Table of Contents

Neben der Berücksichtigung von Default Werten ist die Einschränkung auf “erlaubte Werte” ein weiterer Pluspunkt von JVx.

Die “erlaubten Werte”, für Spalten, werden in der Datenbank üblicherweise mit Check Constraints definiert. Diese werden von JVx ausgewertet und als sogenannte erlaubte Werte übernommen.

Die erlaubten Werte wirken sich im User Interface unmittelbar aus und zwar bekommt der Anwender nur die erlaubten Werte in Form von Choice Cell Editoren angeboten.

Anwendungsbeispiel

Aufbauend auf dem Beispiel für Default Werte, definieren wir folgende Check Constraints (Oracle Syntax):

```
ALTER TABLE USERS
  ADD CONSTRAINT USER_ACTIVE_CHECK
  CHECK (active IN ('Y', 'N'));

ALTER TABLE USERS
  ADD CONSTRAINT USER_CHANGE_PASSWORD_CHECK
  CHECK (CHANGE_PASSWORD IN ('Y', 'N'));
```

Folglich dürfen die Felder “ACTIVE” und “CHANGE_PASSWORD” nur “Y” oder “N” enthalten.

Damit im User Interface auch der korrekte Choice Cell Editor verwendet wird, muss dieser natürlich erst bekannt gegeben werden. Das geschieht global und zwar durch den Aufruf von:

```
UIChoiceCellEditor.addDefaultChoiceCellEditor(editor);
```

Für die Werte “Y” und “N” wurde bereits ein Choice Cell Editor vom ApplicationUtil definiert.

Um die Auflösung von Check Constraints zu ignorieren, können folgende Methoden verwendet werden:

```
//per instance
users.setAllowedValues(false);

//for all instances (static)
DBStorage.setDefaultAllowedValues(false);
```

Wenn in der Datenbank keine Check Constraints verwendet werden, können die erlaubten Werte auch über das API gesetzt werden:

```
users.open();

//sets allowed/possible values
users.getMetaData().getColumnMetaData("ACTIVE").setAllowedValues(new
Object[] {"Y", "N"});
```

From:
https://doc.sibvisions.com/ - **Documentation**

Permanent link:
https://doc.sibvisions.com/de/jvx/server/storage/dbcheck_constraints

Last update: **2018/02/02 12:58**