Documentation
Projet gus05

Titre : Chantier JDBC

Auteur : Augustin Delale

Version : 14/12/2009

Chantier JDBC

Ce document doit servir de base de départ au chantier gus05 dédié à la technologie Java JDBC qui permet d'utiliser des bases de données dans les applications Java.

Chargement en mémoire des drivers JDBC

Pour pouvoir accéder à un type particulier de SGBD (Système de Gestion de Base de Données), il faut tout d'abord charger en mémoire la classe de driver JDBC nécessaire à ce type. Le lien suivant proposé par Sun vous permettra de trouver tous les drivers JDBC disponibles :

http://developers.sun.com/product/jdbc/drivers

Le chargement d'un driver doit intervenir au tout début de l'application et se fait toujours de la même manière, en utilisant l'entité de multiplicité UNIQUE suivante :

```
package gus05.entity.gus.database.loaddriver;

import java.io.PrintStream;
import java.sql.Driver;
import java.sql.DriverManager;
import gus05.framework.core.Entity;
import gus05.framework.core.Outside;
import gus05.framework.features.Give;

public class LoadDriver implements Entity, Give {

    public String getName()             {return "gus.database.loaddriver";}
    public String getCreationDate()     {return "2009.02.18";}

    private PrintStream out;

    public LoadDriver() throws Exception
    {out = (PrintStream) Outside.resource(this,"out");}

    public void give(Object obj) throws Exception
    {
       String path = (String) obj;
       Class driverClass = Class.forName(path);
       Driver driver = (Driver)driverClass.newInstance();
       DriverManager.registerDriver(driver);
       out.println("JDBC driver loaded: "+path);
    }
}
```

Les entités présentées ci-dessous utilisent toutes cette entité générique de chargement, vous permettant ainsi de charger en mémoire le ou les drivers que vous souhaitez utiliser. Vous devrez pour ce faire inclure les entités dans la "start list" de votre application gus05.

Le driver "JDBC-ODBC"

Le driver "JDBC-ODBC" est une passerelle permettant à l'API JDBC d'utiliser ODBC (Open Database Connectivity) – lien wiki – notamment nécessaire pour se connecter aux bases de données Access.

Ce driver est déjà inclu dans dans la JRE.
Classe du driver : *sun.jdbc.odbc.JdbcOdbcDriver*

```
package gus05.entity.gus.database.loaddriver.odbc;

import gus05.framework.core.Entity;
import gus05.framework.core.Outside;

public class OdbcDriver implements Entity {

    public String getName()             {return "gus.database.loaddriver.odbc";}
    public String getCreationDate()     {return "2006.08.31";}

    public static final String PATH = "sun.jdbc.odbc.JdbcOdbcDriver";

    public OdbcDriver() throws Exception
    {Outside.service(this,"gus.database.loaddriver").give(PATH);}
}
```

Le driver pour MySQL

MySQL est un SGBD Open Source parmi les plus populaires au monde puisqu'il est gratuit et comparable aux autre grand SGBD du marché (Oracle, Sysbase…) en terme de performance et de capacités.

Vous pouvez télécharger la dernière version de MySQL ici :
http://dev.mysql.com/downloads/

Le driver pour Java est disponible ici :
http://www.mysql.com/products/connector/

Bien sûr, avant de pouvoir l'utiliser avec votre application Java, vous devrez d'abord installer MySQL sur votre ordinateur (ou sur un serveur), puis ajouter le jar du driver au répertoire lib\ext de votre JVM.

Classe du driver : *com.mysql.jdbc.Driver*

```
package gus05.entity.gus.database.loaddriver.mysql;

import gus05.framework.core.Entity;
import gus05.framework.core.Outside;

public class MySqlDriver implements Entity {

    public String getName()             {return "gus.database.loaddriver.mysql";}
    public String getCreationDate()     {return "2009.02.18";}

    public static final String PATH = "com.mysql.jdbc.Driver";

    public MySqlDriver() throws Exception
    {Outside.service(this,"gus.database.loaddriver").give(PATH);}
}
```

Le driver pour SQLite

SQLite est une librairie applicative dans le domaine public qui permet de gérer une base de données sous forme d'un fichier, sans aucune installation de SGBD. Il s'agit donc d'une des solutions les plus simples et les plus accessibles pour utiliser JDBC. En fait, vous n'aurez besoin que du driver SQLiteJDBC dont le fichier jar est téléchargeable ici (que vous devrez ajouter au répertoire lib\ext de votre JVM) :

http://www.zentus.com/sqlitejdbc/

Classe du driver : *org.sqlite.JDBC*

```
package gus05.entity.gus.database.loaddriver.sqlite;

import gus05.framework.core.Entity;
import gus05.framework.core.Outside;

public class SQLiteDriver implements Entity {

    public String getName()             {return "gus.database.loaddriver.sqlite";}
    public String getCreationDate()     {return "2009.11.15";}

    public static final String PATH = "org.sqlite.JDBC";

    public SQLiteDriver() throws Exception
    {Outside.service(this,"gus.database.loaddriver").give(PATH);}
}
```

Le driver pour Oracle

Personnellement, je n'ai jamais utilisé Oracle. Voici cependant l'entité qui vous permettra de l'utiliser. Tout d'abord, vous devez récupérer et installer le driver JDBC pour Oracle, disponible ci-dessous :

http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html

Classe du driver : *oracle.jdbc.driver.OracleDriver*

```
package gus05.entity.gus.database.loaddriver.oracle;

import gus05.framework.core.Entity;
import gus05.framework.core.Outside;

public class OracleDriver implements Entity {

    public String getName()             {return "gus.database.loaddriver.oracle";}
    public String getCreationDate()     {return "2006.08.31";}

    public static final String PATH = "oracle.jdbc.driver.OracleDriver";

    public OracleDriver() throws Exception
    {Outside.service(this,"gus.database.loaddriver").give(PATH);}
}
```

Se connecter à une base de données

Après avoir chargé en mémoire les drivers nécessaires, l'étape suivante consiste à se connecter à des bases de données afin de stocker manipuler ou récupérer des données. Ceci s'éffectue de la manière suivante