EJB Entité

La création d'un EJB entité se résume à la création d'une classe telle que nous le ferions dans une application Java SE, en ajoutant des annotations.

```
@Entity
@Table(name="client")
@SequenceGenerator(
                    name = "client_sequence",
                    sequenceName = "client_id_seq"
)

public class Client implements Serializable{
        ....
}
```

`@Entity` : défini la classe comme une entité.
`@Table` : défini le nom de la table qui contiendra les informations des clients dans la base de données.
`@SequenceGenerator` : défini un générateur de séquence qui sera utilisé pour générer les clés primaires de la table.

Il est possible de spécifier le champ qui fait office de clef primaire et de spécifier une séquence utilisée.

```
@Id
@GeneratedValue(  strategy = GenerationType.SEQUENCE, generator="client_seq")
private Integer identifiant;
```

EntityManager

Afin de pouvoir manipuler notre entité Client, il faut utiliser l'interface EntityManager.

L'EntityManager permet d'effectuer des opérations d'accès aux données (sauvegarde, modification, suppression).

Il existe plusieurs manières d'obtenir un objet EntityManager qui ont chacune des spécificités quant à leur utilisation :

- Injection par annotation : utilisée exclusivement en environnement Java EE. Par exemple depuis un bean session avec l'annotation `@PersistenceContext`

- Utilisation de la fabrique `EntityManagerFactory` : utilisée pour gérer de façon manuelle la création d'EntityManager.

EJB Entité

Quelques exemples de manipulation de données à l'aide de l'EntityManager :

Enregistrement de données :

```
Client client = new Client(null,"Dubois","paul");
entityManager.persist(client);
```

Modification

```
entityManager.merge(client);
```

Suppression

```
entityManager.remove(client);
```

Requêtes EJB-QL.

```
String query = "SELECT c FROM Client c where c.nom=:nomRecherche ";
List liste = entityManager.createQuery(query)
        .setParameter(":nomRecherche", "Dubois")
        .getResultList();
```