Table of Contents

Die Metadaten sind ein wesentlicher Bestandteil des JVx Models. Jede Spalte die in einem DataBook oder Editor verwendet wird, hat eine ColumnDefinition im Hintergrund. Damit wird gesteuert welcher Editor verwendet werden soll (Datum, Text, Nummer, Auwahl, ...), von welchem Datentyp die Spalte ist, wieviele Zeichen eingegeben werden dürfen usw. Eine ColumnDefinition wird aus den Metadaten erzeugt. Die Metadaten selbst werden serverseitig erstellt.

Im einfachsten Fall, werden die Metadaten aus der Datenbank ausgelesen, sofern eine Datenbank zum Einsatz kommt. Die Erzeugung der Metadaten wird vollautomatisch durchgeführt und bedarf im Normalfall keiner Änderung.

Es kann aber in bestimmten Situationen erforderlich sein, den Standard Mechanismus an die eigenen Bedürfnisse anzupassen um ggf. zusätzliche Informationen zu speichern, Standardwerte abhängig von eigenen Algorithmen zu setzen oder ev. die Spaltenbezeichner speziell vorzubelegen.

Um mit eigenen Metadaten zu arbeiten, muss das Standardverhalten verstanden werden:

Die Klasse DBAccess ist verantwortlich für die Ermittlung der MetaDaten von Datenbank Tabellen und Views. Diese bietet auch die Möglichkeit in die Metadaten Ermittlung einzugreifen um die eigenen Anforderungen umzusetzen.

Bei DBAccess handelt es sich um eine Server-seitige Klasse. Die damit erzeugten Metadaten werden bei Bedarf zum Client übertragen und das Model kümmert sich um die Konvertierung.

Die Erzeugung von benutzerdefinierten Metadaten wird mit folgendem Beispiel erklärt.

Wir wollen die Metadaten um ein zusätzliches Feld columnInfo erweitern. Darin werden Detailinformationen zum Feld gespeichert.

1. Erstellen einer eigenen Implementierung von ColumnMetaData und ColumnDefinition

```
public static class MyColumnMetaData extends ColumnMetaData
{
    private String columnInfo;

    public String getColumnInfo()
    {
        return columnInfo;
    }

    public void setColumnInfo(String pColumnInfo)
    {
        columnInfo = pColumnInfo;
    }

    public ColumnDefinition createColumnDefinition() throws ModelException
    {
        MyColumnDefinition columnDef = new MyColumnDefinition();
        initializeColumnDefinition(columnDef);

        columnDef.setColumnInfo(columnInfo);

        return columnDef;
```

```
    }
}

public static class MyColumnDefinition extends ColumnDefinition
{
    private String columnInfo;

    public String getColumnInfo()
    {
        return columnInfo;
    }

    public void setColumnInfo(String pColumnInfo)
    {
        columnInfo = pColumnInfo;
    }
}
```

Wie bereits zu erkennen ist, sind beide Klassen miteinander verknüpft. Die Klasse ColumnMetaData erzeugt eine Instanz von ColumnDefinition.

Im **2. Schritt** konfigurieren wir ein DBAccess Objekt um unsere neuen Klassen zu verwenden:

```
DBAccess dba = DBAccess.getDBAccess("<jdbcurl>");
dba.setUsername("<username>");
dba.setPassword("<password>");
dba.open();

dba.setColumnMetaDataCreator(new IColumnMetaDataCreator()
{
    public ColumnMetaData createColumnMetaData(DBAccess pDBAccess,
                                               ResultSetMetaData
pResultSetMetaData,
                                               int pResultSetColumnIndex)
    {
        MyColumnMetaData cmd = new MyColumnMetaData();
        pDBAccess.initializeColumnMetaData(pResultSetMetaData,
pResultSetColumnIndex, cmd);

        cmd.setColumnInfo(cmd.getName() + "(" + cmd.getLabel() + ")");

        return cmd;
    }
});
```

Wir implementieren dazu lediglich das Interface IColumnMetaDataCreator und liefern unsere eigene Klasse, die wir zuvor noch um die columnInfo ergänzen.

Von nun an erzeugt die DBAccess Instanz sämtliche Metadaten mithilfe des gesetzten `ColumnMetaDataCreator`.

Die erzeugte MyColumnMetaData Instanz wird wie gewohnt zum Client übertragen. Das Model erzeugt weiterhin vollautomatisch unsere MyColumnDefinition Instanz, durch den Aufruf von createColumnDefinition. Sie müssen sich um keine weiteren Details kümmern.