

camos.

**camos Develop
Developer training**

Libraries

- **Prerequisites**

- Knowledge base „Carconfigurator“ at the end of the 3rd day of the modeler training
- Knowledge bases camos.Toolbox and camos.Basic (= camos libraries) with camosStandard frame
- Knowledge base FrameApplication (present as packet export)

Libraries

- **Contents**
 - Libraries
 - Advantages
 - Use
 - Customizing possibilities
 - Exercise 1: Integrate the configurator in FrameApplication
 - Customizing of forms and methods from library classes
 - Exercise 2: Integrate camos.Toolbox in the configurator
 - File dialog for saving/loading the configuration

Training targets

- **After these exercises you should...**
 - Know advantages of libraries
 - Import knowledge bases and integrate as library
 - Use library classes
 - Use virtual classes for the customizing of library classes

Libraries

- **Libraries are used...**
 - to use already existing knowledge, e.g. file dialogs, conversion routines
 - to unite several knowledge bases, e.g. three configuration applications in one form
- **Principle also used in camos applications**
 - camos Selling, camos Quotation, camos SalesCenter etc. consist of individual modules (knowledge bases) that are connected to each other via a library structure
- **Attention**
 - All libraries have to use the same frame!

Libraries: Customizing

- **Customizing = Changing the behavior of the library**
- **Wikipedia:**
- Customizing is the expression for the adaptation of a serial product such as e.g. a vehicle or a software to the necessities of a customer.

Libraries: Customizing

- **Advantages of the customizing:**
 - Deriving a local class from the library class
 - Creating a virtual class from the library class and deriving a local class from the virtual class
 - In some libraries (e.g. camos Selling, camos Quotation) the starting-point for a customizing is already preset in the existence of virtual classes without real reference. Via creating and outprogramming the real class, the developer can directly affect the behavior of the class that is derived from the virtual class.
- **The original class in the library knowledge base is not changed with this procedure!**

Libraries: Customizing





- **Notes for the customizing of camos libraries**
 - The library classes that are provided by camos are not allowed to be changed!
 - Reason: The libraries are further developed by camos. All overloads and changes get lost with the import of a later version!
 - As far as virtual classes do not already exist, the recommended procedure with the customizing of camos library classes is to derive a local class and to adapt this class.

Libraries: Allocation













- **There are the following possibilities to integrate a library**
 - Latest version
 - Always the work- or release version with the highest version number is integrated. If a new version of this library is generated, this version is used automatically.
 - Latest release
 - Always the release version with the highest version number is integrated. If a new release version of this library is generated, this version is used automatically.
 - Selected version
 - The work- or release version is integrated that was defined with the allocation. If a new version of this library is generated, the allocation does not change.

Libraries: Allocation

- In the KNB-properties is displayed which version was defined and which version is actually used
 - Example 1:

Libraries	Naming	Used	Defined
  camos.Toolbox		 10.2	* 












- Example 2:

Libraries	Naming	Used	Defined
  camos.Toolbox		 10.9	* 
 camos.Basic		 9.6	10.9  
 camos.Basic		 9.6	9.6  

Libraries: Conflicts

- **Version conflict**

- If a library is integrated several times in different versions (via another library), this conflict is indicated by the icon 
- This conflict should be recovered

Libraries	Naming	Used	Defined	
 camos.Toolbox		 10.9	*	
 camos.Basic		 9.6	10.9	 
 camos.Basic		 9.6	9.6	 

Different versions in the definition of the libraries

- **Load conflict**

- When there are classes with the same name -> the classes of the library are not integrated
- Displays the duplicate classes in a dialog


Exercise: Library: FrameApplication

Application pool

Application Language Quotation

Applications
Car configurator

Model selection



Price Details

List price	20.934,00 €
Discount	3.0 %
End price	20.305,98 €

Current Configuration

	Price
✓ New Beetle Cabriolet	
✓ Soft Top	0,00 €
✓ Otto engine 120	800,00 €
✓ Wheels	0,00 €
✓ 155 tyres	400,00 €
✓ Steel rims	500,00 €
✓ Regular paintwork	730,00 €
✓ Leather interior	800,00 €
✓ Radio	605,00 €
✓ Air conditioning	1.020,00 €
✓ Remote control key	60,00 €
✓ Emergency key	20,00 €

✓ Engine & Wheels ✓ Interior Decoration ✓ Accessories

Engine

Engine	Power	Price
✓ Otto		
✓ <input type="radio"/> Otto engine 50	50	350,00 €
✓ <input checked="" type="radio"/> Otto engine 120	120	800,00 €
✗ <input type="radio"/> Diesel engine 70	70	330,00 €
✗ <input type="radio"/> Diesel engine 110	110	750,00 €

✓ Steel rims 500,00 €

Possible engines

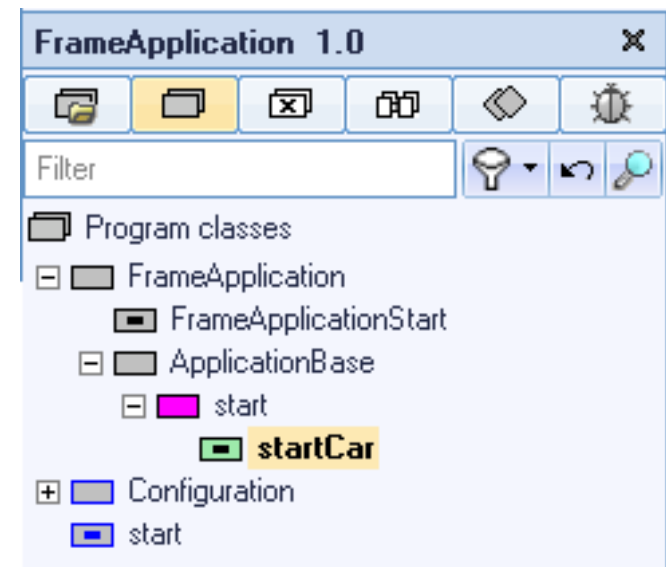
	Price
<input type="radio"/> Ø	
✓ <input checked="" type="radio"/> 155 tyres	400,00 €
✓ <input type="radio"/> 175 tyres	542,00 €
✓ <input type="radio"/> 185 tyres	680,00 €
✓ <input type="radio"/> 205 tyres	755,00 €

Exercise: Library: FrameApplication

- **Integrate carconfigurator in the frame application**
 - Import the knowledge base `FrameApplication.pgx`
 - Create a new frame „MixedFrame“. Mix the „MixedFrame“ with the frames „TrainingFrame“ and „camosStandard“
 - Assign the „MixedFrame“ to the knowledge bases „FrameApplication“ and „TrainingExample“
 - Integrate the knowledge base „TrainingExample“ as library in the knowledge base „FrameApplication“ (selected version)

Exercise: Library: FrameApplication

- **Customizing the class „startCar“**
 - Under the class „ApplicationBase“ create a virtual class from the class „start“ of the carconfigurator. Under this you create a new object class „startCar“.
 - Give the class „startCar“ a meaningful naming, e.g. „Car configurator“
 - Overload the method new()
 - Remove the WinOpen()



Exercise: Library: FrameApplication

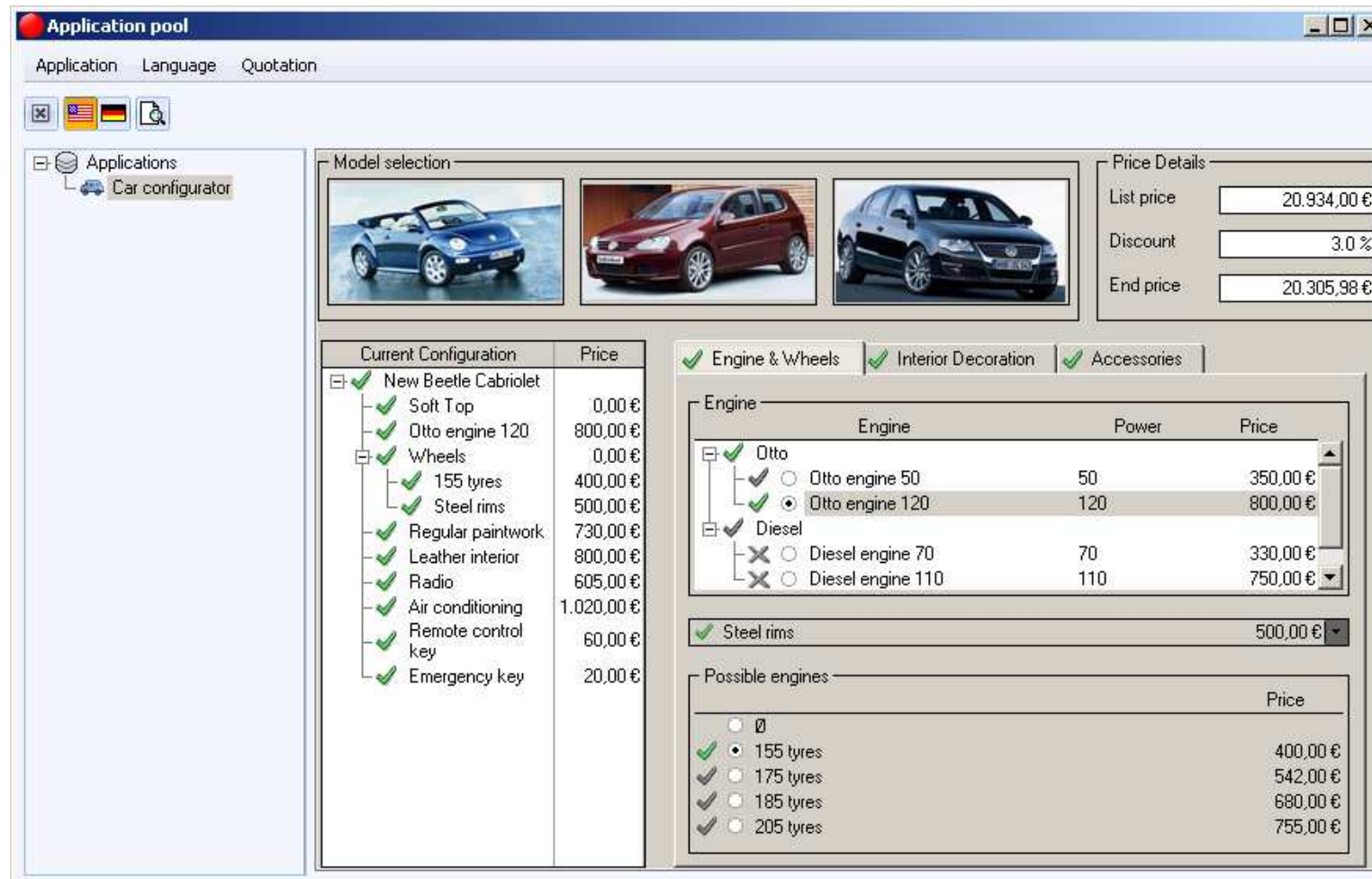
- **Customizing the class „startCar“**
 - Overload the form „Form“
 - Remove the inherited dynamic label
 - Give the form „Form“ the same size as the size of „Main form“ of the carconfigurator
 - Create a dynamic subform of the same size
 - Assign the form „MainForm“ and the object „Self“ to the subform
 - Overload the menu „MainMenu“
 - Create a dynamic submenu
 - Assign „Administration“ as submenu and „Self“ as object
 - Enable „Icon in toolbar“
 - Insert a separator between the two icons

Exercise: Library: FrameApplication

- **Customizing the class „startCar“**
 - Optionally you can overload the constant !Icon and load the graphic file Carconfigurator.gif
- **Test by starting the class „FrameApplicationStart“**
 - The dialog Application pool is opened
 - All integrated applications are displayed in the left navigation tree
 - The main form of the flagged application is displayed in the right splitter window (in this case the main form of the carconfigurator)
 - The menu and the toolbar contain the Exit icon as well as the menu items and icons of the carconfigurator

Exercise: Library: FrameApplication

- You are welcome to integrate further applications!



Exercise: Library: camos.Toolbox

- **camos provides various libraries**
 - XSysInterface for editing knowledge bases
 - camos.DocViewer for creating an online help
 - camos.Toolbox contains various selection dialogs (file, color, folder...)
 - camos.Basic contains icons
 - TxMInterface for communicating with the text manager
- **Exercise: File selection dialog from camos.Toolbox**
 - The configuration has to be saved
 - User can select storage location and storage file via file dialog
 - -> Library class camosToolbox_DLGFile

Exercise: Libraries: camos.Toolbox

- **Import the knowledge base camosToolbox.pgx**
 - Attention: Mix frame „camosStandard“ from PGX with „TrainingFrame“ or use „MixedFrame“!
- **Integrate the camos.Toolbox as library in the carconfigurator**
 - Selected version
- **In „start“ you create a component of the class „camosToolbox_DLGFile“**
 - Change the name of the component to „_DLG“

Exercise: Libraries: camos.Toolbox

- **Create the method Save()**
 - Variables: Filter_[] (String), File_ (String), fhandle (num)

```
# Create object of the component, initialize features
_DLG := 'camosToolbox_DLGFile';
Filter_[] := {'Configuration files', 'sot'};
File_ := '[Client]C:\Temp\Car.sot';
#
# if a file was selected
IF _DLG.SaveFile(File_, Filter_[]) THEN
# Open file writing, write blob, close file
  fhandle := FileOpen(File_, 'WB');
  IF FileWriteBin(fhandle, OT2Bin(_Car), 'RAW') THEN
    WinMessage('INFO', 'Configuration successfully saved!');
  ELSE
    WinMessage('ERROR', GetLastError());
  ENDIF;
  FileClose(fhandle);
ENDIF;
```

Exercise: Libraries: camos.Toolbox

- **Create the title „Configuration“ in the menu „Administration“**
 - Create a new menu item „Save as...“
 - Use !SaveIcon.bmp as icon of the menu item
 - Enable „Icon in toolbar“
 - Call the method Save() in the menu item
- **Define the following expression in the field „Enabled“:**
`_Car <> NOVALUE`
- **Test the saving of the configuration!**

Exercise: Libraries: camos.Toolbox

- **The saved blobs have to be loaded too**
 - The selection of the file is carried out again via the file selection dialog
- **Create the menu item „Load...” under the title „Configuration”**
 - Use !LoadIcon.bmp as icon of the menu item
 - Enable „Icon in toolbar”

Exercise: Libraries: camos.Toolbox

- Create the method Load()
 - Variables Filter_[] and File_ (String), binVar (binary), fhandle (num)

```
# Create object of the component, initialize features
_DLG := 'camosToolbox_DLGFile';
Filter_[] := {'Configuration files', 'sot'};
File_ := '[Client]C:\Temp\Car.sot';
# if a file was selected ...
IF _DLG.OpenFile(File_, Filter_[]) THEN
    # Open file, read blob
    fhandle := FileOpen(File_, 'RB');
    binVar := FileReadBin(fhandle, 'RAW');
    IF binVar = ANYVALUE THEN
        # Restore object tree, close file
        _Car := NOVALUE;
        IF not Bin2OT(binVar, _Car) THEN
            WinMessage('ERROR', GetLastError());
        ENDIF;
    ELSE
        WinMessage('ERROR', GetLastError());
    ENDIF;
    FileClose(fhandle);
ENDIF;
```

Exercise: Libraries: camos.Toolbox

- Call the method Load() in the menu item „Load...”



- Test the correct saving and loading of the configuration