



BUT 3

TP Dev Avancé #2

Modernisation d'une application Web Java EE avec JPA / Hibernate.

Contexte pédagogique :

L'application MasterAnnonce, développée précédemment à l'aide de Servlets, JSP et JDBC, va être refactorisée et modernisée afin d'introduire une approche plus robuste, plus maintenable et plus proche des standards industriels actuels.

Objectifs pédagogiques :

À l'issue de ce TP, l'étudiant doit être capable de :

- Remplacer une couche JDBC par **JPA/Hibernate**
- Concevoir un **modèle de données relationnel** et le mapper avec JPA
- Structurer une application Web en **couches cohérentes**
- Gérer correctement les **transactions**
- Comprendre et corriger les **problèmes classiques JPA**
- Mettre en place des **tests unitaires et d'intégration**
- Justifier des choix techniques

Prérequis :

- TP AIR #1 validé
- HTML / CSS
- Java / Java EE (Servlets, JSP)
- SQL (PostgreSQL)
- Maven

Nouvelles notions abordées :

- JPA (Jakarta Persistence / javax.persistence)
- Hibernate ORM
- Mapping objet–relationnel (ORM)
- EntityManager, EntityManagerFactory
- Transactions JPA
- JPQL
- Relations JPA (@ManyToOne, @OneToMany)
- Lazy loading / Fetch strategies
- Bean Validation
- Architecture en couches (Web / Service / Persistence)
- Tests unitaires et d'intégration
- Mockito (bonus)

Modèle de données cible

Table User

- id (Long, clé primaire)
- username (String, unique)
- email (String, unique)
- password (String)
- createdAt (Timestamp)

Table Category

- id (Long, clé primaire)
- label (String, unique)

Table Annonce

- id (Long, clé primaire)
- title (String, 64 caractères)
- description (String, 256 caractères)
- adress (String, 64 caractères)
- mail (String, 64 caractères)
- date (Timestamp)
- status (ENUM : DRAFT, PUBLISHED, ARCHIVED)
- author (relation vers User)
- category (relation vers Category)

Relations attendues

- Un utilisateur peut publier plusieurs annonces
- Une catégorie peut contenir plusieurs annonces
- Une annonce appartient à un seul utilisateur et une seule catégori


Exercice 1 – Mise en place de JPA / Hibernate

1. Ajouter les dépendances nécessaires dans le pom.xml
2. Créer le fichier persistence.xml
3. Configurer la connexion à PostgreSQL
4. Mettre en place une classe utilitaire permettant d'obtenir un EntityManager

 **Livable** : application démarrable avec JPA opérationnel.

Exercice 2 – Mapping des entités JPA

1. Créer les entités User, Category, Annonce
2. Définir les annotations JPA appropriées
3. Mettre en place les relations entre entités
4. Ajouter des contraintes de validation (Bean Validation)

 **Livable** : schéma de base de données généré ou validé par Hibernate.

Exercice 3 – Couche Repository (DAO JPA)

1. Implémenter les classes suivantes :
 - a. AnnonceRepository
 - b. UserRepository
 - c. CategoryRepository
2. Fonctionnalités attendues :
 - a. CRUD complet
 - b. Recherche par mot-clé
 - c. Filtrage par catégorie et statut
 - d. Pagination des résultats

Contraintes

- Utilisation exclusive de JPQL
- Pas de JDBC

Exercice 4 – Couche Service & Transactions

1. Créer un service métier AnnonceService gérant :
 - a. Création d'annonce
 - b. Modification
 - c. Publication
 - d. Archivage
 - e. Suppression
 - f. Recherche et listing paginé

Important

- Les transactions doivent être gérées dans la couche service
- Aucune transaction ne doit être ouverte dans les Servlets

Exercice 5 – Refonte Web (Servlets & JSP)

1. Authentification
 - a. **Création d'un système de login simple (session)**
 - b. **Mise en place d'un filtre de sécurité**
2. Fonctionnalités Web
 - a. Liste paginée des annonces
 - b. Formulaire de création
 - c. Formulaire de modification
 - d. Détail d'une annonce
 - e. Actions Publish / Archive selon le statut

Exercice 6 – Validation et gestion des erreurs

1. Validation serveur des formulaires
2. Affichage des messages d'erreur dans les JSP
3. Conservation des valeurs saisies en cas d'erreur

 **Un court paragraphe expliquant chaque problème rencontré devra être fourni dans le README.**

★ Bonus – Tests unitaires et d'intégration (pour les plus rapides)

1. Niveau 1 – Tests Repository (intégration)
 - a. Tests CRUD avec base réelle
 - b. Tests de recherche et pagination
2. Niveau 2 – Tests Service (unitaires)
 - a. Mockito pour mocker les repositories
 - b. Tests des règles métier
 - c. Vérification des appels et des états
3. Niveau 3 – Tests d'intégration métier
 - a. Enchaînement complet : création → publication → recherche
 - b. Tests reproduisant un problème Lazy / N+1
4. Niveau 4 – Tests Web
 - a. Tests de Servlets avec mocks HTTP
 - b. Test du filtre d'authentification

Livrables attendus

1. Projet Maven complet
2. README expliquant :
 - architecture
 - problèmes rencontrés
 - solutions apportées
3. Scripts SQL éventuels
4. Tests automatisés

Critères d'évaluation

1. Respect des consignes
2. Qualité du mapping JPA
3. Bonne gestion des transactions
4. Compréhension des pièges JPA
5. Clarté du code et du README
6. Bonus tests (valorisés)
7. Le nombre de café que vous m'apporterez durant la journée
 - Seront accepté tout accompagnement pour le café