestion de contenus
  - Dénombrer les différents types de contenus insérables dans l'arborescence de documents
  - Décrire le modèle des contenus
  - Gestion d'objets métier
    - proposer une façon d'intégrer des objets métiers dans l'arborescence
    - Proposer une liste d'objets génériques utiles dans un cms et dénombrer les permissions associées à chacun de ses objets
  - Dénombrer les différentes actions et permissions associées aux objets dossier et contenu
  - Expliquer la différence entre la gestion des permissions au niveau arborescence (sur les dossiers et contenus) et la gestion au niveau des objets métiers
  - Proposer une implémentation pour la gestion des permissions sur l'arborescence

## **modèle de base de données pour un CMS en symfony**

- Proposer un modèle complet de base de données permettant d'implémenter simplement en symfony le cms décrit ci dessus
- Proposer un modèle internationalisé

## **système de gestion des fichiers**

- Proposer une gestion de fichiers pour les contenus de type fichiers
- Proposer une gestion de fichiers pour les fichiers attachés au type html
- Proposer une gestion de fichiers pour les fichiers associés aux objets métiers

## **pour le TP**

### ***Le Rapport***

- Présente l'étude générale vue en TD
- Présente les fonctionnalités effectivement implémentées dans le TP rendu
- Présente toutes les fonctionnalités de symfony que vous avez appréciées et qui vous ont fait gagner du temps par rapport à une implémentation MVC pure comme vue dans le TP précédent.
- Présente également toutes les pistes optionnelles explorées

### ***minimum à implémenter***

- une gestion d'utilisateur simple (sans groupe)
- une gestion de permission minimale (aucun, utilisateur, administrateur)
- le contenu page web
- l'objet métier link factory, intégré dans l'arborescence via le cms

### ***pistes optionnelles à explorer (avec ou sans utilisation de plugin)***

- gestion complète des utilisateurs
  - <http://www.symfony-project.org/plugins/sfGuardPlugin>
- gestion d'un ensemble complet et fermé de permissions sur l'arborescence
- gestion de tous les types de contenus insérables dans l'arborescence
- Mis au point d'un système d'url de navigation
- gestion de tags
  - <http://www.symfony-project.org/plugins/sfPropelActAsTaggableBehaviorPlugin>
  - <http://www.symfony-project.org/plugins/sfTagtoolsPlugin>
- Gestion de la concurrence d'accès
  - il doit exister un plugin ...
- Versioning des contenus
  - <http://www.symfony-project.org/plugins/sfPropelVersionableBehaviorPlugin>
- Moteur de recherche interne
  - <http://www.symfony-project.org/plugins/sfLucenePlugin>
- Solution pour cacher l'information via http (mode intranet)
  - [http://www.stadtaus.com/fr/php\\_scripts/download\\_center\\_lite/](http://www.stadtaus.com/fr/php_scripts/download_center_lite/)
- Gestion des commentaires
  - <http://www.symfony-project.org/plugins/sfPropelActAsCommentableBehaviorPlugin>
- Possibilité de personnaliser l'affichage css d'un dossier ou d'un contenu
- mise en place d'un système de kit graphique pour une personnalisation rapide du layout global
- Transformation du cms en plugins symfony

May the force be with you !

## Données

### *les admin generator*

<http://www.symfony-project.org/screencast/admin-generator>

### *Simuler une permission 'propriétaire' ('owner') dans symfony*

<http://snippets.symfony-project.org/snippet/18>

## Problématique

Imaginons que votre sites gère des utilisateurs qui peuvent créer des posts. Vous voulez restreindre l'édition de chaque post à l'administrateur ou au propriétaire du post. Vous souhaiteriez dans ce cas avoir un fichier `security.yml` de ce genre là

```
edit:
  is_secure:on
  credentials: [[administrateur owner]]
```

mais le problème est que les permissions symfony sont données de manières statiques au moment du login

## Solution

La solution est d'attribuer ou de retirer dynamiquement la permission. La seule façon d'effectuer cette opération est de réécrire la méthode `getCredential` de la classe `sfAction`.

Voici un extrait du controleur du module post:

```
// fichier /apps/mapp/modules/post/actions/actions.class.php
function getCredential()
{
    // on retrouve l'objet post courant
    $this->post = PostPeerRetrieveByPk($this->getParameters('id'));
    if ($this->getUser()->isOwnerOf($this->post))
        $this->getUser()->addCredential('owner');
    else
        $this->getUser()->removeCredential('owner');
    // on a réécrit la partie qui nous intéressait
    // on reprend le traitement normal de la méthode
    return parent::getCredential();
}
```

Bien entendu la méthode `sfUser::isOwnerOf` est à écrire et dépend de la structure des données. Cette méthode doit retourner vraie si le post est nouveau, sinon les utilisateurs ne pourront pas créer de posts.

Maintenant un utilisateur peut éditer ses posts mais pas ceux des autres, sauf dans le cas où il est administrateur. Tout est géré dans le fichier `security.yml`. Par exemple il est maintenant trivial de sécuriser la méthode `delete`