**Compte rendu TD3 WebDynamique :**

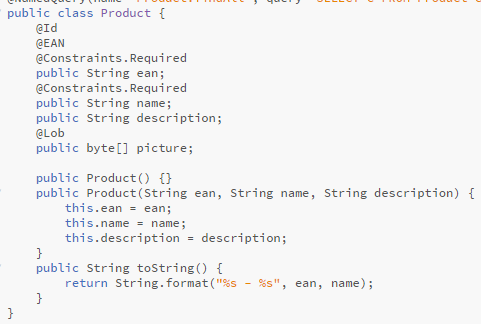
**1.1 Persistence des données**

JPA est le sigle pour « Java Persistance API », elles permettent de créer des relations entre des informations pour les applications utilisant Java, elles permettent également de stocker des données sur un serveur distant.

**1.2 Présentation des modèles (dans app/models)**

Les deux modèles sont « product » et « user » tout deux avec l’extension java.

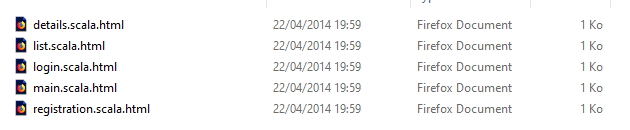
Il faut savoir déjà que le « at » (@) permet a scala de récupérer une donnée, ou de renvoyer a une autre chose. Le @ID est une variable contenant l’identifiant d’un produit. La classe Product quand a elle permet de définir de quoi a besoin un produit



Le dossier « validator » contient des codes validant ou invalidant (testant) la possibilité que ce que l’on entre ou scan soit un code barre.

Il faut voir les codes se trouvant dedans comme des vérificateurs de format de codes-barres

**1.3 Présentation de la vue (dans app/views)**



Le fichier details.scala.html renseigne les propriétés de l’objet, c’est-à-dire qu’il donne à un Product un code barre par exemple, et un nom (il détaille mon objet)

Pour comprendre la ligne Listing 1, il faut d’abord comprendre l’intérêt de la page dans tt son code



On voit dans la case rouge tout là-haut que je créer ici un tableau, dans la case bleue, au début et a la fin, on peut voir les balises <tr>. En Html lorsque l’on créer un tableau, on créer une colonne avec tr (table row), et td (on la voit plusieurs fois a l’intérieur des tr). Graphiquement ce qui en ressortira sera donc un tableau avec 3 cases

Maintenant la ligne, le @route.Products.details(product.ean) dit a scala de renvoyer la donnée « ean » d’un product qui nous as été donné par le fichier details

Juste avant de quitter (pour moi plus que la note) :

Le @ est un peu tricky a comprendre, il peut s’agir de tt un code contenu dans une ligne, ou de seulement une ligne