#### Briot Loïck – Rapport de TD n°2

#### Sur la <u>JPA && EJB</u>



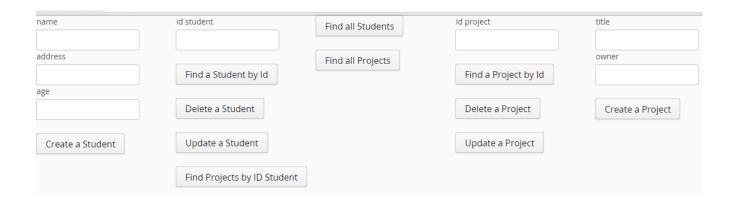
#### Table des matières

I.	Ma	nuel d'utilisation	. 2
	A.	Structure du programme	. 2
	B.	Utilisation de l'application	. 3
II	R	Réponses aux questions	6
11		•	. 0
	A. projet	En quoi, le type de Session Bean utilisé dans le projet est adapté aux besoins du	. 6
	В.	Quelle est l'utilité de proposer l'interface remote dans un fichier jar séparé de ication EE ?	
	C. quels	Si vous devriez exporter votre projet vers un serveur EE dans une machine distante fichiers devraient être téléchargés sur la machine distante ?	
		Est-ce possible d'exécuter l'application depuis une machine différente de celle où se l'application EE ? Quelles sont les caractéristiques de l'application (en termes de le EJB et ressources utilisées pour la persistance) qui motivent votre réponse ?	

#### A. <u>Structure du programme</u>

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.

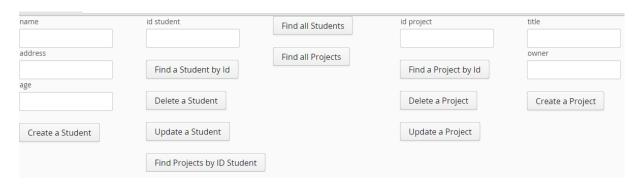


Le projet contient également 2 fichiers .xml :

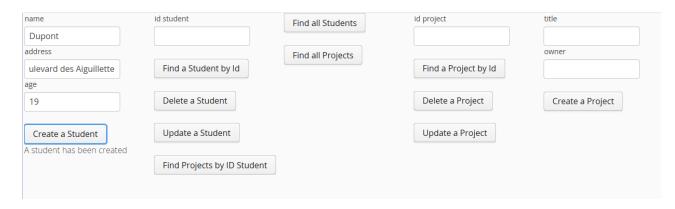
- web.com qui gère les paramètres des pages web et les dépendances entre elles (pas utilisé ou très peu dans ce projet)
- persistance.xml qui gère toute la relation entre la base de données et le programme grâce à l'utilisation de la JPA (Java Persistance Application) et contient donc tous les paramètres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

#### B. <u>Utilisation de l'application</u>

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



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



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

Dupont	id student	Find all Students	id project	title
address		Find all Projects		owner
ulevard des Aiguillette	Find a Student by Id	All Students:	Find a Project by Id	
age		ID: 1401   name: Loick   address:		
19	Delete a Student	2   age: 22 ID: 1451   name: Loick   address: 133 clos   age: 22	Delete a Project	Create a Project
Create a Student	Update a Student	ID: 1452   name: dede   address:	Update a Project	
A student has been created		22 boulevard   age: 12		
	Find Projects by ID Student	ID: 1501   name: Dupont   address: 13 boulevard des Aiguillette   age: 19		

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

name	id student	Find all Students	id project	title
Dupont	1501			
address		Find all Projects		owner
ulevard des Aiguillette	Find a Student by ld	All Students:	Find a Project by Id	
age		ID: 1401   name: Loick   address:		
19	Delete a Student	2   age: 22	Delete a Project	Create a Project
		ID: 1451   name: Loick   address:		
Create a Student	Update a Student	133 clos   age: 22 ID: 1452   name: dede   address:	Update a Project	
A student has been created		22 boulevard   age: 12		
	Find Projects by ID Student	ID: 1501   name: Dupont		
	ID: 1501   name: Dupont	address: 13 boulevard des		
	address: 13 boulevard des	Aiguillette   age: 19		
	Aiguillette   age: 19			

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

name	id student	Find all Students	id project	title
				TD_JPA
address		Find all Projects		owner
	Find a Student by ld	All Projects :	Find a Project by Id	1501
age		ID: 1502   title: TD_JPA   owner :		
	Delete a Student	1501	Delete a Project	Create a Project
				A project has been created
Create a Student	Update a Student		Update a Project	
	Find Projects by ID Student			

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

name	id student	Find all Students	id project	title
	1501			TD_JPA
address		Find all Projects		owner
	Find a Student by ld	All Projects :	Find a Project by ld	1501
age		ID: 1502   title: TD_JPA   owner	:	
	Delete a Student	1501	Delete a Project	Create a Project
				A project has been created
Create a Student	Update a Student		Update a Project	
	Find Projects by ID Student Projects of student n° 1501 ID: 1502   title: TD_JPA   owner: 1501			

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

name	id student	Find all Students	id project	title TD_Database
address		Find all Projects		owner
	Find a Student by ld		Find a Project by ld	119
age				
	Delete a Student		Delete a Project	Create a Project
				ERROR: This student doesn't exist
Create a Student	Update a Student		Update a Project	
	Find Projects by ID Student			

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



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

name	id student 1501	Find all Students	id project	title
address		Find all Projects		owner
	Find a Student by ld	All Students:	Find a Project by Id	
age		ID: 1501   name: Dupont		
	Delete a Student	address: 13 boulevard des	Delete a Project	Create a Project
		Alguillette   age: 19 All Projects :		
Create a Student	Update a Student	ID: 1502   title: TD_IPA   owner :	Update a Project	
		1501		
	Find Projects by ID Student	All Students:		
	A student has been deleted	All Projects :		

### 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. <u>Si vous devriez exporter votre projet vers un serveur EE dans une machine distante, quels fichiers devraient être téléchargés sur la machine distante?</u>

On peut tout empaqueter 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. <u>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 ?</u>

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.