Table of Contents

In addition to the use of default values, the restriction to **“allowed values”** is another advantage of JVx.

The “allowed values” for columns are usually defined in the database using check constraints. These are evaluated by JVx and assumed as allowed values.

The constraints directly impact the user interface, as the user is only offered the allowed values in the form of choice cell editors.

Example

Based on the example for default values, we define the following 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'));
```

Therefore, the fields “ACTIVE” and “CHANGE_PASSWORD” can only contain “Y” or “N”.

So that in the user interface the correct choice cell editor is used, this must be defined. This is done globally by calling:

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

A choice cell editor from the ApplicationUtil has already defined the values “Y” and “N”.

The following methods can be used to disable the check constraints detection:

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

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

If no check constrains are used in the database, the allowed values can be set via the API:

```
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/jvx/server/storage/dbcheck_constraints

Last update: **2020/06/29 13:32**