

# Briot Loïck – Rapport de TD n°2

## Sur la JPA && EJB



### Table des matières

I.	Manuel d'utilisation .....	2
A.	Structure du programme .....	2
B.	Utilisation de l'application .....	3
II.	Réponses aux questions.....	6
A.	En quoi, le type de Session Bean utilisé dans le projet est adapté aux besoins du projet ?.....	6
B.	Quelle est l'utilité de proposer l'interface remote dans un fichier jar séparé de l'application EE ? .....	6
C.	Si vous devriez exporter votre projet vers un serveur EE dans une machine distante, quels fichiers devraient être téléchargés sur la machine distante ? .....	6
D.	Est-ce possible d'exécuter l'application depuis une machine différente de celle où se trouve l'application EE ? Quelles sont les caractéristiques de l'application (en termes de type de EJB et ressources utilisées pour la persistance) qui motivent votre réponse ?.....	6

## I. MANUEL D'UTILISATION

---

### A. Structure du programme

Le projet contient au total 8 classes java :

- main.java qui permet d'essayer et de tester les méthodes des autres classes sans l'utilisation de Vaadin pour l'aspect graphique.
- Student.java qui implémente l'objet de type Student.
- StudentEJB.java qui contient toutes les méthodes du conteneur relatives à l'entité Student.
- StudentEJBRemote.java qui est l'interface implémentée par StudentEJB.java
- Project.java qui implémente l'objet de type Project.
- ProjectEJB.java qui contient toutes les méthodes du conteneur relatives à l'entité Project.
- ProjectEJBRemote.java qui est l'interface implémentée par ProjectEJB.java
- Td\_jpaUI.java qui lance le programme avec l'interface graphique générée par le Framework Vaadin. C'est cette classe qu'il faut lancer pour lancer l'application sur le serveur avec l'interface graphique montrée ci-dessous.

The screenshot displays a web application interface with two main sections for managing 'Student' and 'Project' entities. Each section includes input fields for attributes (name, address, age for Student; id project, title, owner for Project) and a set of buttons for CRUD operations (Create, Update, Delete, Find all, Find by Id). The 'Find Projects by ID Student' button is located at the bottom of the Student section.

Student Section	Project Section
name: <input type="text"/>	id project: <input type="text"/>
address: <input type="text"/>	title: <input type="text"/>
age: <input type="text"/>	owner: <input type="text"/>
Buttons: Find all Students, Find a Student by Id, Delete a Student, Create a Student, Update a Student	Buttons: Find all Projects, Find a Project by Id, Delete a Project, Create a Project, Update a Project
Find Projects by ID Student	

Le projet contient également 2 fichiers .xml :

- web.com qui gère les paramètres des pages web et les dépendances entre elles (pas utilisé ou très peu dans ce projet)
- persistance.xml qui gère toute la relation entre la base de données et le programme grâce à l'utilisation de la JPA (Java Persistence Application) et contient donc tous les paramètres nécessaires pour se connecter à ma base de données. J'ai choisi d'utiliser EclipseLinks et non Hibernate pour gérer le dialogue avec ma base de données.

## B. Utilisation de l'application

Lorsque l'on lance l'application, l'écran ci-dessous est généré.

The interface is divided into two main sections: Student and Project. Each section has input fields for name, address, age, and ID, along with buttons for 'Find all', 'Find by ID', 'Delete', 'Update', and 'Create'. The 'Find all Students' button is highlighted.

Il est alors facile de créer un étudiant dans la base de données comme on peut le voir ci-dessous :

The 'Create a Student' button is highlighted with a blue border. Below it, a message states 'A student has been created'. The other input fields and buttons remain visible.

On peut regarder la liste de tous les étudiants stockés dans la base de données :

The 'Find all Students' button is highlighted with a blue border. Below it, a list of all students is displayed, including their ID, name, address, and age.

ID	name	address	age
1401	Loick	2	22
1451	Loick	133 clos	22
1452	dede	22 boulevard	12
1501	Dupont	13 boulevard des Aiguillette	19

On peut également rechercher, effacer ou mettre à jour un étudiant à partir de son ID, comme il est illustré ci-dessous :

The screenshot shows a web application interface for managing students and projects. The 'Find a Student by Id' button is highlighted. The interface includes fields for name, address, age, id student, id project, and title. It also has buttons for 'Find all Students', 'Find all Projects', 'Find a Project by Id', 'Delete a Student', 'Delete a Project', 'Create a Student', 'Update a Student', 'Update a Project', and 'Create a Project'. A message 'A student has been created' is visible.

La même façon, on peut facilement créer un projet et observer tous les projets créés.

The screenshot shows a web application interface for managing projects. The 'Find all Projects' button is highlighted. The interface includes fields for name, address, age, id student, id project, and title. It also has buttons for 'Find all Students', 'Find a Student by Id', 'Delete a Student', 'Update a Student', 'Find Projects by ID Student', 'Find all Projects', 'Find a Project by Id', 'Delete a Project', 'Update a Project', and 'Create a Project'. A message 'A project has been created' is visible.

On peut bien évidemment observer tous les projets pour un étudiant donné à partir de son ID :

The screenshot shows a web application interface for managing projects for a specific student. The 'Find Projects by ID Student' button is highlighted. The interface includes fields for name, address, age, id student, id project, and title. It also has buttons for 'Find all Students', 'Find a Student by Id', 'Delete a Student', 'Update a Student', 'Find all Projects', 'Find a Project by Id', 'Delete a Project', 'Update a Project', and 'Create a Project'. A message 'A project has been created' is visible.

Des sécurité ont été prises pour ne pas avoir de données erronées dans la base. Ainsi, un projet ne être créé si l'étudiant qui lui est rattaché n'existe pas.

The screenshot shows a web application interface with two main sections: 'Student Management' and 'Project Management'. In the 'Student Management' section, the 'id student' field is set to '119'. In the 'Project Management' section, the 'id project' field is empty, and the 'title' field is 'TD\_Database'. The 'owner' field is set to '119'. The 'Create a Project' button is highlighted with a blue border. Below it, an error message reads: 'ERROR: This student doesn't exist'.

De la même façon, lorsque l'on efface un étudiant, on prend garde que l'ensemble de ses projets soient effacés également.

The screenshot shows the same web application interface. In the 'Student Management' section, the 'id student' field is set to '1501'. The 'Delete a Student' button is highlighted with a blue border. In the 'Project Management' section, the 'Find all Projects' button is also highlighted with a blue border. The 'All Projects' list shows a project with ID 1502, title 'TD\_JPA', and owner '1501'.

Résultat après l'effacement de l'étudiant qui a pour ID 1501. On constate que son projet a également été effacé.

The screenshot shows the same web application interface. In the 'Student Management' section, the 'id student' field is set to '1501'. The 'Find all Projects' button is highlighted with a blue border. The 'All Projects' list is now empty, indicating that the project associated with the deleted student has been removed. A message at the bottom of the 'Student Management' section reads: 'A student has been deleted'.

## II. REPONSES AUX QUESTIONS

---

**A. En quoi, le type de Session Bean utilisé dans le projet est adapté aux besoins du projet ?**

Un Stateless Session Bean suffit dans ce projet puisque quand on invoque une des méthodes de StudentEJB (create, update, find, delete...), on n'a pas besoin des états passés correspondant aux différents échanges avec le même client.

**B. Quelle est l'utilité de proposer l'interface remote dans un fichier jar séparé de l'application EE ?**

Le fichier jar séparé de l'application permet d'accéder à l'EJB depuis un client qui n'est pas contenu dans la Java Virtual Machine.

**C. Si vous devriez exporter votre projet vers un serveur EE dans une machine distante, quels fichiers devraient être téléchargés sur la machine distante ?**

On peut tout emballer dans une archive Web (WAR) puisque notre projet est de faible envergure. Cependant, séparer les EJB des autres classes permettrait de le faire plus proprement. On dépose ensuite l'archive WAR dans la machine distante au niveau de le répertoire webapps.

**D. Est-ce possible d'exécuter l'application depuis une machine différente de celle où se trouve l'application EE ? Quelles sont les caractéristiques de l'application (en termes de type de EJB et ressources utilisées pour la persistance) qui motivent votre réponse ?**

Les données étant stockées dans une base de données, il est possible d'exécuter l'application depuis une machine différente de celle où se trouve l'application EE.