UniBank User Manual

DB Hercules

Autor: Adham Haddad
Datum: 19.05.2014

Ottostr. 15
96047 Bamberg
Tel.: +49 951 98046-200
Fax: +49 951 98046-150
Email: info@db-hercules.de
Web: www.db-hercules.de

Index of Contents

1 Preface

Welcome in the user community of bank sort code module UniBank.

UniBank provides the following functionalities:

- validating german bank sort codes and account number
- validating IBANs
- querying information about credit institutes by their bank sort code such as name, city, postal code, institute number etc.

By using UniBank you can validate bank connections before accepting and using them and thus avoid unnecessary costs resulting for example from return credit notes and detect fraud attempts or failure in early stages.

UniBank is implemented in the platform-independent programming language JAVA allowing easy integration.

UniBank is very useful whereever bank connection are used and thus it has a wide application area including:

- Online-Forms
- Call Centers
- Onlilne Shops
- Systems for accounts
- systems for comissions

More information about installation, connetion of UniBank in your own software, you will find in this maual.

Sincerely

DB Hercules -Team

2 Features

UniBank offers the following features:

- Import the data file of the german bank codes in a database
  This feature offers the possibility to integrate the File of German bank codes of the „Deutsche Bundesbank“ in a database. This is supposed for the following features
- Validation of present bank code numbers, account numbers andIBAN

3 Conventions in this Documentation

In this manual we use the following typefaces and symbols, to mark spcial textes and contexts:

3.1 Typofaces

`WORDS WRITTEN IN CAPITALS WITH EXPENDED TYPES` are used to sign spezial File Types.
Example: „`JAR`“
Bold face types mark the name of Java Classes and tables in the text.
Example: „**FgBoPlz**“, „**TbFgPlz**“
Cursive, bold face types mark the name of a method in the text.
Example: „***getPlzInfo()***“
Cursive faca types (1) mark names of parameters in a method in the text.
Example: „***getBankInfo(****String blz****)***“
Cursive face types (2) mark simple Java file types and their value.
Example: „*boolean*“,“*true*„
Cursive face types (3) mark names of index, path, filename or URL in the text.
Example: „*C:\Temp*“

3.2 Terms / Shortcuts

The following diagram define the terms in this document:
BLZ: Bankleitzahl – bank code number

4 Technical Requirements

JAVA runtime environment version 1.5, or higher
index of the bank sort codes of the German Federal Bank (Deutsche Bundesbank) to download here for free:
http://www.bundesbank.de/Redaktion/DE/Standardartikel/Kerngeschaeftsfelder/Unbarer_Zahlungsverkehr/bankleitzahlen_download.html

5 Supported Databases

- Oracle Version 9i or higher (tested)
- DB2 Version 5.2 or higher
- SAPDB / MAXDB Version 7.1

6 Included in Delivery

You get the delivery in one data file „dbh-*unibank-binary-XXX.zip*“ as the case maybe „dbh-unibank-source-XXX.***zip****“* (XXX ist he actual version)

7 Download und Installation

1. Download the Zip-File
2. Unpack UniBank.zip in an optinal index. You get the following structure:
   - apidoc: here you find the Javadoc of UniBank
   - manual: here you find the manual
   - lib: includes the JAR-Files of UniBank and the necessary libraries, including:
     - log4j-XXX.jar (XXX is the actual version)
     - ojdbc-XXX.jar (XXX is the actual version)
     - dbh-unibank-source-XXX.jar bzw. dbh-unibank-source-XXX.jar (XXX is the actual version)
   - sql: offers the SQL-scripts to apply and delete the table of the data files for the supported database system
   - src: includes the source (only if you buy the source-version)
3. add all JAR-Files in the lib-table in the Verzeichnis classpath

8 Create Tables and Views

Here we show, how to create the necessary tables and views.
These tables and views are required for the features of UniBank. To create the tables and views you execute the SQL-script „create.sql“in the direcory „*sql*“.

8.1 Delete Tables and Views

If you don`t need the tables any more, you can delete them with executing the SQL-script „drop.sql“. You find this script in the directory „*sql*“. The applied tables and views will be deleted completed.

Die unter Punkt 8 angelegten Tabellen und Views werden dabei aus der Datenbank entfernt.

8.2 Truncate tables

To truncate the table, you execute the SQL Scripts „delete.sql“. This script you find in the directory „*sql*“. The content of the tables, you create (explained under 9) will be deleted completed.

9 Load of the data „bank codes“ into a Database

Here we show, how to load the data „Bank codes“ into the database. To load the data use the class *de.dbh.blz.business.BSCImport*. This class offers the following method:

```
/**
* Diese Methode liest die Postleitzahlendatei und spielt sie in die Datenbank ein.
* Folgende Tabellen müssen vorhanden sein:
* TBBLZBANK
* TBBLZBANKDATEI
*
* Der Ablauf des Imports und eventuelle Fehler werden in eine Logdatei
* protokolliert.
*
* Diese Methohde dient zum Importieren der Bankleitzahlendatei der Deutschen
* Bundesbank in eine Datenbank
*
* @param bankSortCodeFile File
*            ein File-Objekt zur Bankleitzahlendatei der Deutschen Bundesbank
* @param consoleOutput boolean,
*            ein Flag zur Steuerung der Ausgabe auf der Console. <em>true</em>
*            wenn eine Ausgabe auf der Console erwünscht ist.
* @param licenseFile File,
*            die Lizenzdatei für die Benutzung des Moduls
* @throws SQLException,
*             wenn ein Fehler beim Einspielen der Bankleitzahldatei in die
*             Datenbank auftritt
* @throws IOException,
*             wenn ein Fehler beim Lesen der Datei aufritt
* @throws DbhBSCLicenseInvalidException wenn die Lizenzdatei ungültig ist
*/
public void importFile(File bankSortCodeFile, File licenseFile, boolean
consoleOutput) throws SQLException, IOException, DbhBSCLicenseInvalidException
```

You find a detailed description in the Javadoc. Following an example for how to use:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import de.dbh.blz.business.BSCImport;
import java.io.File;
import java.io.IOException;
import de.dbh.blz.exception.DbhBSCLicenseInvalidException;

/**
```

```
 * This class demostrates using UniBank for importing the bank sort code of the
 * german Bundesbank
 */
public class DoImport {

   public static void main(String[] args) {
      /*
      * JDBC Parameter
      */
      String url = "jdbc:oracle:thin:@server:port:db_instanz";
      String user = "username";
      String password = "userpassword";
      String driverClassName = "oracle.jdbc.driver.OracleDriver";

      //the database scheme into which the bank sort code was imported. This
      //parameter must be set to the empty string "" if no database scheme is
      //used.
      String scheme = "scheme";

      /*
       *  create the database connection object
       */
      try {
            Class.forName(driverClassName);
      }
      catch (ClassNotFoundException ex) {
            ex.printStackTrace();
      }
      Connection con = null;
      try {
            con = DriverManager.getConnection(url, user, password);
      }
      catch (SQLException ex) {
            ex.printStackTrace();
      }
      // the bank code sort file
      String filename_plz = "C:" + File.separator +
                            "temp" + File.separator +
                            " BLZ_20070305.txt";
      File blzFile = new File(filename_plz);

      // the license file
      String filename_lic = "C:" + File.separator +
                           "temp" + File.separator +
                           "Key.bin";
      File licFile = new File(filename_lic);

      // create an BSCImport-Object
      BSCImport bscImport = new BSCImport(con,scheme);

      try {
         // start the import
         bscImport.importFile(infile, licfile, true);
      }
      catch(DbhBSCLicenseInvalidException ex) {
       System.out.println("the license file is not valid");
```

```
        }
        catch(IOException ex) {
         System.out.println("error while reading the bank sort file");
        }
        catch(SQLException ex) {
           System.out.println("error while querying the database");
        }
        finally{
           con.close();
        }

    }
}
```

Please take into account, that thereby already existing entries, from a previous import procedure, are possibliy removed.
The result of imports and/or error messages you can let log in two kinds:

1. assisted by Log4j: UniBank uses Log4j. You can configurate your Log4j according to your requirements.
2. Output on the Console: therefor you send "`true`„ to the `importFile(...)` as last parameter „`consoleOuput`" as described in the example above.(For details please consult the Javadoc)

10 Functions for querying

Here you see, how to query the imported data of bank codes
To query the data of bank codes, you use the class *de.dbh.blz.business.BSCInfo*.
UniBank offers 3 features to query the data of bank codes:

10.1 Method to check if the combination of bank code and account number is correct

```
/**
* diese Methode prüft ob eine BLZ-Kontonummer-Kombination gültig ist.
*
* @param bankSortCode
*            die zu prüfende Bankleitzahl
* @param accountNumber
*            die zu prüfende Kontonummer
* @return true wenn die BLZ-Kontonummer-Kombination gültig ist, false sonst
* @throws DbhBSCcheckMethodNotImplementedException
*             wenn die Bank eine Prüfmethode verwendet die in Ihrer Version
*             nicht implemntiert ist weil sie zum Release-Zeitpunkt noch nicht
*             bekannt war. Upgraden Sie bitte Ihren UniBank-Modul
* @throws SQLException
*             wenn ein Datenbankfehler bei der Bearbeitung der Abfrage
```

```
*             auftritt
*/
public boolean checkAccount(String bankSortCode, String accountNumber)
throws DbhBSCcheckMethodNotImplementedException, SQLException
```

10.2 Method to check wether the bank code exists

```
/**
* diese Methode prüft ob eine Bankleitzahl existiert oder nicht
*
* @param bankSortCode
*            die zu prüfende Bankleitzahl
* @return true wenn die Bankleitzahl existiert, false sonst
* @throws SQLException
*             wenn ein Datenbankfehler bei der Bearbeitung der Abfrage
*             auftritt
*/
public boolean checkBankSortCode(String bankSortCode) throws SQLException
```

10.3 Method to check if the IBAN is valid

```
/**
  * This method validates an IBAN. If no exception occurs the provided
  * <code>iban</code> is correct.
  * The check includes the following rules.
  * <ul>
  *     <li>Format of IBAN must meet ISO 13616:2007</li>
  *     <li>IBAN is check with ISO 7064 (Modulus 97-10)</li>
  * </ul>
  * For IBAN with country identifier DE the following rules are added.
  * <ul>
  *     <li>IBAN length is 22</li>
  *     <li>
  *         Bank number is extracted and checked against the
  *         valid bank number list of the german federal bank.
  *     </li>
  *     <li>
  *         Account and bank number extracted and checked against
  *         the rules of the german federal bank.
  *     </li>
  * </ul>
  *
  * @param iban the IBAN to be validates
  * @throws IBANNotValidException if the passed IBAN is not valid
  */
 public void checkIban(String iban) throws IBANNotValidException
```

10.4 Methode to check if the BIC is valid

```
/**
  * This method validates a BIC. If no exception occurs the provided
```

```
 * <code>bicCode</code> is correct.
 * The check includes the following rules.
 * <ul>
 *     <li>Format of BIC must meet ISO 9362</li>
 * </ul>
 * For BIC with country identifier DE the following rules are added.
 * <ul>
 *     <li>
 *         Bank number is checked against the valid bank number list of
 *         the german federal bank.
 *     </li>
 * </ul>
 * <br/>
 * This method is not useable in trial mode.
 *
 * @param bicCode the bic code to be validated
 * @throws BICNotValidException If the provided BIC is not valid.
 */
public void checkBicCode(String bicCode) throws BICNotValidException
```

10.5 Method to get information about the bank

```
/**
* Diese Methode dient zur Abfrage der Bankinformationen, die aus der
* Bankleitzahlendatei importiert wurden. Diese Informationen beinhalten:
* Bankleitzahl
* Bankname
* BIC
* Ort
* Postleitzahl
* ob die Bank BLZ-führend ist
* Institutsnummer
* Kurzbezeichnung
* Nummer
* verwendete Prüfmethode
*
* Die Methode gibt ein Array von Objekten vom Typ BlzResultBank. Jedes solche
* Objekt umfasst die obengenannten Informatione zu einer Bank.
* Falls keine Bank gefunden wurde ist die Rückgabe ein leeres Array.
*
* @param bankSortCode
*            die Bankleitzahl (es kann auch mit einer Teilbankleitzahl gesucht
*            werden wenn false für den Parameter exakt eingegeben wird)
* @param isMainBranch
*            falls true werden nur Banken gesucht die BLZ-führend sind. Bei
*            false werden alle passenden Filialen gesucht
* @param exact
*            falls true werden Banken gesucht deren BLZ blz exakt entspricht.
*            bei false wird die blz als Anfangsstück der BLZ verstanden.
* @return BSCResultBank[] die gefunden Banken. Falls keine Bank gefunden wurde
*            wird ein leeres Array zurückgegeben
* @throws SQLException
*             wenn ein Fehler beim Abfragen der Datenbank auftritt.
```

```
 */
 public BSCResultBank[] getBankInfo(String bankSortCode, boolean isMainBranch,
boolean exact) throws SQLException
```

11 Codeexamples

The following code examples show how to use UniBank. Not all queries which are possible are shown. Look at the Javadoc to get a general idea of all the possible applications.

11.1 The Bank Code „77050000“ exists?

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import de.dbh.blz.business.BSCInfo;

/**
 * This class demonstrates using UniBank for validating a bank sort code
 */
public class CheckBankSortCode {

   public static void main(String[] args) {

      // JDBC Parameter
      String url = "jdbc:oracle:thin:@server:port:db_instanz";
      String user = "username";
      String password = "userpassword";
      String driverClass = "oracle.jdbc.driver.OracleDriver";

      /*
       the database scheme into which the bank sort code was imported. This
       parameter must be set to the empty string "" if no database scheme is
       used.
       */
      String scheme = "scheme";
      Connection con = null;

      // create the database connection
      try {
         Class.forName(driverClass);
         con = DriverManager.getConnection(url, user, password);
      }
      catch (Exception ex) {
         ex.printStackTrace();
      }

      // create an BSCInfo-object
      BSCInfo bscInfo = new BSCInfo(con,scheme);

      try {
         // validate the bank sort code
         if (bscInfo.checkBankSortCode("77050000")) {
```

```
            System.out.println("the bank sort code is valid");
         }
         else {
            System.out.println("the bank sort code is not valid");
         }
      }
      catch (SQLException e) {
         e.printStackTrace();
      }
      finally{
          try {
              con.close();
          }
          catch (SQLException e) {
            e.printStackTrace();
          }
      }
   }
}
```

11.2 The IBAN „CH100023 00A1 0235 0260 1“ is real?

```
import de.dbh.blz.business.BSCInfo;
import de.dbh.blz.exception.DbhBSCIbanFormatException;

/**
 * This class demonstrates using UniBank for validating an IBAN
 */
public class CheckIban {

   public static void main(String[] args) {

      try {
         // validate the IBAN
         if (BSCInfo.checkIban("CH100023 00A1 0235 0260 1")) {
            System.out.println("The IBAN is valid");
         }
         else {
            System.out.println("The IBAN is not valid");
         }
      }
      catch (DbhBSCIbanFormatException ex) {
            System.out.println("The IBAN does not follow the IBAN-format and
                                therefore cannot be valid");
      }
   }
}
```

11.3 Is the bank connection correct: Bank Code (BLZ) =77050000, account number =4466347 ?

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import de.dbh.blz.business.BSCInfo;
import de.dbh.blz.exception.DbhBSCcheckMethodNotImplementedException;


/**
 * This class demonstrates using UniBank for validating an account number
 */
public class CheckAccountNumber {

   public static void main(String[] args) {

      // JDBC Parameter
      String url = "jdbc:oracle:thin:@server:port:db_instanz";
      String user = "username";
      String password = "userpassword";
      String driverClass = "oracle.jdbc.driver.OracleDriver";

      /*
       the database scheme into which the bank sort code was imported. This
       parameter must be set to the empty string "" if no database scheme is
       used.
      */
      String scheme = "scheme";

      Connection con = null;
      // create the database connection object
      try {
         Class.forName(driverClass);
         con = DriverManager.getConnection(url, user, password);
      }
      catch (Exception ex) {
         ex.printStackTrace();
      }

      // create an BSCInfo-Object
      BSCInfo bscInfo = new BSCInfo(con,scheme);

      try {
         // validate the account number
         if (bscInfo.checkAccount("77050000","4466347")) {
            System.out.println("the account number is valid for the given bank
                                sort code");
         }
         else {
            System.out.println("the account number is not valid for the given bank
                                sort code");
         }
```

```
        }
        catch (SQLException e) {
            e.printStackTrace();
        }
        catch (DbhBSCcheckMethodNotImplementedException e) {
            StringBuffer sb = new StringBuffer();
            sb.append("The credit institute uses a check-digit calculation ");
            sb.append("method that is not implemented in your version of ");
            sb.append("UniBank. Please upgrade your UniBank.");
            System.out.println(sb.toString());
        }
        finally{
            try {
                con.close();
            }
            catch (SQLException e) {
              e.printStackTrace();
            }
        }
    }
}
```

11.4 What is the name of the bank with the bank code „77050000“ and where is it?

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import de.dbh.blz.business.BSCInfo;
import de.dbh.blz.business.BSCResultBank;

/**
 * This class demonstrates using UniBank for querying information about credit
 * institutes by their bank sort code
 */
public class GetBankInfo {

   public static void main(String[] args) {
      // JDBC Parameter
      String url = "jdbc:oracle:thin:@server:port:db_instanz";
      String user = "username";
      String password = "userpassword";
      String driverClass = "oracle.jdbc.driver.OracleDriver";

      /*
       the database scheme into which the bank sort code was imported. This
       parameter must be set to the empty string "" if no database scheme is
       used.
      */
      String scheme = "scheme";
      Connection con = null;
      // create the database connection object
      try {
         Class.forName(driverClass);
         con = DriverManager.getConnection(url, user, password);
      }
      catch (Exception ex) {
         ex.printStackTrace();
      }

      // create an BSCInfo-Object
      BSCInfo bscInfo = new BSCInfo(con,schema);


      try {
         // compute the results and get the array of results
         BSCResultBank[] results = bscInfo.getBankInfo("77050000",false,true);
         if (results.length == 0) {
            System.out.println("no matching credit institues found");
         }
         else{
            // iterate over the array
            for (int i = 0; i<results.length; i++) {
               System.out.println("name=" + results[i].getName());
               System.out.println("city=" + results[i].getCity());
```

```
                System.out.println("postal code=" + results[i].getPostalCode());
                System.out.println("");
            }
        }
      }
      catch (SQLException e) {
         e.printStackTrace();
      }
      finally{
          try {
              con.close();
          }
          catch (SQLException e) {
            e.printStackTrace();
          }
      }
   }
}
```