Gestion des langues

Je commencerai par modifier ma base de données. Il faudra créer des tables avec les différents pays et régions, les différentes langues et la « correspondance pays / langues / régions ». Pour chaque entité pour laquelle je dois traduire un libellé ou une description, je créerai une seconde table. Par exemple, « productEntity » contiendra les informations qui ne changeront pas selon la langue telle que la référence du produit, la catégorie à laquelle il est rattaché ou encore la date à laquelle il a été inséré dans la base de données. « productEntityLangue » contiendra les ID de la langue et du produit, le libellé, la description… Ces deux tables seront liées par une relation de type « OneToMany ».

Je modifierai ensuite la configuration de mon projet, et préciserai, dans le fichier « config.yml » quelle est la langue par défaut. J'activerai l'outil utilisé pour effectuer les traductions : « translator: { fallbacks: ['%locale%'] } ». Je créerai ensuite, pour chaque langue, un fichier dans lequel, pour chaque chaine de caractères présente sur le site, j'indiquerai sa traduction. Je repasserai sur tous mes templates pour m'assurer que j'ai bien utilisé les méthodes « trans » et « transChoice », implémentée dans Symfony. Je vérifierai également les fichiers PHP, notamment pour les « flashbag ». Je peux utiliser la commande «symfony i18n :extract frontend en --autosave » pour éviter de créer le catalogue à la main.

Je donnerai à l'utilisateur la possibilité de choisir une langue (une valeur par défaut lui sera attribuée grâce aux entête http) et je stockerai l'information dans un cookie. S'il est connecté, je la conserverai aussi dans son profil utilisateur.

Il faudra également que je m'assure qu'il n'y a aucun problème d'encodage, notamment pour les caractères spéciaux. Là encore, je peux modifier l'encodage du site en modifiant le fichier « settings.yml » et définir le paramètre « i18n » à « on ».

Gestion des devises

Pour les devises, il faudra créer des tables avec les différents pays, les différentes devises et la « correspondance pays / devise». je commencerai par localiser l'utilisateur (l'entête HTTP peut me fournir cette information). Je lui proposerai une devise en fonction de sa localisation. Je stockerai ces informations dans un cookie. S'il s'est enregistré, alors je peux lui demander directement sa provenance.

Dans le fichier de configuration, j'indiquerai un pays et une langue par défaut.

Je créerai, pour la table productEntity, une seconde table dans laquelle je stockerai, pour chaque devise, le prix de l'article (relation « OneToMany »). Je recalculerai plusieurs fois par jour les prix de chaque article en paramétrant une « Cron tab ».