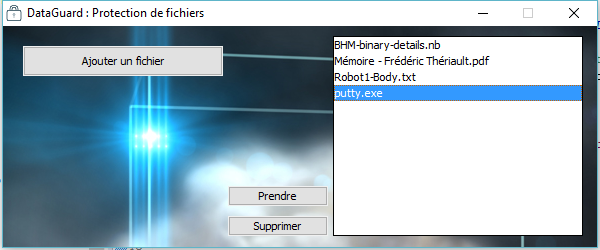
# **Introduction à BerkeleyDB**

L’objectif de ce laboratoire est de s’initier à une base de données de type Key-Value.

## L’application « DataGuard »



Le logiciel DataGuard permet à un usager de protéger ses documents (pdf, images, docx) en les stockant dans une base de données binaire.

Un usager, après s’être authentifié avec un mot de passe, a accès à tous ses fichiers stockés sur la base de données.

Les 3 opérations disponibles dans le logiciel sont :

1. Ajouter un fichier à la base de données
2. Prendre (restaurer) un fichier
3. Supprimer un fichier.

Le logiciel utilise BerkeleyDB pour stocker les informations du système (i.e. les fichiers)

## Notes

* Eclipse a été utilisé pour créer le programme, vous devriez donc l’utiliser pour compléter ce travail.
* Pour réaliser ce projet, je vous recommande FORTEMENT de suivre les étapes en ordre.
* Si vous avez des questions, n’hésitez pas à me demander de l’aide.
* En vérité, puisque les fichiers de la base de données du logiciel ne sont pas protégée, elle peut être facilement lue par un tierse programme. Pour rendre le programme réellement sécuritaire, il faudrait faire des opérations supplémentaires sur les fichiers de la base de données ou encore mettre la base de données sur un serveur et protéger les accès.
* Le mot de passe du logiciel est également en *clear text* dans le code, ce qui réduit également la sécurité de l’application.

## Exemple d’opérations pour BerkeleyDB

### Ajouter un élément

Database connection = DBConnection.getConnection();

String key = "cleElement";

byte[] data = ...;

try {

DatabaseEntry theKey = new DatabaseEntry(key.getBytes("UTF-8"));

DatabaseEntry theData = new DatabaseEntry(data);

connection.put(null, theKey, theData);

}

catch (Exception e) {

// Exception handling

}

### Supprimer un élément

Database connection = DBConnection.getConnection();

String key = "cleElement";

try {

DatabaseEntry theKey = new DatabaseEntry(key.getBytes("UTF-8"));

connection.delete(null, theKey);

}

catch (Exception e) {

// Exception handling

}

### Accéder à un élément

Database connection = DBConnection.getConnection();

String key = "cleElement";

try {

DatabaseEntry theKey = new DatabaseEntry(key.getBytes("UTF-8"));

DatabaseEntry theData = new DatabaseEntry();

if (connection.get(null, theKey, theData, LockMode.DEFAULT) == OperationStatus.SUCCESS) {

byte[] retData = theData.getData();

String foundData = new String(retData, "UTF-8");

System.out.println("Clé: '" + key + "' donnée: '" + foundData + "'.");

}

else {

System.out.println("Element inexistant");

}

}

catch (Exception e) {

}

### Accéder à la liste des éléments

Database connection = DBConnection.getConnection();

Cursor myCursor = null;

try {

myCursor = connection.openCursor(null, null);

DatabaseEntry foundKey = new DatabaseEntry();

DatabaseEntry foundData = new DatabaseEntry();

while (myCursor.getNext(foundKey, foundData, LockMode.DEFAULT) == OperationStatus.SUCCESS) {

String keyString = new String(foundKey.getData(), "UTF-8");

//String dataString = new String(foundData.getData(), "UTF-8");

//System.out.println("Clé|Valeur: " + keyString + " | " + dataString + "");

}

}

catch (DatabaseException de) {

System.err.println("Erreur de lecture de la base de données: " + de);

}

catch (UnsupportedEncodingException e) {

e.printStackTrace();

}

finally {

try {

if (myCursor != null) {

myCursor.close();

}

}

catch(DatabaseException dbe) {

System.err.println("Erreur de fermeture du curseur: " + dbe.toString());

}

}

## Les tâches à réaliser

Ces tâches doivent être réalisées en ordre, autrement il sera difficile de tester que tout fonctionne.

### Tâche #1 : Modifier le mot de passe pour se connecter au logiciel

Le fichier *UserDAO* possède le mot de passe utilisé lors de la connexion de l’usager (à l’ouverture du programme). Modifiez le mot de passe pour l’un de votre choix.

#### Notes/Énoncés :

private static final String password = "AAAaaa222";

### Tâche #2 : Ajouter un item/fichier dans la base de données

Il s’agit de terminer la méthode *ItemDAO.addItem(), qui* permet d’insérer les données du fichier dans la base de données.

Note : Le module de connexion à la base de données est déjà codé et disponible (voir classe DBConnection.java). Lorsque l’application démarre, elle créée la base de données BerkeleyDB dans le dossier « database » (si c’est la première fois).

#### Notes/Énoncés :

public static boolean addItem(String key, byte[] fileData) {  
 boolean success = false;  
 Database connection = DBConnection.*getConnection*();  
  
 String keyTemp = key;  
 byte[] data = fileData;  
  
 try {  
 DatabaseEntry theKey = new DatabaseEntry(key.getBytes("UTF-8"));  
 DatabaseEntry theData = new DatabaseEntry(data);  
 connection.put(null, theKey, theData);  
 success = true;  
 }  
 catch (Exception e) {  
 // Exception handling  
 }  
 return success;  
}

### Tâche #3 : Accéder à la liste des items/fichiers

En utilisant un curseur dans la méthode *ItemDAO.getItemList()*, il s’agit de parcourir les éléments de la base de données tout en construisant une liste d’*Items* qui sera retournée par la méthode.

#### Notes/Énoncés :

public static List<Item> getItemList() {  
 List<Item> items = new ArrayList<Item>();  
 Database connection = DBConnection.*getConnection*();  
 Cursor myCursor = null;  
  
 try {  
 myCursor = connection.openCursor(null, null);  
  
 DatabaseEntry foundKey = new DatabaseEntry();  
 DatabaseEntry foundData = new DatabaseEntry();  
  
 while (myCursor.getNext(foundKey, foundData, LockMode.*DEFAULT*) == OperationStatus.*SUCCESS*) {  
 String keyString = new String(foundKey.getData(), "UTF-8");  
 items.add(new Item(keyString));  
  
 //String dataString = new String(foundData.getData(), "UTF-8");  
 //System.out.println("Clé|Valeur: " + keyString + " | " + dataString + "");  
 }  
 }  
 catch (DatabaseException de) {  
 System.*err*.println("Erreur de lecture de la base de données: " + de);  
 }  
 catch (UnsupportedEncodingException e) {  
 e.printStackTrace();  
 }  
 finally {  
 try {  
 if (myCursor != null) {  
 myCursor.close();  
 }  
 }  
 catch(DatabaseException dbe) {  
 System.*err*.println("Erreur de fermeture du curseur: " + dbe.toString());  
 }  
 }  
 return items;  
}

### Tâche #4 : Supprimer un item/fichier

La méthode *ItemDAO.deleteItem()* doit supprimer l’élément spécifié de la base de données.

#### Notes/Énoncés :

public static boolean deleteItem(String key) {  
 boolean success = false;  
 Database connection = DBConnection.*getConnection*();  
 String keyTemp = key;  
  
 try {  
 DatabaseEntry theKey = new DatabaseEntry(keyTemp.getBytes("UTF-8"));  
 connection.delete(null, theKey);  
 success = true;  
 }  
 catch (Exception e) {  
 // Exception handling  
 }  
   
 return success;  
}

### Tâche #5 : Restaurer un fichier

Lorsque l’usager désire reprendre un fichier de la base de données, la méthode *ItemDAO.restoreItem()* est appelée. Celle-ci doit donc aller récupérer l’élément grâce à la clé, puis créer le fichier à l’endroit voulu.

Note : Pour créer un fichier à partir d’un byte[], voici comment faire :

FileOutputStream fos = new FileOutputStream(destinationFile);

fos.write(data);

fos.close();

#### Notes/Énoncés :

public static boolean restoreItem(String key, File destinationFile) {  
 boolean success = false;  
 Database connection = DBConnection.*getConnection*();  
  
 String keyTemp = key;  
  
 try {  
 DatabaseEntry theKey = new DatabaseEntry(keyTemp.getBytes("UTF-8"));  
 DatabaseEntry theData = new DatabaseEntry();  
  
 if (connection.get(null, theKey, theData, LockMode.*DEFAULT*) == OperationStatus.*SUCCESS*) {  
 byte[] retData = theData.getData();  
 String foundData = new String(retData, "UTF-8");  
 FileOutputStream fos = new FileOutputStream(destinationFile);  
 fos.write(retData);  
 fos.close();  
 System.*out*.println("Clé: '" + keyTemp + "' donnée: '" + foundData + "'.");  
 success = true;  
 }  
 else {  
 System.*out*.println("Element inexistant");  
 }  
 }  
 catch (Exception e) {  
 }  
 return success;  
 }  
}