



# Team Developer 7.0

Teil 3

OpenText Workshop

# Agenda



- TD 7.1 Ausblick
  - Multi-Threading
  - Team Features

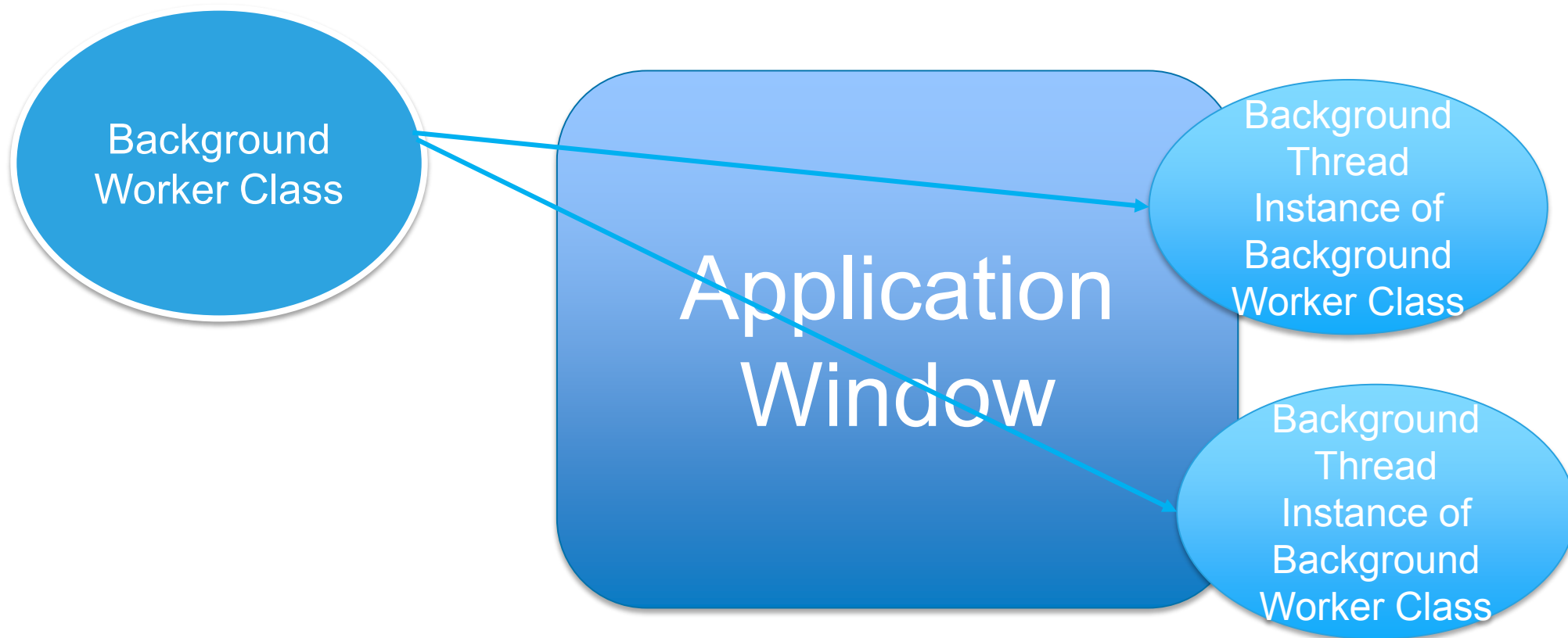
# Multithreading



- Ziel:
  - Definierte Abläufe einer Anwendung sollen im Hintergrund abgearbeitet werden
  - Der Anwender soll weiterhin mit der Anwendung arbeiten können
  - Einfache OOP basierende Implementation
  - Kontrolle und Interaktion mit der Hauptanwendung, basierend auf Events

# Multithreading

- Implementierung



# Multithreading



- Implementierung
  - Definieren einer Background Worker Class
  - Hinzufügen der Background Worker Class in ein (MSI-) Window
  - Innerhalb des Window Objekts
    - Start des Background Thread mit `SalBackgroundWorkerStart(oBkgdWorker)`
    - Der Thread wird gestartet und das Thread Event System wird initialisiert

## ■ Beispiel einer Background Worker Class:

- ◆ Background Worker Class: `bwFileIO`
  - ◇ Description:
  - ◆ Derived From
    - ◇ Class: `fcCreateCSVFromCSV`
  - ◆ Instance Variables
    - ◇ String: `sInFile`
    - ◇ String: `sOutFile`
    - ◇ Number: `nTimes`
  - ◇ Functions
  - ◆ Thread Start
    - ◇ Local variables
    - ◆ Actions
      - ◇ Call `fCreateCSVFromCSV( nTimes, sInFile, sOutFile )`

- **Thread Klassen Definition**
- **Instanz Variablen zum initialisieren des Prozesses**
- **Thread Code**
  - **Lang laufende Database Prozesse**
  - **Lang laufende .NET Reports**
  - **Etc.**
- **Achtung: Keine GUI interaction möglich!**

# Multithreading



Threads File IO

Invoices to new CSV

10

Cancel Thread 1 Execution    Check if Thread 1 is Running  Active

Cancel Thread 2 Execution    Check if Thread 2 is Running  Active

Thread Status

22%

20%

Thread Progress Message

Thread 1: Processed 22%  
Thread 2: Processed 20%  
Thread 1: Processed 21%  
Thread 2: Processed 19%  
Thread 1: Processed 20%  
Thread 2: Processed 18%  
Thread 1: Processed 19%  
Thread 2: Processed 17%  
Thread 1: Processed 18%  
Thread 2: Processed 16%  
Thread 1: Processed 17%

Start Time    10:35:53:708244    10:35:55:210978

End Time    10:35:23:070525    10:35:27:569416

Duration (s)    45,04061100000003840    47,90727599999999040

Query Database    Reset Grid

COMPANY_ID	COMPANY_NAME	ADDRESS	CITY	STATE
101	Clothing Connoisseurs	8 Lombard Street	San Francisco	CA
102	Maui Mu-Mus	1011 Likelike Lane	Honolulu	HI
103	Fashion Outlet	16 E. Pennsylvania Av...	Washington	DC
104	Garcia Tiedye Barn	1965 Shakedown Street	Sausalito	CA
105	Zanzibar Limited	36 Laibon Road	Dar es Salaam	
106	Genevieve Casuals	1313 Minnow Motorway	Honolulu	HI
107	Montana Activewear	2827 Arrowhead Way	Billings	MT

# Multithreading



- \* Form Window: frmThreads
  - ◇ Description:
  - ◇ Ribbon
  - ◇ Named Menus
  - ◇ Menu
  - \* Tool Bar
  - \* Contents
- \* Background Threads:
  - \* bwFileID: oThread
    - \* Thread Events
      - \* Thread Before Start
        - \* Actions
      - \* Thread Report Progress
        - \* Progress Parameters
        - \* Actions
      - \* Thread Finished
    - \* bwFileID: oThread1
  - ◇ Functions
  - ◇ Window Parameters
  - \* Window Variables
    - ◇ Sql Handle: hSql
  - ◇ Message Actions

- Thread Instanzen beinhalten Thread-Events
- `SalBackGroundWorkerStart(ThreadInstance )` startet den Thread
- Thread Events werden ausgelöst



# Event: Thread Before Start

- ◆ Thread Before Start
  - ◆ Actions
    - ◆ Call SalListInsert( lbThreadProgress, 0, "Entering On Thread 1 Before Start." )
    - ◆ Call SalListInsert( lbThreadProgress, 0, "Thread 1 in file = invoice-in.csv, out file = invoice-out.csv." )
    - ◆ Set oThread.sInFile = "invoice-in.csv"
    - ◆ Set oThread.sOutFile = "invoice-out.csv"
    - ◆ Set oThread.nTimes = dfLoops
    - ◆ Call SalPicSet( picThread, orange, PIC\_FormatPNG )
    - ◆ Call SalMeterSetRange( prbThread, 0, 100 )
    - ◆ Call SalMeterSetStepSize( prbThread, 1 )
    - ◆ Call SalMessageBox( "Thread 1 is going to start", "Thread Hint", MB\_Ok )
    - ◆ Call SalPicSet( picThread, green, PIC\_FormatPNG )
    - ◆ Call SalListInsert( lbThreadProgress, 0, "Leaving On Thread 1 Before Start." )

- Initialisierung des Hintergrund Prozesses
- Thread startet am Ende des Events
- Returns FALSE wenn bereits ein Hintergrund Prozess dieser Worker Klasse läuft.

# Event: Thread Report Progress

- ◆ Thread Report Progress
- ◆ Progress Parameters
- ◆ Actions
  - ◇ Call `SaListInsert( lbThreadProgress, 0, "Thread 1: " || strProgress )`
  - ◇ Set `prbThread = nProgress`

- Hintergrundverarbeitung läuft
- Zeige Fortschritt
- Usw.

# Event: Thread Finished



- ◆ Thread Finished
  - ◆ Actions
    - ◆ Call SalPicSet( picThread , red, PIC\_FormatPNG )
    - ◆ Call SalListInsert( lbThreadProgress, 0, "Thread 1 has finished." )
    - ◆ Set dfEndTime = SalDateCurrent( )
    - ◆ Set dfDuration = (dfEndTime - dfStartTime) \* 24 \* 60 \* 60
    - ◆ Call SalDisableWindow( pbThreadCancel )
    - ◆ Call SalEnableWindow( pbExport )

- Thread wurde beendet
- Daten darstellen
- Usw.

# Starten eines Thread



```
◆ Pushbutton: pbExport
◆ Message Actions
◆ On SAM_Click
  ◆ Set dfStartTime = SalDateCurrent( )
  ◆ Call SalBackgroundWorkerStart( oThread )
  ◆ Set dtStartTime1 = SalDateCurrent( )
  ◆ Call SalBackgroundWorkerStart( oThread1 )
  ◆ Call SalEnableWindow( pbThreadCancel )
  ◆ Call SalEnableWindow( pbThreadCancel1 )
  ◆ Call SalDisableWindow( pbExport )
  ◆ Set prbThread = 0
  ◆ Set prbThread1 = 0
```

- Der Background Worker wird als Instanz gestartet und löst die Eventverarbeitung aus

# Threading API



- SalBackgroundWorkerStart(oBkgdWorker)
- SalBackgroundWorkerReportProgress(objBkgdWorker, nProgress, sMessage)
- SalBackgroundWorkerIsBusy(objBkgdWorker)
- SalBackgroundWorkerCancel(objBkgdWorker)
- SalBackgroundWorkerIsCanceled()

# Team Features



- Integration zu Source Management & Versioning Systemen
  - GIT / SVN / Team Foundation Server



# OPENTEXT™

[www.opentext.com](http://www.opentext.com)



[twitter.com/opentext](https://twitter.com/opentext)



[facebook.com/opentext](https://facebook.com/opentext)



[linkedin.com/company/opentext](https://linkedin.com/company/opentext)