camos.

camos Develop
Developer training

Basics SQL

Prerequisites

- **Carconfigurator on the state 3. day modeler training according to training documents**
  - Especially the class Footer – if it is missing, see last slide
- **Database „OfferData.mdb“**
- **DSN = „DataCarconfigurator“**

- **Contents**
  - Form element DB-table
    - Show car dealerships
  - Transmitting Select-statements
    - Reading in address data

Training targets

- **After these exercises you should…**
  - Be able to establish a connection to a database
  - Display data from the database on the form
  - Read in data from the database to the application

Database communication

- **Establishing a connection via SQLConnect()**
  - Via a DSN in the ODBC-manager
  - Via a free defineable connection string

- **The ODBC-handle that is provided by SQLConnect() is required for all database operations**
  - SQLExec() executes the transferred SQL-statement
  - SQLNext() reads the by a Select-statement determined values into features
  - SQLWriteBin() and SQLReadBin() for writing and reading of BLOBs
  - Display of database values in the form element „DB table“

Exercise: Form element DB-table

- **Target**
  - A car dealership has to be selected before an offer can be generated
- **Note**
  - The possible car dealerships are in the table „Supplier“ of the Access database „OfferData.mdb“
- **Procedure**
  - 1) Display all car dealerships in a table
  - 2) Read in address data of the desired car dealership
  - 3) Display this address data as „Footer“ in the result

Exercise: Database connect

- **Connect to database**
  - Create the numerical feature ODBCHnd in class „start“
  - Create the method DBConnect()

```
# Establish connection to the database
ODBCHnd := SQLConnect('DSN=DataCarconfigurator');
# With unsuccessful connect -> Display error message
# and return 0
IF ODBCHnd THEN
  RETURN 1;
ELSE
  WinMessage('ERROR', GetLastError());
  RETURN 0;
ENDIF;
```

Exercise: Database connect

- **Call DBConnect() in new()**
  - The form should only be opened if the connect to the database was successful:

```
IF DBConnect() THEN
  WinOpen('MainForm');
ENDIF;
```

- **Method Delete() in order to close the database connection with exiting the application:**

```
SQLDisconnect(ODBCHnd);
```

- **Test the database connect**

Form element DB-table

- **Read out car dealerships from the database:**
  - The form element DB table displays the contents of any table columns of the database

  - The columns of the DB table have to be named like the columns in the database or like the alias that is used in the SELECT-statement

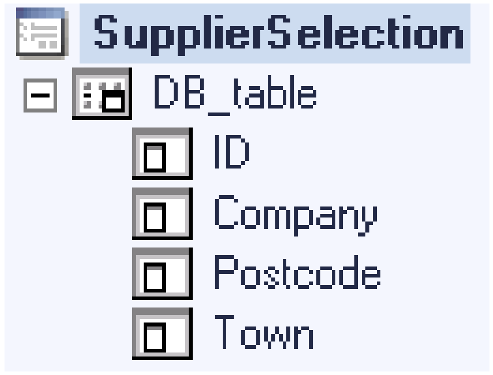
  - In order to be able to flag a line in the table, a Selected feature has to be deposited on the primary key column and the option „Is part of key“ has to be enabled

Exercise: Form element DB-table

- **Create the form „SupplierSelection“ in „start“**
  - Create a new „DB table“
    - Enter the feature „ODBCHnd“ in the field „SQL handle“

  - Create a DB column under the DB-table
    - Name: ID, Column type: Numeric
    - Create numeric feature „SelectedID“ and enter it in „Selected“
    - Enable option „Is part of key“, disable option „Visible“

  - Add three further DB columns
    - 1. Name: Company, column type: String
    - 2. Name: Postcode, column type: Numeric
    - 3. Name: Town, column type: String



Exercise: Form element DB-table

- **Formulate SELECT-statement**
  - Deposit the following on the tab page „SQL“ of the DB-table

```
SELECT * FROM Supplier
```

- **Add a pushbutton „Cancel“**

```
WinClose(WinGetHandle(), 0);
```

- **Add a pushbutton „OK“**

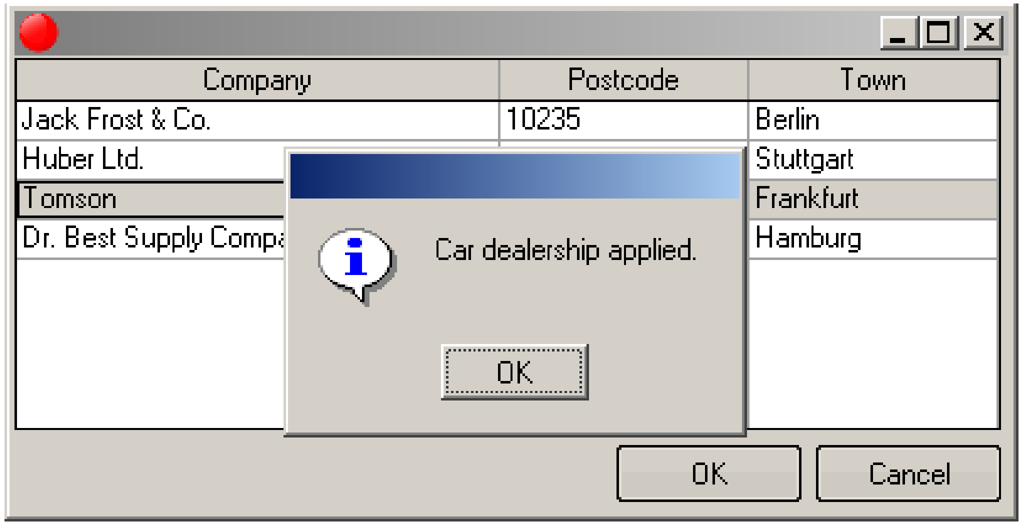
```
# If a data record was selected in the DB-table
IF selectedID THEN
  # Provisional dummy action
  WinMessage('INFO', 'Car dealership applied.');
  WinClose(WinGetHandle(), 1);
ELSE
  WinMessage('ERROR', 'No car dealership selected!');
ENDIF;
```

Exercise: Form element DB-table

- **Extend the menu item „Quotation“**
  - The form SupplierSelection is opened before the offer is opened:

```
IF WinStartModal(WinOpen('SupplierSelection')) THEN
  WinStartModal(WinOpenDoc('Quotation', 0, 0, 800, 600));
ENDIF;
```

- **Test the displaying of the supplier table**



Company	Postcode	Town
Jack Frost & Co.	10235	Berlin
Huber Ltd.		Stuttgart
Tomson		Frankfurt
Dr. Best Supply Compa		Hamburg

Car dealership applied.

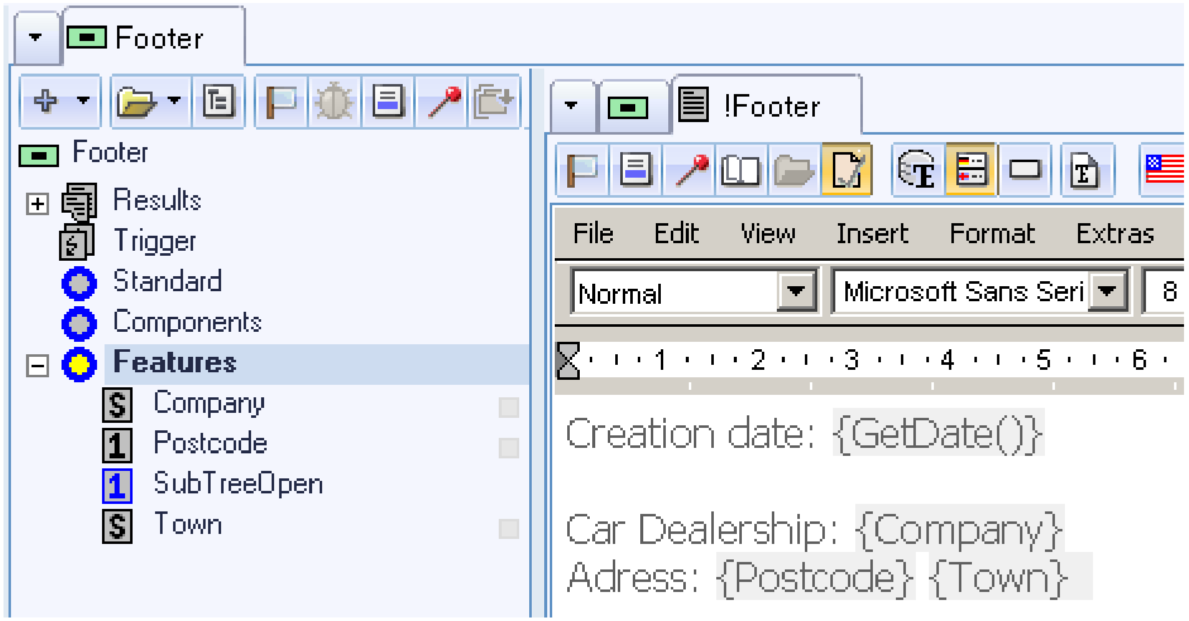
Exercise: Transmit Select-statement

- **Target**
  - Address data of the selected car dealership should be displayed in the quotation
  - To do so, the address data is read in from the database to features
  - Features are integrated as cause variables in RTF-constant

- **Procedure with the reading out of values from the database**
  - Formulate SELECT-statement
  - Transmit statement via SQLExec()
    - The database holds found values in an internal table
  - Read in values via SQLNext() to features
  - Close connection to temporary table

Exercise: Transmit Select-statement

- Display address data of the selected car dealership in the result
- The class „Footer“ is used
- Display the data of the selected car dealership instead of „VW Müller from Stuttgart“
- Create features in class „Footer“ and integrate them as cause variables in the constant !Footer:



Creation date: {GetDate()}

Car Dealership: {Company}
Adress: {Postcode} {Town}

Exercise: Transmit Select-statement

- **In class „Footer“ you create:**
  - Predecessor component on „start“

  - Method ReadAddressDataFromDB()
    - Variables:
      - statement (String)
      - count (numeric)
      - stmtHandle (numeric)
    - Return value:
      - numeric

Exercise: Transmit Select-statement

- Method ReadAddressDataFromDB():

```
statement := "Select Company, Postcode, Town FROM Supplier WHERE
   ID = |@start.SelectedID|";
#
IF SQLExec(@start.ODBCHnd, statement, count, stmtHandle)THEN
  SQLNext(stmtHandle);
ELSE
  WinMessage("ERROR", GetLastError());
ENDIF;
  SQLCloseHandle(stmtHandle);
RETURN;
```

Exercise: Transmit Select-statement

- **The predecessor component is required for the access to**
  - the handle to the database
  - the ID of the data record that is selected in the DB-table (SelectedID)

- **Cause variables in the SQL-statement are masked with pipes**
  - Insert pipe: Alt Gr + key with angle brackets
  - Exceptional feature: ||Cause variable| -> Adaptation of the data type

- **If the DB-column is not named like the target feature, an alias (Postcode AS ZIP) has to be used in the SQL-statement**

Exercise: Transmit Select-statement

- Call this method in the pushbutton „OK“ on the form „SupplierSelection“ instead of the WinMessage:

```
IF SelectedID THEN
  _Car._Footer.ReadAddressDataFromDB();
  WinClose(WinGetHandle(), 1);
ELSE
  WinMessage('ERROR', 'No car dealership selected!');
ENDIF;
```

- Test the application

Create class Footer

- Create the object class „Footer“ under Configuration
- Create a component of „Footer“ in „Car“ and initialize it
- Enable the properties „Sort“ and „Output to result“
- In the dialog „Sort of components“ in „Car“ you move the component _Footer to the last position
- In the class „Footer“ you create the RTF-constant !Footer with the contents:

  Creation date: {GetDate()}
  VW Müller
  Schwabstr. 23
  70123 Stuttgart

- Assign the constant to the quotation in the structure tree