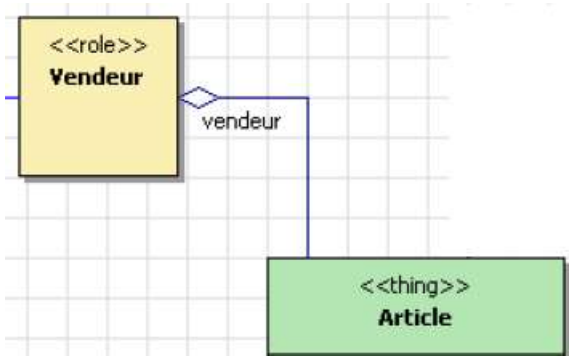
[T03] Méthodologies Agiles / J2EE et webServices
Projet ezBay
Prototype technique

1. Développement

1.1. Partie Serveur

Nous nous sommes concentrés sur les classes Vendeur et Article.



- 2 CMP Entity Bean :

  - ArticleBean

    Attributs :

```
private String id;
private String libelle;
private String marque;
private String modele;
private Double prixVente;
private Integer anneeFabrication;
private java.lang.String description;
private Date dateLimite;
```

    Méthodes :

```
Accesseurs en lecture et écriture
public abstract VendeurLocal getVendeurLocal();
public abstract void setVendeurLocal(VendeurLocal vendeurLocal);
public abstract ArticleDTO getArticleDTO();
public void updateArticle(ArticleDTO articleDTO)
```

    Finder (EJB-QL) :

```
java.util.Collection findAll()
java.util.Collection findByLibelle(java.lang.String libelle)
java.util.Collection findByMarque(java.lang.String marque)
java.util.Collection findByModele(java.lang.String modele)
java.util.Collection findByPrixVente(java.lang.Double prixVente)
java.util.Collection findByAnneeFabrication(java.lang.Integer anneeFabrication)
java.util.Collection findByDateLimite(java.util.Date dateLimite)
java.util.Collection findByDescription(java.lang.String description)
```

  - VendeurBean

    Attributs :

```
private String id;
private String nom;
```

    Méthodes :

```
Accesseurs en lecture et écriture
public abstract Collection getArticle();
public abstract void setArticle(Collection article);
public abstract VendeurDTO getVendeurDTO();
public void updateVendeur(VendeurDTO vendeurDTO) ;
```

    Finder (EJB-QL) :

```
java.util.Collection findAll()
java.util.Collection findByNom(java.lang.String nom)
```

  Création des DTO : ArticleDTO et VendeurDTO

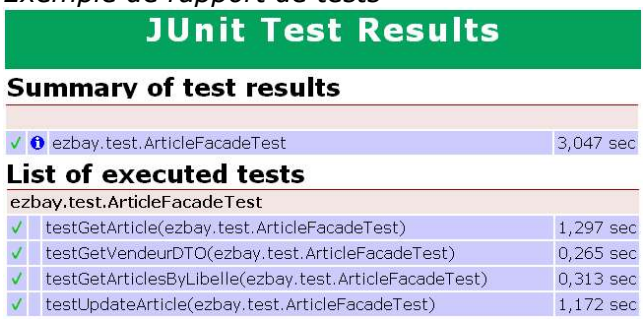
- 2 EJB Stateless Session Facade :
  - ArticleFacadeBean
  - VendeurFacadeBean

- 1 relation CMR entre les EJB Entity
  - Article-Vendeur (bidirectionnelle) : à un VendeurBean sont associés plusieurs ArticleBean.

1.2. Partie Client

Des recherches sur Internet (et en toute franchise, des conseils avisés d'un autre groupe), nous ont permis de comprendre que l'API JUnit n'était pas utilisable dans le cadre des EJB. C'est pourquoi nous avons utilisé JUnitEE pour les tests de nos Beans à travers les interfaces locales (pour les Entity), et les interfaces distantes (pour les Session).

Exemple de rapport de tests



JUnit Test Results

Summary of test results

✓	ezbay.test.ArticleFacadeTest	3,047 sec

List of executed tests

ezbay.test.ArticleFacadeTest		
✓	testGetArticle(ezbay.test.ArticleFacadeTest)	1,297 sec
✓	testGetVendeurDTO(ezbay.test.ArticleFacadeTest)	0,265 sec
✓	testGetArticlesByLibelle(ezbay.test.ArticleFacadeTest)	0,313 sec
✓	testUpdateArticle(ezbay.test.ArticleFacadeTest)	1,172 sec

ArticleFacadeTest.java et VendeurFacadeTest.java qui comme leur nom l'indique sont des classes de test pour Article et Vendeur.

2. Difficultés rencontrées

La construction de la liaison CMR entre Article et Vendeur nous a posé quelques problèmes, pour les raisons suivantes :

- difficultés de compréhension de la gestion CMR par le « container » : Que fait-il ? Que ne fait-il pas ?...

- méthode de programmation : l'utilisation des tags xDoclet ainsi que la structure imposée par les EJB imposent une implémentation assez lointaine de la programmation Java telle que nous la connaissions auparavant.