Table of Contents

Hinter einer gut geplanten Datenbank Anwendung steckt immer ein gut designtes Datenmodell. Dieses erfüllt im Idealfall die 3. Normalform, oder zumindest ein vernünftiges Mittel aus 2. und 3. Normalform. Das User Interface visualisiert im einfachsten Fall das Datenmodell und ermöglicht die Erfassung von Stamm- und Bewegungsdaten.

Die Datenbank bietet jedoch sehr viele Vorteile die man im User Interface nutzen möchte, ohne dabei viel Zeit zu verlieren. Einer dieser Vorteile ist der “Default Wert” von Spalten in Tabellen.

Häufig existieren sogenannte Flag Spalten, die bereits in der Datenbank einen Default Wert erhalten. Ein klassisches Beispiel hierfür sind J/N (= Ja/Nein) Spalten. Der Default Wert ist z.B.: N (= Nein).

Der Entwickler müsste im User Interface beim Erstellen eines Datensatzes diese Default Werte berücksichtigen und für den Anwender vorbelegen. Das ist einerseits eine immer wiederkehrende Tätigkeit und andererseits sehr Fehleranfällig. Außerdem möchte der Entwickler lieber knifflige Probleme lösen anstatt seine Zeit mit langweiligen Tätigkeiten zu verbringen.

Und genau hier beginnt die Arbeit von JVx. Das Framework erkennt die Default Werte von Spalten vollautomatisch und übernimmt die definierten Werte beim Erstellen von neuen Datensätzen, direkt ins User Interface. Es ist jedoch zu beachten, daß nur Konstante Default Werte berücksichtigt werden, da Funktionsaufrufe immer Datenbankabhängig sind und somit Logik enthalten können!

Anwendungsbeispiel

Unsere Applikation enthält eine Benutzerverwaltung für die Erstellung und Bearbeitung von Applikations-Benutzern. Die zugrunde liegende Datenbank Tabelle wurde wie folgt erstellt (Oracle Syntax):

create.sql

```
CREATE TABLE USERS
(
  ID               NUMBER(16) NOT NULL,
  USERNAME         VARCHAR2(200) NOT NULL,
  PASSWORD         VARCHAR2(200),
  CHANGE_PASSWORD CHAR(1) DEFAULT 'N' NOT NULL,
  ACTIVE           CHAR(1) DEFAULT 'Y' NOT NULL,
  VALID_FROM       DATE,
  VALID_TO         DATE,
  CREATED_BY       VARCHAR2(200) NOT NULL,
  CREATED_ON       DATE DEFAULT sysdate NOT NULL,
  CHANGED_BY       VARCHAR2(200),
  CHANGED_ON       DATE,
  TITLE            VARCHAR2(64),
  FIRST_NAME       VARCHAR2(200),
  LAST_NAME        VARCHAR2(200),
  EMAIL            VARCHAR2(200),
  PHONE            VARCHAR2(200),
  MOBILE           VARCHAR2(200)
);

-- Create/Recreate primary, unique and foreign key constraints
```

```
ALTER TABLE USERS
  ADD CONSTRAINT USER_PK PRIMARY KEY (ID);

ALTER TABLE USERS
  ADD CONSTRAINT USER_UK UNIQUE (USERNAME);
```

Wir benötigen nun ein Server Objekt für den Zugriff auf die Datenbank bzw. Tabelle:

```
public DBStorage getUsers() throws Exception
{
    DBStorage users = (DBStorage)get("users");
    if (users == null)
    {
        users = new DBStorage();
        users.setDBAccess(getDBAccess());
        users.setWritebackTable("USERS");
        users.setDefaultSort(new SortDefinition("USERNAME"));
        users.open();

        put("users", users);
    }
    return users;
}
```

Am Client erstellen wir nun ein Objekt für den Server Zugriff:

```
rdbUsers.setDataSource(dataSource);
rdbUsers.setName("users");
rdbUsers.open();
rdbUsers.getRowDefinition().setTableColumnNames(
                            new String[] {"USERNAME", "ACTIVE",
"CHANGE_PASSWORD"});
```

Wenn wir nun einen neuen Benutzer erstellen, so werden die Felder “ACTIVE” und “CHANGE_PASSWORD” automatisch mit den Default Werten, aus der Datenbank, befüllt. Das Feld “ACTIVE” erhält den Wert “Y” und “CHANGE_PASSWORD” erhält den Wert “N”.

Um die Default Werte zu ignorieren, können folgende Methoden verwendet werden:

```
//per instance
users.setDefaultValue(false)

//for all instances (static)
DBStorage.setDefaultDefaultValue(false)
```

Wenn in der Datenbank keine Default Werte definiert wurden, können diese auch über das API gesetzt werden:

```
users.open();
```

```
//sets the default value to "X"
users.getMetaData().getColumnMetaData("ACTIVE").setDefaultValue("X");
```

Falls auf die Verwendung der Default Werte verzichtet wird, so erreicht man die Default Belegung über die Verwendung von Client seitigen events wie z.B.

```
rdbUsers.eventAfterInserting().addListener(this, "afterInserting");
...
public void afterInserting(DataBookEvent pEvent) throws Exception
{
   pEvent.getChangedDataBook().setValue(COLUMN, VALUE);
}
```