camos.

camos Develop
Developer training

camos.Toolbox: XMLReader and XMLWriter

camos.

XML with camos.Toolbox

- **Prerequisites**
  - Knowledge base „CarConfigurator" at the end of the 3. day of the modeler training
  - Knowledge base camos.Toolbox (with camos.Basic as library)
  - camos.Toolbox has to be a library of Carconfigurator
  - All three knowledge bases have to use a mixed frame (training frame + camosStandard frame)

- **Contents**
  - Use of the XML-functions of the camos.Toolbox
    - XMLWriter -> Save offer as XML
    - XMLReader -> Read-in XML and show information

camos.

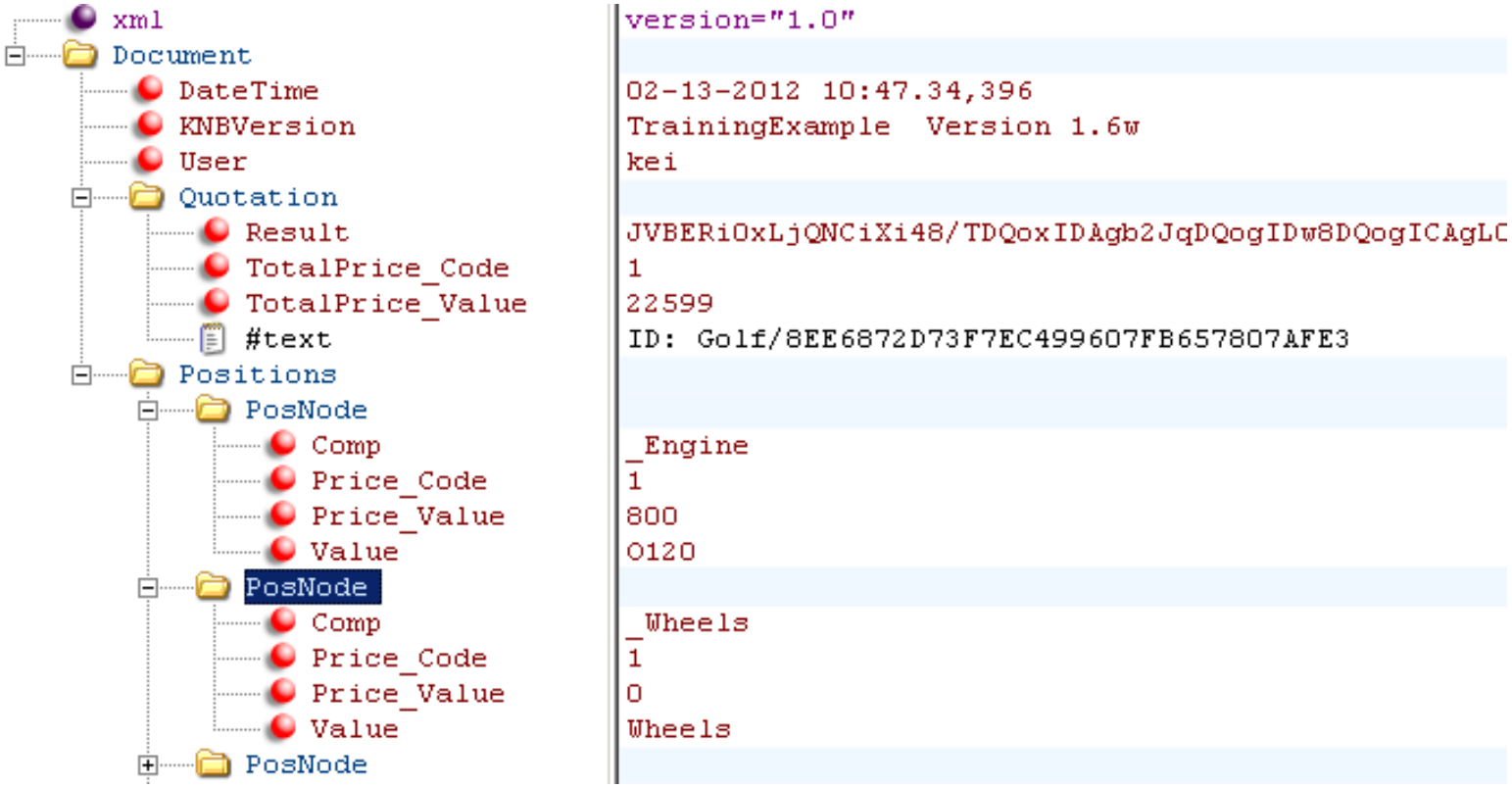
General information for XML

- **XML in camos Develop**
  - Within camos Develop a XML is only a string
  - The coding is only specified with the saving, e.g. UTF8
  - There are various XML-functions via which you can navigate in the tree structure of the XML (DOM)
  - All imports/exports in camos Develop are carried out via XML, e.g. SysClass2XML, OT2XML, FrameXML, etc.
  - The library camos.Toolbox provides two interfaces for easy writing and reading, following the .Net-Framework implementation

- **Online Help: /Function reference/XML Functions/General information**

Exercise: XMLWriter

- **Target**
  - A XML with the most important information for the current configuration has to be created



camos.

Exercise: XMLWriter

- **In the class start:**
  - Create a component of the library class cT_XMLWriter: „_XMLWriter"
  - Create a new menu trigger „Save offer as XML" in the menu „Administration"
  - In the field „Enabled" you deposit: `_Car <> NOVALUE`
  - Create the method CreateXML() and call this method in the new menu item

- **In the further course you need a method that provides the component name of an object**
  - Create the method GetCompName() in the class „Modules", remove side effects
  - Numeric value parameter „mode_", return type String

```
Return GetMyCompName(mode_);
```

camos.

Exercise: XMLWriter

- **Via the methods of the XMLWriter a XML has to be created that contains following data**
  - Date and time
  - Creator
  - Version of the used configurator
  - Offer-ID (Car type + HexGUID)
  - End price of the model
  - Result of the offer as PDF
  - All selected modules incl. value and price

- **The XML has to be saved on the harddisk (C:\Temp\Offer.xml)**

Exercise: XMLWriter

- **Create the following variables in start, method CreateXML()**
  - i, iMax, filehandle (numeric)
  - xml (string)
  - pModules[] (object pointer list on „Modules")
- **Source code line 1-8 (incl. comments)**
  - Date & time, user and version as attributes of „Document"

```
_XMLWriter := 'cT_XMLWriter';
_XMLWriter.Create();
#
_XMLWriter.WriteStartElement('Document');
_XMLWriter.WriteAttributeString('DateTime', GetDate('UTC'));
_XMLWriter.WriteAttributeString('User', GetUser());
_XMLWriter.WriteAttributeString('KNBVersion', GetVersion('KNB'));
#
```

Exercise: XMLWriter

- **Source code line 9-15 (incl. comments)**
  - Tag „Quotation" with attributes for the generated ID, the price (separated in the numerical value and the currency code) as well as the result as Base64-string (for binary PDF)

```
_XMLWriter.WriteStartElement('Quotation');
_XMLWriter.WriteString('ID: ' & _Car & '/' & CreateGUID());
_XMLWriter.WriteAttributeString('TotalPrice_Value',
   Currency2Num(_Car.EndPrice, GetCurrency()));
_XMLWriter.WriteAttributeString('TotalPrice_Code',
   GetCurrency());
_XMLWriter.WriteAttributeString('Result',
   Bin2Base64(RTF2PDF(GetDoc('Quotation'))));
# End Quotation-tag
_XMLWriter.WriteEndElement();
```

Exercise: XMLWriter

- **Source code line 16-31 (incl. comments)**
  - Determine pointers to all module objects and write value & price for each object to a „PosNode" element

```
# Write positions list
_XMLWriter.WriteStartElement('Positions');
pModules[] := ObjChildGet(_Car, ,Modules', {'R', 'Sort:'});
iMax := MaxIndex(pModules[]);
FOR i := 1 TO iMax DO
_XMLWriter.WriteStartElement('PosNode');
_XMLWriter.WriteAttributeString('Comp', pModules[i].GetCompName(0));
_XMLWriter.WriteAttributeString('Value', pModules[i]);
_XMLWriter.WriteAttributeString('Price_Value',
   Currency2Num(pModules[i].Price, GetCurrency()));
_XMLWriter.WriteAttributeString('Price_Code', GetCurrency());
_XMLWriter.WriteEndElement();
ENDFOR;
# End Positions
_XMLWriter.WriteEndElement();
# End Document
_XMLWriter.WriteEndElement();
#
```

Exercise: XMLWriter

- **Source code line 32-44 (incl. comments)**
  - Generate XML and save in file

```
xml := _XMLWriter.XMLGetDocumentString();
# Open file writing for UTF8 coding
filehandle := FileOpen('[Client]C:\Temp\Offer.xml', 'W', 'UTF8');
IF filehandle THEN
  IF FileWrite(filehandle, xml) THEN
    FileClose(filehandle);
    WinMessage('INFO', 'XML successfully generated and saved!');
    _XMLWriter := NOVALUE;
  ENDIF;
ELSE
  WinMessage('ERROR', 'Error with saving the XML!^n' &
   GetLastError());
ENDIF;
RETURN;
```

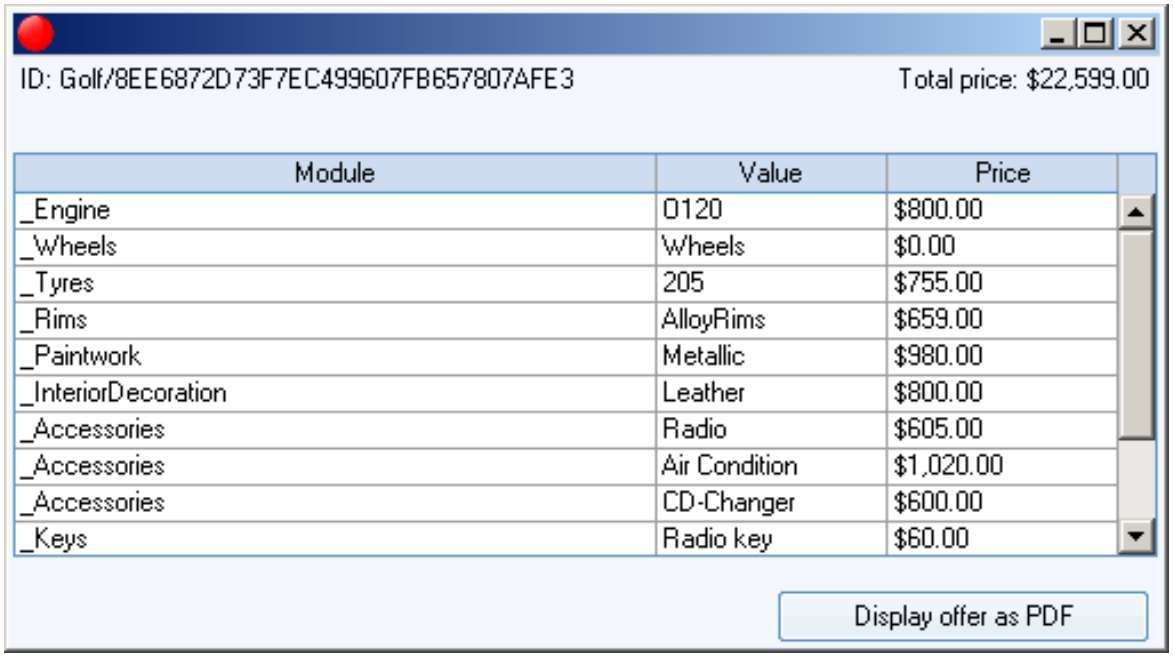
camos.

Exercise: XMLWriter

- **Start the configurator and test the creating of the XML**

Exercise: XMLReader

- **Target**
  - The saved XML has to be read-in again
    - Displaying the ID and the total price
    - Displaying the selected modules in a table
    - Opening the offer-PDF



ID: Golf/8EE6872D73F7EC499607FB657807AFE3 Total price: $22,599.00

Module	Value	Price
_Engine	0120	$800.00
_Wheels	Wheels	$0.00
_Tyres	205	$755.00
_Rims	AlloyRims	$659.00
_Paintwork	Metallic	$980.00
_InteriorDecoration	Leather	$800.00
_Accessories	Radio	$605.00
_Accessories	Air Condition	$1,020.00
_Accessories	CD-Changer	$600.00
_Keys	Radio key	$60.00

Display offer as PDF

Exercise: XMLReader

- **In the class „start"**
  - Create a component of the library class „cT_XMLReader": „_XMLReader"
  - Create the method ReadXML()
  - Create the menu trigger „Read in offer-XML" in the menu „Administration" and call the method ReadXML() in this menu item

  - Create the following features

camos.

Exercise: XMLReader

- **The following variables are needed in the method ReadXML()**
  - i, filehandle, tCurValue, tCurCode (numeric)
  - xml (string)

- **Source code line 1-9 (incl. comments)**
  - Open saved XML-file and read in to the variable XML

```
filehandle := FileOpen('[Client]C:\Temp\Offer.xml', 'R', 'UTF8');
IF filehandle THEN
  FileReadAll(filehandle, xml);
  FileClose(filehandle);
ELSE
  WinMessage('ERROR', 'Error with opening the XML!^n' &
   GetLastError());
  RETURN 0;
ENDIF;
#
```

Exercise: XMLReader

- **Source code line 10-21 (incl. comments)**
  - Generate DOM, jump to quotation node, read in attributes

```
_XMLReader := 'cT_XMLReader';
_XMLReader.Create(xml);
# jump to the quotation node
_XMLReader.ReadToFollowing('Quotation');
# read in ID and result
Offer_ID := _XMLReader.ReadElementContentAsString();
Offer_Quotation :=
   Base642Bin(_XMLReader.GetAttributeByName('Result'));
# read in currency value and currency code, calculate price
tCurValue :=
   String2Num(_XMLReader.GetAttributeByName('TotalPrice_value'));
tCurCode :=
   String2Num(_XMLReader.GetAttributeByName('TotalPrice_Code'));
Offer_EndPrice := Num2Currency(tCurValue, tCurCode);
#
```

camos.

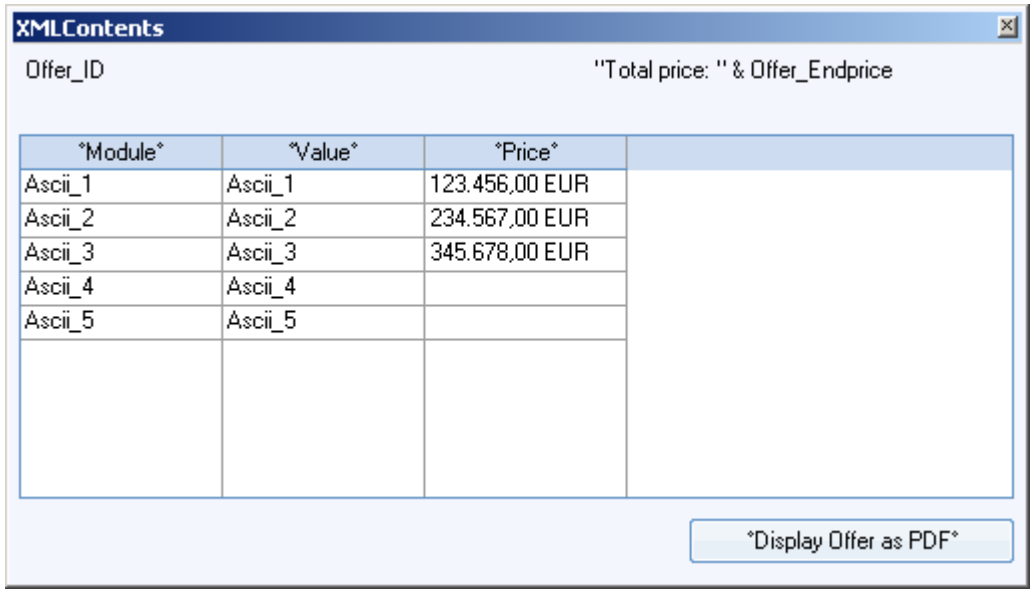
Exercise: XMLReader

- **Source code line 22-35 (incl. comments)**
  - Jump to PosNode elements, read in attributes, delete DOM, open form

```
# Jump to each PosNode node and read out the attributes
WHILE _XMLReader.ReadToFollowing('PosNode') DO
  i := i + 1;
  Offer_Position[i] := _XMLReader.GetAttributeByName('Comp');
  tCurValue :=
   String2Num(_XMLReader.GetAttributeByName('Price_value'));
  tCurCode :=
   String2Num(_XMLReader.GetAttributeByName('Price_Code'));
  Offer_Price[i] := Num2Currency(tCurValue, tCurCode);
  Offer_Value[i] := _XMLReader.GetAttributeByName('Value');
ENDWHILE;
#
_XMLReader := NOVALUE;
#
WinStartModal(WinOpen('XMLContents'));
RETURN;
```

Exercise: XMLReader

- **Create the form „XMLContents"**
  - ID and Endprice are displayed in a dynamic label
  - The lists Offer_Position[], Offer_Value[] and Offer_Price[] are displayed in a table



Exercise: XMLReader

- **The pushbutton „Display offer as PDF" has the following code in the selection trigger:**
  - Local variable filehandle (numerical)

```
filehandle := FileOpen('[Client]C:\Temp\tmpResultAsPDF.pdf',
   'WB');
IF filehandle THEN
  FileWriteBin(filehandle, Offer_Quotation, 'RAW');
  FileClose(filehandle);
  Call('[Client]cmd /C C:\temp\tmpResultAsPDF.pdf &$');
ELSE
  WinMessage('ERROR', 'Error with writing the PDF-file!^n' &
   GetLastError());
ENDIF;
```

- **Start the application and test the reading in of the XML**