

# TPB4 : PERSISTANCE DES DONNÉES AVEC DB4O

## Objectif du TP :

- créer la persistance des données (ici des objets manipulés)
- Utiliser une BDDO comme **DB4O**.
- visualiser cette BDDO avec **OME** (ObjectManager Enterprise) avec Eclipse

## QU'EST-CE QUE LA PERSISTANCE DES OBJETS ?

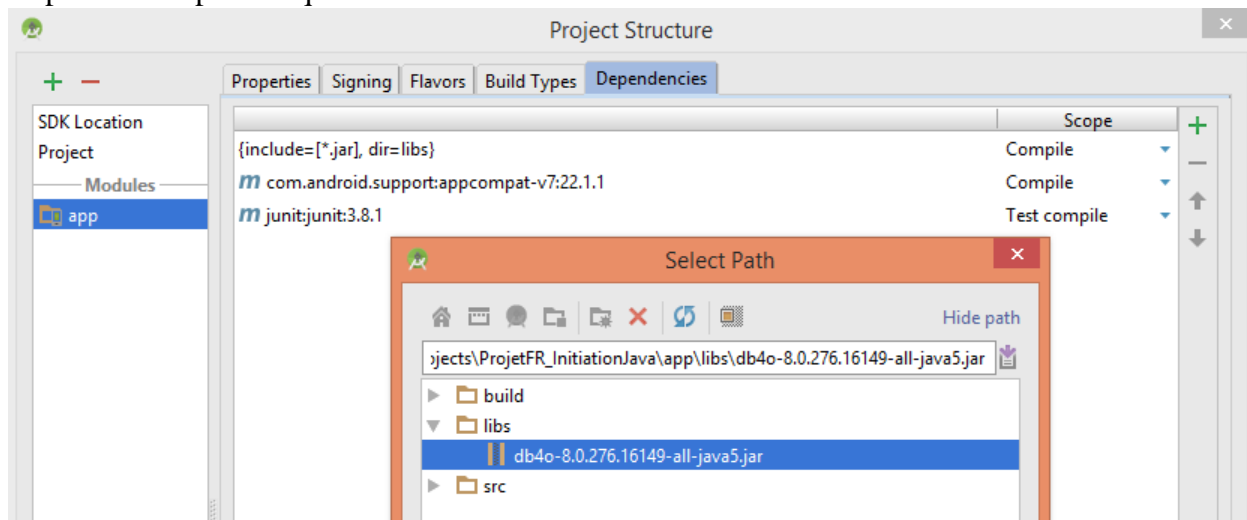
La persistance est la sauvegarde de *données* soit dans un fichier, soit dans une BDD relationnelle ou Objet. Ici, comme nous programmons en Objet, le plus logique est de choisir une BDD Objet comme DB4O.

## MISE EN PLACE D'UNE BASE DE DONNÉES OBJET (AFIN D'ASSURER LA PERSISTANCE DES DONNÉES)

## AJOUTER LA LIBRAIRIE POUR IMPLANTER UNE BD OBJET SUR VOTRE MOBILE

Dans votre projet, il faut ajouter cette nouvelle librairie pour pouvoir utiliser de la sauvegarde dans une BDD Objet.

- **Copier** dans le dossier libs de votre projet, le fichier db4o db4o-8.0.276.16149-all-java5.jar (fourni dans Partage)
  - Ajouter cette librairie dans les dépendances de votre projet : Aller Project Structure, onglet Dependencies
- Dependencies puis Cliquer sur + et faites OK!



Attention pour pouvoir écrire sur la carte SD de votre téléphone ou tablette vous devez rajouter, dans le fichier AndroidManifest.xml de votre projet, la permission d'écriture :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.frederiquederobien.projetfr_initiationjava" >

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
```

## CLASSE MODELE

- Créez la class Modele pour assurer la persistance des données

Le projet tente de respecter l'approche MVC. (Rappel : dans Android une Activity = une vue + un contrôleur)

Attributs :

```
private String db4oFileName;
private ObjectContainer dataBase;
private File appDir;
```

- Ajoutez la méthode open()

```
public void open() {
    db4oFileName = Environment.getExternalStorageDirectory() +
"/baseDB4o" + "/BaseCours.db4o";
    Db4oEmbedded.openFile(Db4oEmbedded.newConfiguration(), db4oFileName);
}
```

- Ajoutez la méthode createDirectory()

Cette méthode est utile pour stocker la BDD dans un dossier accessible.

```
public void createDirectory() {
    appDir = new File(Environment.getExternalStorageDirectory()
+ "/baseDB4o");

    if (!appDir.exists() && !appDir.isDirectory()) {
        appDir.mkdirs();
    }
}
```

- Ajoutez le constructeur Modele()

```
public Modele() {
    createDirectory();
    //Plus tard, on fera l'import de donnees si on les recupere d'une
autre BDD (MySQL)
}
```

Exemple d'utilisation du SGBD DB4o (cf db4o-8.0-java\db4o-8.0\doc\tutorial)

Operation	SQL	HTTP	DB4O
Create	INSERT	POST	.store
Read (Retrieve)	SELECT	GET	.queryByExample .AsQueryable
Update	UPDATE	PUT / PATCH	.store
Delete	DELETE	DELETE	.delete

- Pour trouver tous les objets d'une classe (exemple Cours) :  

```
ObjectSet<Cours> result = dataBase.queryByExample(Cours.class);
while (result.hasNext()) {
    // traitement à faire en récupérant l'objet en cours
    // result.next();
}
```
- Pour trouver un objet d'une classe (exemple Cours) ou tous les objets d'une classe avec une spécificité (ici le Cours ayant l'identifiant id) :

```
Cours p = new Cours();
p.setIdentifiant(id);
ObjectSet<Cours> result = dataBase.queryByExample(p);
p= (Cours) result.next();
```

Remarques : S'il y a plusieurs Cours, utilisez un while et une ArrayList.

- Pour sauvegarder un objet d'une classe (exemple Cours, objet **unCours**)
 

```
open();
dataBase.store(unCours);
dataBase.close();
```

#### ⚠Attention :

L'insertion et la mise à jour des objets d'une classe utilisent le même ordre : **store**.

DB4o utilise ses propres pointeurs pour différencier une mise à jour, d'une création.

Le contexte STA de notre projet impose que toutes modifications de la base de données s'accompagnent d'un open et d'un close de la base de données, ce pour assurer la persistance des données.

La fermeture de la base de données s'accompagne de la perte des pointeurs, et impose pour une mise à jour d'objet, une recherche préalable de cet objet puis une mise à jour.

#### ➤ Ajoutez la méthode SaveCours (Cours **unCours**)

- sauve le Cours passé en paramètre

```
public void saveCours(Cours unCours) {
    //méthode pour SAUVEGARDER un nouveau cours et
    // ATTENTION pour le MODIFIER en cas de modification (exemple lors de l'ajout de
    participants)
    // Il faut récupérer le cours au préalable en appelant la méthode recupereCours
    try {
        open();
        dataBase.store(unCours);
        dataBase.commit();// VALIDATION de cet ajout d'objet car le store s'est bien
        passé
    } catch (Exception e) {
        e.printStackTrace();
        dataBase.rollback();// ANNULATION de cet ajout d'objet car le store a "buggé"
    } finally {
        dataBase.close();// dans tous les cas FERMETURE de la BDDO
    }
}
```

⚠CELA MARCHE si l'on ne modifie jamais le cours, or lors de l'ajout des participants (dans l'activity ActivityParticipant, on modifie bien le cours car on ajoute des participants dans sa liste): On verra cela plus tard !

**Attention :** si ce Cours existe déjà dans la BDDO, il faut le récupérer par un identifiant (ici l'intitulé du cours UNIQUE) pour le mettre à jour.

#### ➤ Ajoutez la méthode ListeCours ()

- permet de lister tous les cours présent dans la BDDO

```
public ArrayList<Cours> listeCours(){
    //créer la liste de retour
    open();
    ObjectSet<Cours> result = dataBase.queryByExample(Cours.class);
    //parcourir tous les éléments du résultat de la requête et ajouter chaque élément à la
    nouvelle liste
    //fermer la BDDO
    //retourner la liste
}
```

## TEST DB4O & INSTALLATION DE OME MANAGER AVEC ECLIPSE

Nous allons installer un Plugin nommé **OME (Object Manager Enterprise)** qui va nous permettre de **VISUALISER** les objets stockés en BDO : Vous pouvez visualiser l'ensemble des **classes gérées** et, pour chacune, les **instances d'objets**

Cela a juste un but de vérifier les données stockées. Cet outil sert au développeur pour vérifier si sa persistance des données fonctionne.

**Ome manager** est le plugin d'éclipse permettant de visualiser le contenu d'une base de données DB4O et de faire des requêtes (query)

### mode opératoire d'installation:

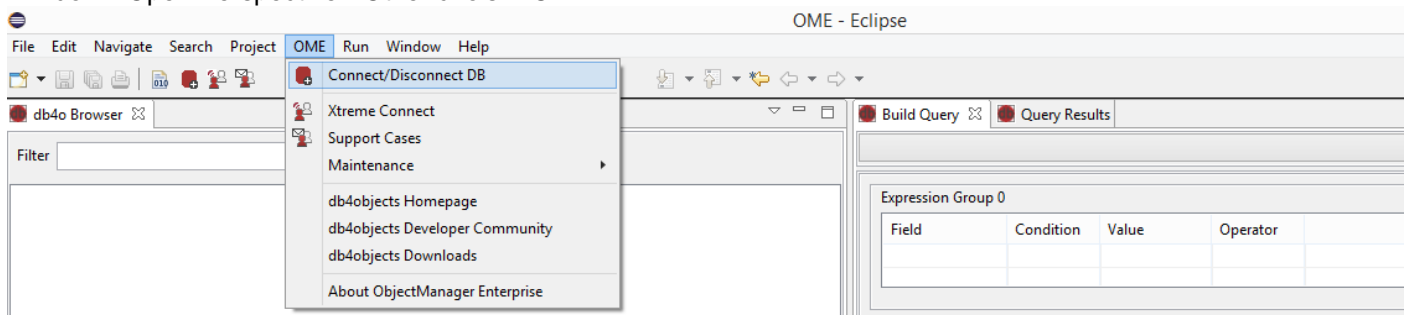
'Help' -> 'Install new Software' -> 'Available Software'

Choisir le bouton Add... -> 'Local...' et choisir ome\ObjectManagerEnterprise-Java-8.0.0 (à récupérer sous Partage)

Donner un nom comme "OME" puis valider et redémarrer Eclipse.

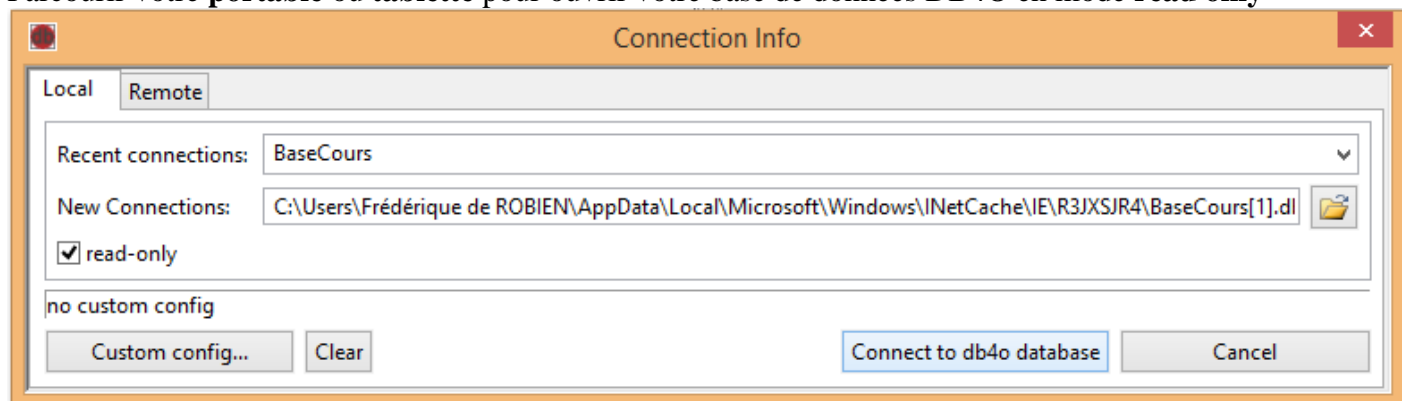
### mode opératoire d'utilisation:

Window->Open Perspective->Other choisir "OME"



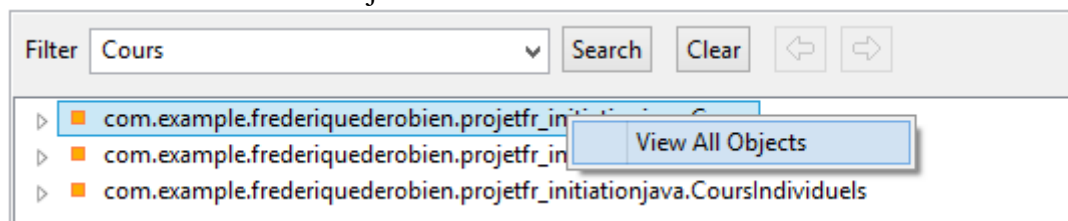
Choisir Connect

Parcourir votre **portable** ou **tablette** pour ouvrir votre base de données DB4O en mode **read only**



Dans Filter inscrire Cours

Avec clic droit "View all object"

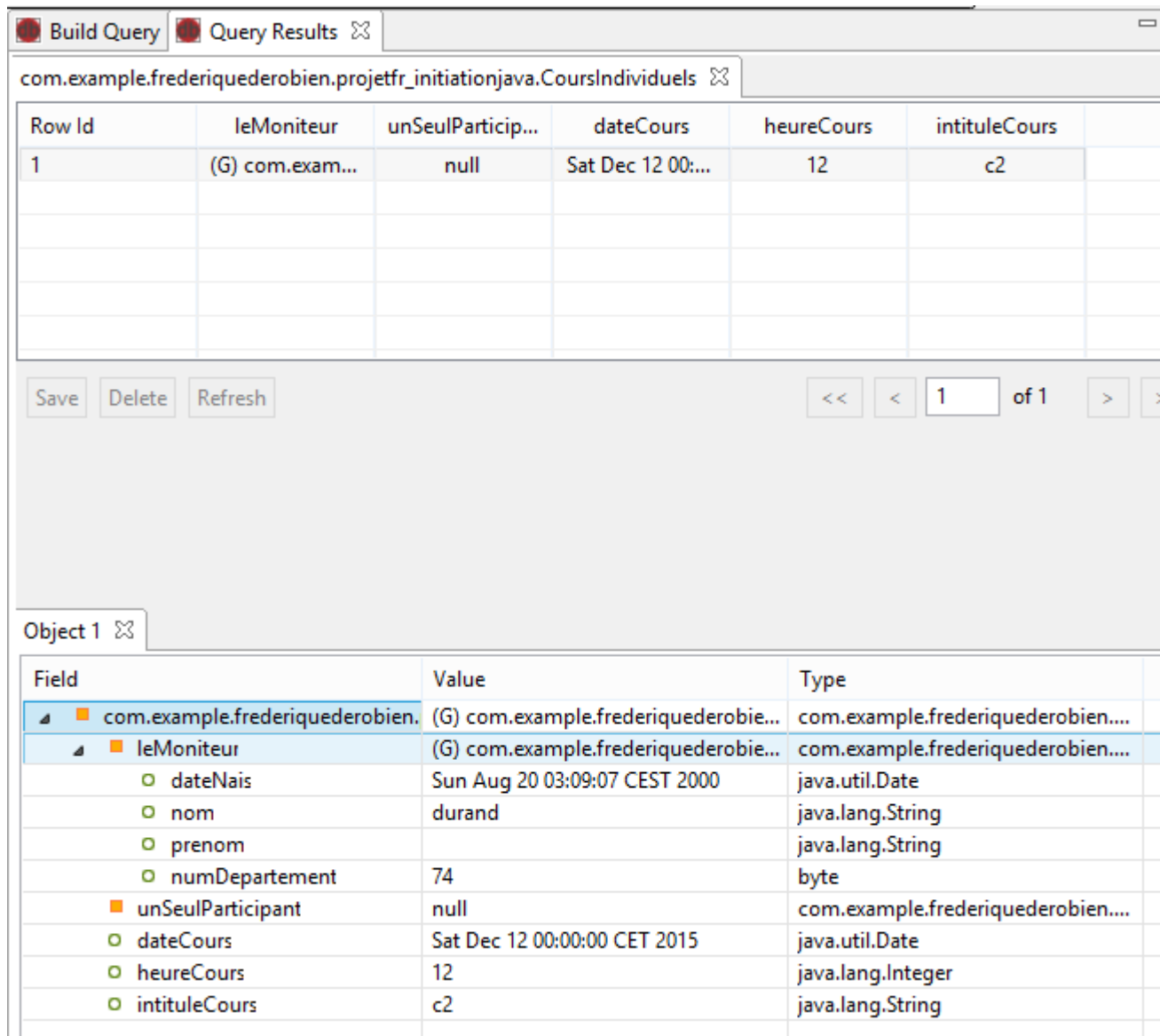


**Voici le résultat obtenu** pour un cours **Individuel** :

on voit les caractéristiques du coursIndividuel

et leMoniteur qui est un participant avec ses différentes caractéristiques.

A ce niveau, le participant n'est pas encore connu !



The screenshot shows the 'Query Results' window in Android Studio. The top bar indicates 'Build Query' and 'Query Results'. The query is 'com.example.frederiquederobien.projetfr\_initiationjava.CoursIndividuels'. The results table has columns: Row Id, leMoniteur, unSeulParticip..., dateCours, heureCours, and intituleCours. The first row shows: 1, (G) com.exam..., null, Sat Dec 12 00:..., 12, c2. Below the table are 'Save', 'Delete', and 'Refresh' buttons, and a pagination control showing '1 of 1'. The 'Object 1' tab is selected, showing a tree view of the object's fields and their values.

Field	Value	Type
com.example.frederiquederobien.	(G) com.example.frederiquederobie...	com.example.frederiquederobien....
leMoniteur	(G) com.example.frederiquederobie...	com.example.frederiquederobien....
dateNais	Sun Aug 20 03:09:07 CEST 2000	java.util.Date
nom	durand	java.lang.String
prenom		java.lang.String
numDepartement	74	byte
unSeulParticipant	null	com.example.frederiquederobien....
dateCours	Sat Dec 12 00:00:00 CET 2015	java.util.Date
heureCours	12	java.lang.Integer
intituleCours	c2	java.lang.String

Vous ne pouvez **pas** via Ome manager **modifier** les enregistrements de DB40.

Vous pouvez visualiser l'ensemble des classes gérées et pour chacune les instances d'objets.

## UTILISATION DE LA BASE DE DONNÉES OBJET

### DANS LE FICHIER MAINACTIVITY.CLASS

```
private Modele maBDO;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    /*creation de la nouvelle BDO*/
    maBDO = new Modele();
    //suite de l'algo déjà fait, ne pas changer
}
```

☛ Choisir l'endroit où positionner ces lignes suivantes ?

*/\*sauvegarde dans la bdd objet du cours créé\*/*

**maBDO**.saveCours(c);

Toast.makeText(getApplicationContext(), "SAUVEGARDE EN BDO", Toast.LENGTH\_SHORT).show();

Vérifier dans Eclipse que votre nouveau cours est bien présent dans la BDDO !

## LISTE DE TOUS LES COURS

- Créer un nouveau bouton listeCours
- Gérer le clic de ce bouton
- appeler la méthode ListeCours () de la classe Modele (liste des cours présents dans la BDDO)
- afficher TOUS les cours dans un toast !

## MODIFICATION D'UN OBJET DE LA BASE DE DONNÉES OBJET

### LISTE DE TOUS LES PARTICIPANTS DU COURS.

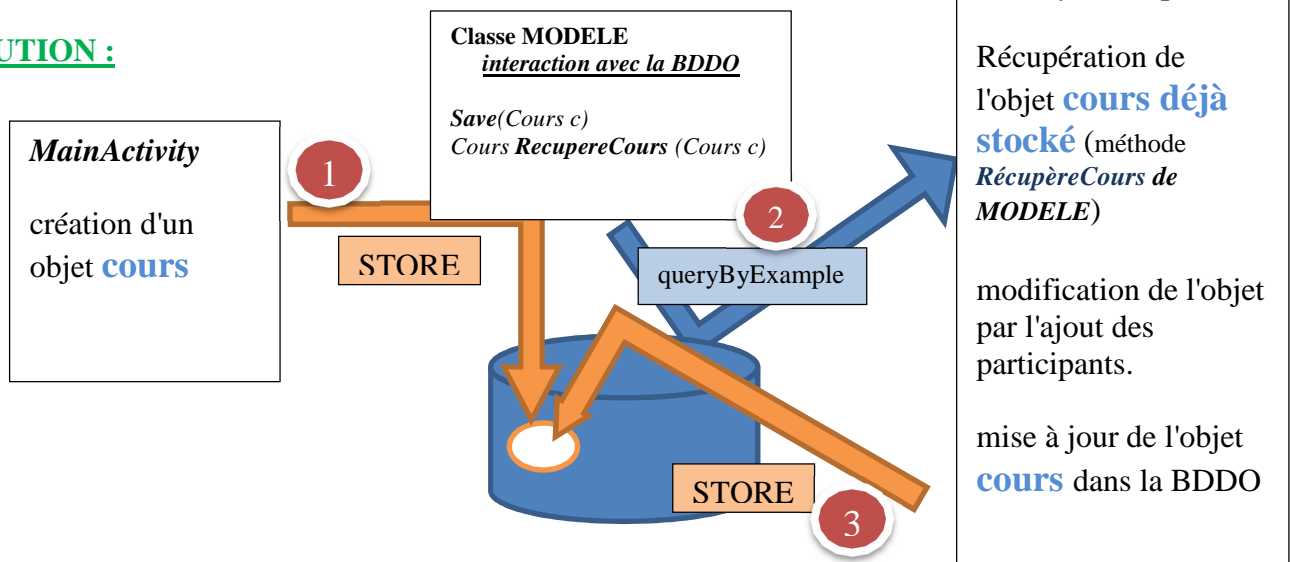
UNE fois votre cours sauvegardé dans la BDDO, faites les tests suivants sur votre AVD:

- Ajouter les participants dans ce cours
- afficher TOUS les participants du cours dans un Toast !
- Vérifier dans Eclipse que votre cours stocké en BDDO contient bien les participants ajoutés.

**ERREUR** : vous constatez que les participants ne sont pas ajoutés au cours pour plusieurs raisons :

- les 2 activités sont indépendantes
- le cours n'est pas automatiquement mis à jour dans la BDDO
- il faut appeler à nouveau la méthode store pour le mettre à jour !

### SOLUTION :



## AIDE :

1. Récupérer et ANALYSER la méthode suivante de la classe MODELE

```
public Cours recupereCours(Cours c) {
    /* je recupere le cours deja stocke dans la BDDO à partir de son intitule sachant
    que celui ci est Normalement unique (regle de gestion)*/
    open();
    try {
        Cours exempleC;
        if (c instanceof CoursCollectifs) {
            exempleC = new CoursCollectifs();
            exempleC.setIntituleCours(c.getIntituleCours());
            ObjectSet<CoursCollectifs> result = DataBase.queryByExample(exempleC);
            if (result.size() == 0) {
                return null;
            } else {
                return result.next();
            }
        } else {
            exempleC = new CoursIndividuels();
            exempleC.setIntituleCours(c.getIntituleCours());
            ObjectSet<CoursIndividuels> result = DataBase.queryByExample(exempleC);
            if (result.size() == 0) {
                return null;
            } else {
                return result.next();
            }
        }
    } finally {
        DataBase.close();
    }
}
```

2. Dans la classe ActivityParticipant

```
public class ActivityParticipant extends Activity {

    Cours coursRecupereMainActivity, coursRecupereBDO ;
    Modele maBDO;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_participant);
        maBDO = new Modele();
        /*recuperation du cours deserialise */
        coursRecupereMainActivity = (Cours) getIntent().getSerializableExtra("cours");

        coursRecupereBDO = maBDO.recupereCours(coursRecupereMainActivity);

        Toast.makeText(getApplicationContext(),coursRecupereBDO.toString(),Toast.LENGTH_SHORT).
        show();
    }
}
```

ENSUITE une fois que vous avez récupéré le cours de la BDDO, vous pouvez :

- lui ajouter les participants (à modifier)
- sauvegarder le coursRécupéré de la BDDO (à mettre au bon endroit dans votre code)

 +  +  = *Bon développement ....*

Observation + Réflexion + Réalisation