# Application "That's Life"

## 1 Introduction

La société « Life is Fun » souhaite lancer un site web surfant sur la vague de la publication des anecdotes de la vie quotidienne, tel que peut le faire le site www.viedemerde.fr.

« Life is Fun » fait donc appel à la société « Stemlaur Inc » pour réaliser cette application nommée « That's Life ». Voulant s'inscrire dans une démarche basée sur les standards, « Life is Fun » souhaite que l'application soit réalisée avec la technologie « Java EE 6 ». JPA, EJB, CDI, JAX-RS, JSF sont les mots clés de l'appel d'offre remporté par « Stemlaur Inc », et les technologies à mettre en oeuvre pour réaliser cette application.

Le but de l'application et de permettre à n'importe qui de poster une anecdote sur sa vie (avec un potentiel humoristique) pour permettre aux autres utilisateur de la consulter. Chaque anecdote pourra bien évidemment être commentée par n'importe quel utilisateur de l'application.

Si le TP se fait en groupes, il est possible de faire ce TP en mode Dojo. Cela signifie que chaque personne d'un groupe passe au clavier pour coder. Au bout de 10 minutes, c'est à la personne suivante de coder. Pour ceux que cela intéresse n'hésitez pas à prévenir Mathieu pour la gestion des « rounds ».

## 2 Préparation

Le TP se déroulera sous Linux avec l'IDE Netbeans et le serveur d'application GlassFish. Si ces outils ne sont pas installés, voyez avec les encadrant pour l'installation des applications.

Commencez par dézipper le dossier de sources fourni et ouvrez le projet qu'il contient dans Netbeans. Le projet contient la structure de base de l'application (squelette JSF, CSS, fichiers de configuration, etc …). L'application devrait pouvoir être déployée telle quelle sans aucun problème.

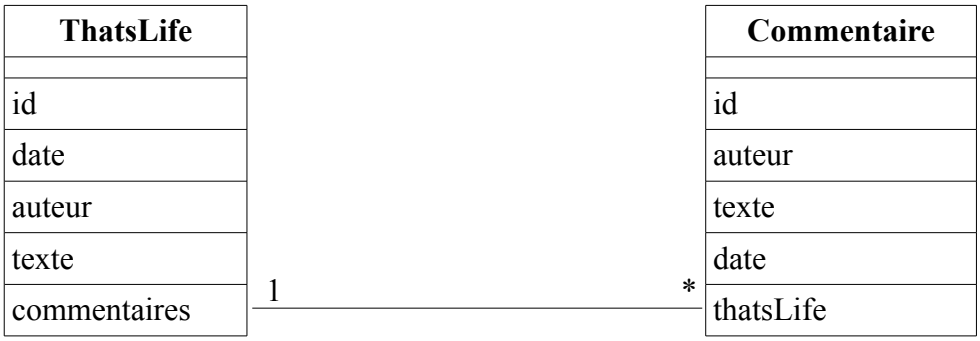
## 3 Entités

Le premier point à mettre en oeuvre pour cette application est la persistance des données en base de données. Celle-ci sera obtenue via l'utilisation de l'API JPA2.

Il va donc falloir créer des POJO (annotés avec **@Entity**) représentant les entités à persister en base suivant le modèle de données décrit plus bas. Je conseille d'utiliser l'assistant Netbeans dédié aux entités JPA pour cela (clic droit sur le projet, new, Entity Class).

**N'oubliez pas les getters et les setters sur toutes les propriétés que vous rajoutez aux entités.**

Vous trouverez également dans le projet une classe nommée Bootstrap.java. Cette classe est un EJB session qui est créé immédiatement au démarrage de l'application (via l'annotation **@Startup**). Vous allez pouvoir vous servir de cette classe pour remplir manuellement la base de données avec des entités de test. Pour ce faire créez des instances d'entités dans la méthode init() et persistez les grâce à l'entity manager présent dans l'EJB ( `em.persist(myComment);` ). Vous pouvez vérifier si les entités sont bien présentes en base grâce à l'explorateur de base de données de Netbeans (voir avec Mathieu).




# 4 Services

Maintenant que les entités sont fonctionnelles, il est nécessaire de créer certaines méthodes permettant de les créer, les supprimer, les trouver toutes ou via leur clé primaire, etc … en bref, toutes les méthodes permettant de consulter les données en base.

Pour ce faire créez un ou plusieurs EJB locales (suivant les responsabilités que vous souhaitez leur donner) qui manipuleront les diverses entités.

Il sera surement nécessaire de faire des requêtes, pour trouver toutes les entités d'un même type par exemple. Plusieurs solutions s'offrent à vous, les requêtes en mode texte directement, les NamedQueries et les Criteria Queries. Il peut-être intéressant d'expérimenter les 3 types.

- entityManager.createQuery(query)
- entityManager.createNamedQuery(nomDeLaNamedQuery)
- pour les critéria, n'hésitez pas a consulter la documentation Java EE 6 ;-)

# 5 Consultation REST

Pour permettre de tester rapidement et simplement vos services de consultation, vous pouvez transformer vos EJBs en services REST accessibles

via un simple client HTTP.

Pour ce faire, commencez par créer une classe telle que :

```
@ApplicationPath("resources")
public class RestApplication extends Application {}
```

Cette classe va permettre d'activer JAX-RS pour l'appel de services REST via HTTP. Il suffit ensuite d'annoter votre EJB par **@Path**("quelquechose") et la méthode (de consultation) à tester par **@GET** et **@Produces** comme suivant :

```
@Stateless
@Path("comments")
public class CommentBean {
	@GET
	@Produces(MediaType.APPLICATION_XML)
	public Collection<Comment> findAll() {
		….
	}
}
```

Pour consulter cette méthode via appel REST, ouvrez un terminal et lancez la commande :

wget -qO- http://localhost:8080/ThatsLife/services/comments

ou

curl http://localhost:8080/ThatsLife/services/comments

# 6 JSF

La partie web de l'application est réalisée en JSF 2. Plusieurs vues doivent être réalisées.

## *6.1 vue de la liste*

En prenant exemple sur le managed bean contrôleur présent dans le projet (ApplicationController.java), faites en sorte d'afficher toutes les entités « ThatsLife » sur la page principale.

La récupération des données se fera dans la méthode doNavigate (via vos services de consultation). Pour l'affichage, les tags **<h:dataTable>** et **<h:column>** vous seront utiles pour le traitement d'une collection. Pour un affichage textuel basique, **<h:outputText>**.

### *6.2 Ajout d'une anecdote*

Ajoutez un formulaire en faut de la page par défaut pour poster une nouvelle anecdote. Le formulaire sera constitué d'un champ pour le nom de l'auteur et d'un espace de texte pour l'anecdote.

Les tags à utiliser sont **<h:form>** pour créer un formulaire, **<h:inputText>** et **<h:inputTextArea>** pour les champs de texte et **<h:commandButton>** pour le bouton de soumission du formulaire.

N'oubliez pas d'ajouter des attributs a votre managed bean contrôleur correspondant aux informations du formulaire. **Voir l'exemple au tableau**

### *6.3 vue d'une anecdote*

Créez une vue spécifique permettant de visualiser une anecdote ainsi que ces commentaires. Il sera surement nécessaire de créer un nouveau managed bean.

Pour créer les liens, utilisez le tag **<h:link>** dans une balise **<h:form>**. Il est également possible de passer des paramétres via les liens en utilisant le tag **<f:param>** à l'intérieur de **<h:link>**. **Voir l'exemple au tableau pour la récupération des paramètres dans le contrôleur.**

### *6.4 ajout d'un commentaire*

Créez un formulaire dans la vue d'une anecdote pour pouvoir poster un commentaire sur cette anecdote. Le formulaire comporte un champ auteur et un champ commentaire.

### *6.5 suppression d'une anecdote*

Ajoutez un lien permettant de supprimer une anecdote (normalement disponible en mode admin).

## *7* Bonus

### *7.1 Gestion de catégories*

Ajouter une notion de catégorie à votre modèle de données. Affichez les catégories dans une liste sur la droite du site (fichier side.xhtml). La navigation sur une catégorie affiche toutes les anecdotes de cette catégorie.

### *7.2 Workflow de modération*

Ajoutez une vue (normalement disponible en mode admin) permettant à un modérateur du site de valider une anecdote avant qu'elle ne soit réellement publiée. Modifiez votre modèle de données en conséquence.

### *7.3 Votes*

Ajoutez des liens sur les anecdotes permettant de voter pour ou contre cette anecdote. Les totaux de pour et de contre seront affichés sur chaque anecdote. Modifiez votre modèle de données en conséquence.