Standard-Funktionen und -Methoden

Modul umbenennen

- **prüfen, ob es den neuen Namen schon gibt**
  - in insights.constants.pas
  - in Dabenbank modules.filename
- Datenbank: Tabelle **modules** Feld **filename**
- Dateiname der dll in IDE ändern
- Verzeichnis unter insights\data anpassen
- im Data-Verzeichnis den *.db Dateinamen anpassen
- im Data-Verzeichnis die *.ini-Datei anpassen
- in dll main ganz am Schluss CodeSite-String anpassen
- in insights.constants.pas **cmi_** und **cmc_** anpassen
- in insights.module.main.pas TinsightsModuleMain.Create **FInfo.Caption** und **FInfo.ID** anpassen

ModulID ändern

- **prüfen, ob es die neue ID schon gibt**
  - in insights.constants.pas
  - in Datenbank modules.id
- Datenbank: Tabelle **modules** Feld **id** und ggf. Feld **catid** anpassen
- in insights.constants.pas **cmi_** anpassen
- ggf. in insights.ini Settings LastModuleID anpassen (wenn das geänderte Modul das letzte aktive Modul war)

Nachrichten und Events

Nachrichten vom Hauptprogramm zum Modul

```
  CurrentModule.NotifyModuleI('DemoModeChanged', IfThen(dxBarButton26.Down,
1, 0));
  CurrentModule.NotifyModuleS('xxx', string);
```

Zum Empfangen der Nachrichten müssen im TInsightsModuleMain die virtuellen Methoden überschrieben werden:

insights.module.base - TinsightsModuleBase

```
    procedure NotifyModuleI(const mInfo: string; mInt: Integer); virtual;
    procedure NotifyModuleS(const mInfo, mStr: string); virtual;
```

insights.module.main - TinsightsModuleMain

```
    procedure NotifyModuleI(const mInfo: string; mInt: Integer); override;
    procedure NotifyModuleS(const mInfo, mStr: string); override;
```

Nachrichten vom Modul zum Hauptprogramm

Um eine Nachricht aus einem Modul zum Hauptprogramm zu schicken, ruft das Modul die Methode Events.OnReceiveIDInt oder Events.OnReceiveIDStr auf. Diese triggern im Hauptprogramm dann

```
      Result.Events.OnReceiveIDInt := formMain.DoReceiveIDInt;
      Result.Events.OnReceiveIDStr := formMain.DoReceiveIDStr;
```

```
procedure TformMain.DoReceiveIDInt(const mEventStr: string; mID: Integer;
var mResult: Integer);
begin
  if sametext('ModuleIsLoaded', mEventStr) then
    mResult := IfThen(FModuleList.ContainsKey(mID), 1, 0);
end;

procedure TformMain.DoReceiveIDStr(const mEventStr: string; mID: Integer;
var mResult: string);
begin
  if sametext('StartPageInitLog', mEventStr) then
    mResult := formInit.cxMemo1.Lines.Text;
end;
```

im Module (TinsightsModuleBase)

Presets

- ActivatePresetsByID Info
  Aktiviert einen Preset für das aktuelle Modul über die PresetID
- CurrentConfigToJson Info
  ...
- CurrentConfigFromJson

TCommon - insights.common

- AddButtonToButtonLayoutGroup Info
  AddButtonToButtonLayoutGroup(dxLayoutGroup, 'btn1', 'btn', 1, 1, GroupButtonClicked);
  Beschreibung

Drag & Drop

- Um das Drag&Drop von Dateien auf ein Modul zu aktivieren, muss man im Modul

TinsightsModuleMain.Create **FInfo.AcceptFiles** auf True setzen.
- Die auf das Formular gedroppten Dateien können dann in der override-Methode **procedure FilesDropped(const mText: string); override;** ausgewertet werden.

```
procedure TinsightsModuleMain.FilesDropped(const mText: string);
begin
  // direkt im Module Code ausführen oder
  // den Event OnFilesDropped des formModuleBase (override im eigentlichen
Formular) ausführen
  if assigned(FConfigForm.OnFilesDropped) then
    FConfigForm.OnFilesDropped(mText);
end;
```

Im ConfigForm kann man dann in InitVars dem OnFilesDropped eine eigene Methode zuweise:

```
procedure TformConfigForm.InitVars;
begin
  inherited;
  OnFilesDropped := FilesDropped;
end;

procedure TformConfigForm.FilesDropped(const mText: string);
begin
  CodeSite.Send(csmGreen, 'mText', mText);
  hsSList(mText, #13#10).ForEachI(procedure(const mStr: string; mInt:
Integer)
    begin
      CodeSite.Send(mInt.ToString, mStr);
    end);
end;
```

From:
https://insights.hschmid-it.de/ - **Insights - ein Programm von hschmid-it.de**

Permanent link:
https://insights.hschmid-it.de/doku.php?id=code:defaultmethods

Last update: **2026/10/02 13:07**