GUPTA

Roadshow 2015

Team Developer 6.3 .NET

Helmut Reimann

Team Developer 6.3 .Net

- .Net Assemblies einbinden
- Exception Handling
- Web Services

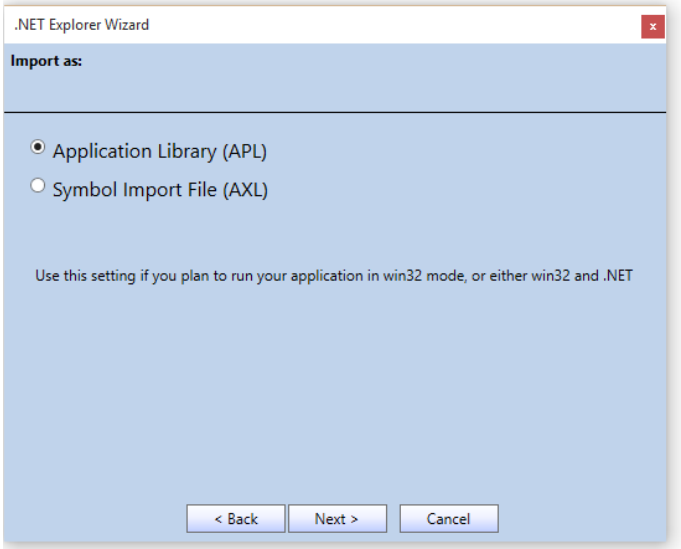
.Net Assemblies

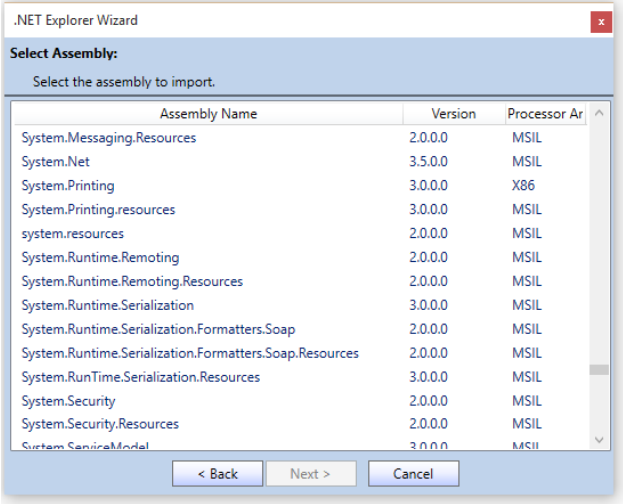
- .Net Assemblies können in Win32 und in .Net Anwendungen eingebunden werden
  - .Net Explorer:

Achtung: In der Windows Version des Team Developers ist der .Net Explorer ein separates Tool!

.Net Assemblies

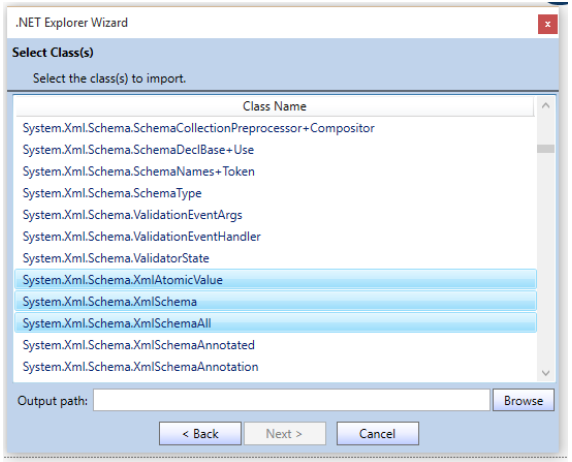
- **Optionen:**

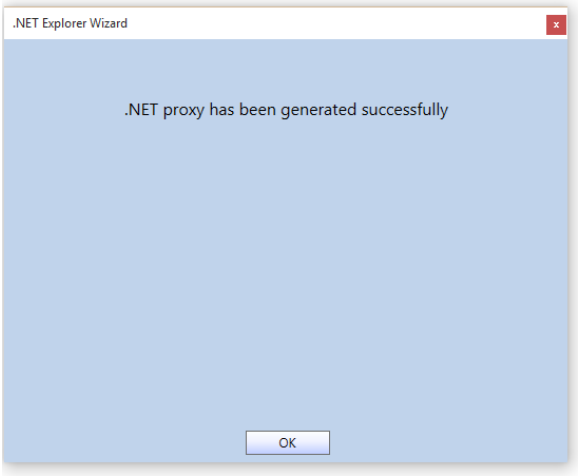


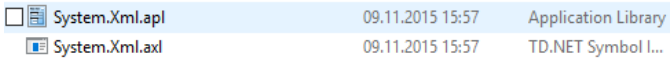


.Net Assemblies

- **Klassen auswählen**







.Net Explorer

- Application Description: Gupta SQLWindows Standard Application Template
  - Libraries
    - File Include: System.Xml.apl
    - File Include: GAILBase.apl
  - Global Declarations
    - Window Defaults
    - Formats
    - External Functions
    - External Assemblies
      - Symbol Import: System.Xml.axl
    - Constants
    - Resources
    - Variables
    - Internal Functions
    - Named Exceptions
    - Named Toolbars
    - Named Menus
    - Class Definitions

Je nach Compiler Setting wird die APL oder die AXL eingebunden

- Class Definitions
  - Functional Class: System_Xml_Schema_XmlSchemaAll
  - Functional Class: System_Xml_Schema_XmlAtomicValue
  - Functional Class: System_Xml_Schema_XmlSchema
  - Functional Class: System_Xml_Schema_XmlSchemaObjectCollection
  - Functional Class: System_Xml_Schema_XmlSchemaGroupBase
  - Functional Class: System_Xml_Schema_XmlSchemaType
  - Functional Class: System_Xml_IXmlNamespaceResolver

Team Developer Web Services

- Entwickeln von .NET Web Services
  - .NET Compiler Einstellungen
  - Neue WS-Class: *Operations*
  - Soap Fault / Exception handling
  - Web Service Sicherheit
  - Veröffentlichen von Web Services
  - Testen von Web Services
  - Debuggen von Web Services

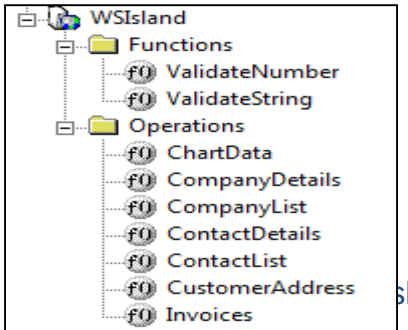
Web Services

- **Verlagern von Business Logik auf einen zentralen MS-IIS Server**
  - **Stateless Datenbankzugriffe**

```
Actions
  Set sSQL = 'select T_ID, Typ, Termin, Bezeichnung, Verein from Ter
    :oTermin.nTerminID, :oTermin.nTerminTyp, :oTermin.dDatum, :oTer
    where T_ID = :npTerminID '
  Set ngSQL_ID = 1421
  If GConnect()
    If SqlPrepareAndExecute(ghSQL, sSQL)
      Call SqlFetchNext(ghSQL, nRet)
    Call SqlDisconnect(ghSQL)
  Return oTermin
```

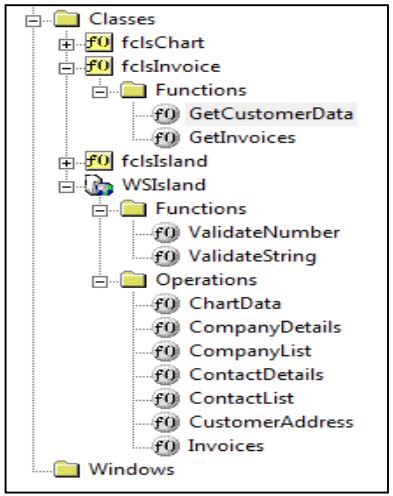
Neuer Klassen Typ: Web Service

- Web Service Klasse hat
  - Lokale Funktionen
    - Nicht sichtbar für den Anwendungsentwickler
  - Operations
    - Operations sind die Schnittstellen zur Client Anwendung
    - Simple Web Service Struktur:



Neuer Klassen Typ: Web Service

- **Web Service Klassen können bestehende *Functional Classes* nutzen:**



Einfache Operation (Receive Parameter)

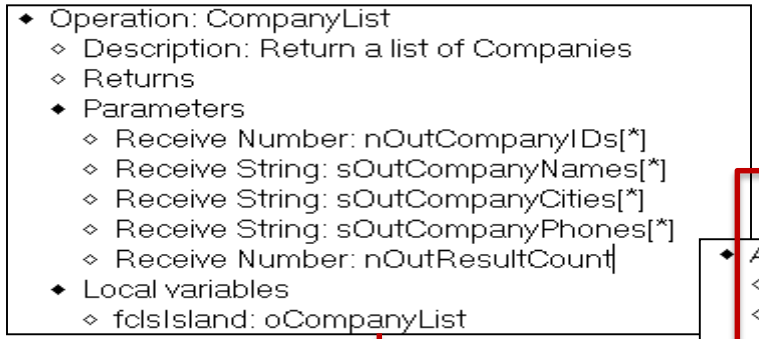
- **Programm Code einer einfachen Operation:**

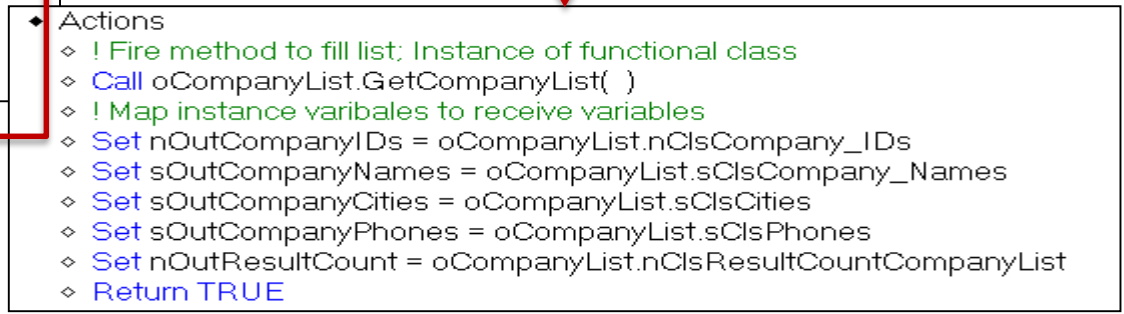
```
Operation: CompanyDetails
  Description: Return company details
  Returns
  Parameters
    Number: nInCompanyID
    Receive String: sOutCompany_Name
    Receive String: sOutAddress
    Receive String: sOutCity
    Receive String: sOutState
    Receive String: sOutZIP
    Receive String: sOutCountry
    Receive String: sOutPhone
    Receive String: sOutFax
    Receive String: sOutTerms
    Receive Number: nOutLine
    Receive String: sOutCorporate_URL
  Local variables
    fclsIsland: oCompanyDetails
    Boolean: bOk
```

```
Actions
  ! Check for valid input
  Set bOk = ValidateNumber( nInCompanyID )
  If bOk = TRUE
    ! Set instance variable for select
    Set oCompanyDetails.nClsCompany_ID = nInCompanyID
    ! Fire method
    Call oCompanyDetails.GetCompanyDetails( )
    ! Copy instance variables to receive values
    Set sOutCompany_Name = oCompanyDetails.sClsCompany_Name
    Set sOutAddress = oCompanyDetails.sClsAddress
    Set sOutCity = oCompanyDetails.sClsCity
    Set sOutState = oCompanyDetails.sClsState
    Set sOutZIP = oCompanyDetails.sClsZip
    Set sOutCountry = oCompanyDetails.sClsCountry
    Set sOutPhone = oCompanyDetails.sClsPhone
    Set sOutFax = oCompanyDetails.sClsFax
    Set sOutTerms = oCompanyDetails.sClsTerms
    Set nOutLine = oCompanyDetails.nClsLine
    Set sOutCorporate_URL = oCompanyDetails.sClsCorparate_Url
    Return TRUE
  Else
    Return FALSE
```

Einfache Operation (Arrays von Receive Parameter)

- **Programm Code mit Arrays als Receive Parameter:**





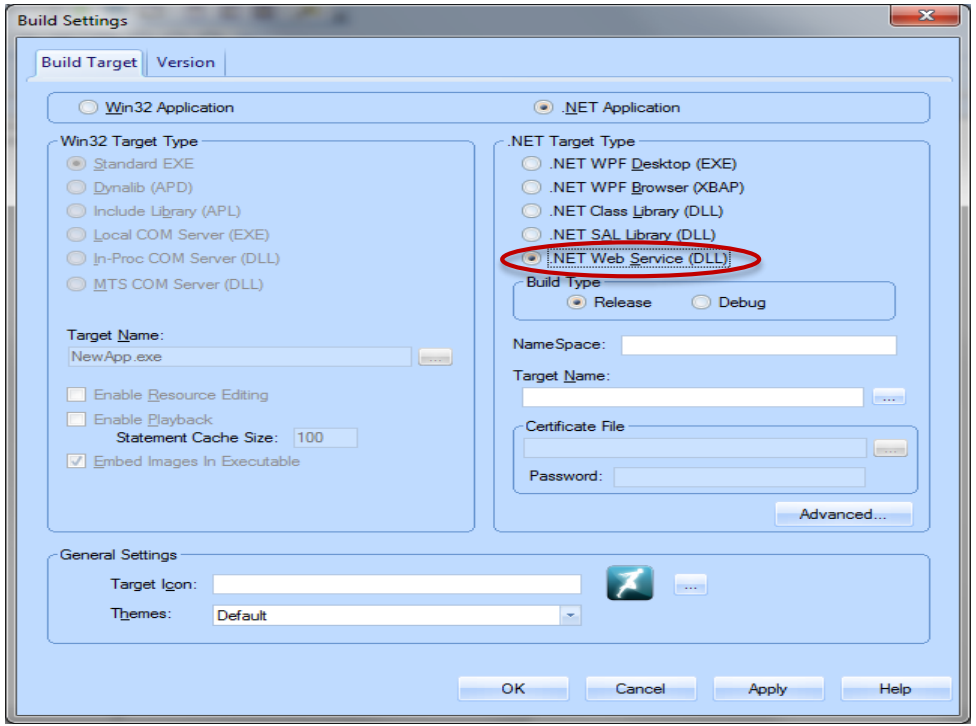
Einfache Operation (Return UDV)

- **Code einer Operation mit UDV als Return Wert:**

```
Operation: CustomerAddress
  Description: Return company details as object
  Returns
    fclsInvoice: oReturn
  Parameters
    Number: nInCompanyID
  Local variables
    fclsInvoice: oLocalCompany
    Boolean: bOk
```

```
Actions
  ! Check for valid input
  Set bOk = ValidateNumber( nInCompanyID )
  If bOk = TRUE
    ! Fire method to fill instance variables
    Call oLocalCompany.GetCustomerData( nInCompanyID )
    ! return whole object
    Return oLocalCompany
  Else
    ! Pass Error as Exception to client
    Call SalThrowSoapFault( 'No valid Company_ID' )
```

.NET Compiler Einstellung



.Net Compiler Einstellung für Web Services

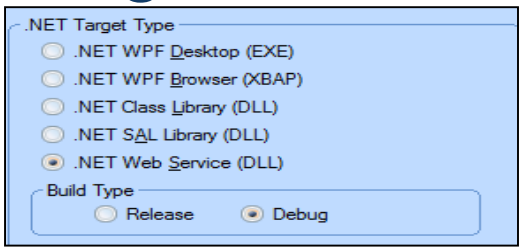
Build Target for .NET Web Service DLL

Achtung: Es dürfen KEINE sichtbaren Elemente in einem Web Service „verbaut“ werden (z. B. Message Boxes usw.)
Nutzt ***SalThrowSoapFault( ),*** um Exceptions oder SQL Fehler zum Client zu senden.

.NET Compiler Einstellungen

- **Compiler generierte Files:**

  - **WSIsland.asmx = Web Service Beschreibung für den IIS**
  - **WS_Island.apt = Team Developer Sourcecode**
  - **WS_Island.dll = compiliertes .NET Web Service Assembly**
- **Compiler Einstellung:**



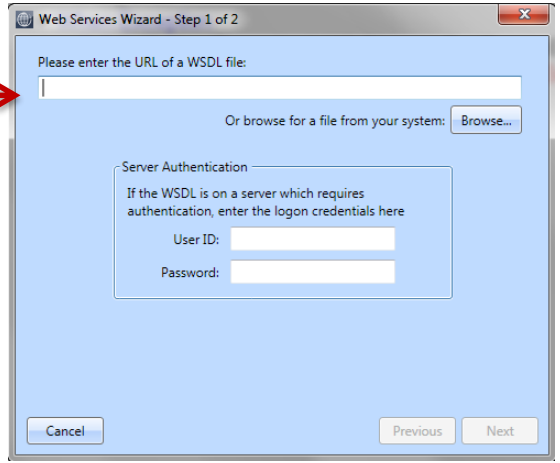
Option: ‚Debug‘:
Generiert zusätzlich
WS_Island.pdb

Datentypen Serilisierung

- ***Binary*** **Datentypen werden zu** ***base64 string***
- ***Bool*** **Datentypen werden zu** ***Boolean***
- ***Date/Time*** **Datentypen werden zu** ***complex data type: SalDate***
- ***Number*** **Datentypen werden zu complex** ***data type*****:** ***SalDecimal***
- ***String*** **Datentypen werden zu** ***Strings***

Web Services Sicherheit

- Web Service Security
  - IIS Security
    - Zugang zur WSDL
    - Zugang zum WS zur Laufzeit




```
◆ Message Actions
  ◆ On SAM_CreateComplete
    ◇ Call oTest.Test61DataTypes()
    ◆ If not SalWSSetCredentials(oTest,'JMG6510','devcon','devcon')
      ◇ Call SalMessageBox("invalid credentials : domain\username\password","Error",0)
```

WS – API Exception Handling

- **WS-API: SalThrowSoapFault(sError)**

bOk = SalThrowSoapFault (sString)
Sendet ein SoapFault an den Client

- **Parameter**
  - **sString: Der Fehlertext**
- **Return Value**
  - **bOk ist TRUE wenn die Funktion erfolgreich war**

```
When SqlError
  Call SqlGetError( hSQL, nError, sError )
  Call WriteTrace( sError )
  Call SalThrowSoapFault( sError )
```

1a. SQL Fehler ermitteln und Throw...

```
Else
  ! Pass Error as Exception to client
  Call SalThrowSoapFault( 'No valid Company_ID' )
```

1b.Throwing einer ‚User' Exception zum Client

```
When Exception
  ! What should be done in case of an exceptions
  Call SalGetLastException(sgWinName,ngResError,sgErrorMessage,sgArrayStack)
  Call SalThrowSoapFault( sgErrorMessage )
```

1c. Throwing a .NET exception to client

Exception handling .Net Client

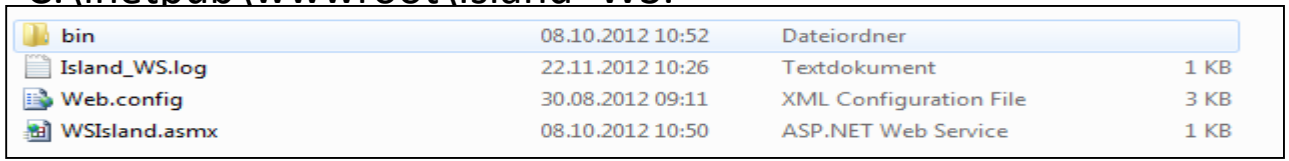
- **Lokales Exception handling**

```
◆ Actions
  ◇ Set sContactID = SalNumberToStrX( nParContactID, 0 )
  ◆ When Exception
    ◇ ! What should be done in case of an exceptions
    ◇ Call SalGetLastException(sgWinName,ngResError,sgErrorMessage,sgArrayStack)
    ◇ Call SalModalDialog( dlgNetExceptions, hWndMDI )
  ◇ Call oContact.ContactDetails( sContactID, dfFirstName, dfLastName, dfTitle,
                                  dfPhoneCon, dfFaxCon, dfEmail )
  ◇ End Exception
  ◇ Return TRUE
```

Deploying Web Services im IIS

- **File Strukture im IIS (für das Island WS Beispiel)**

C:\inetpub\wwwroot\Island_WS:



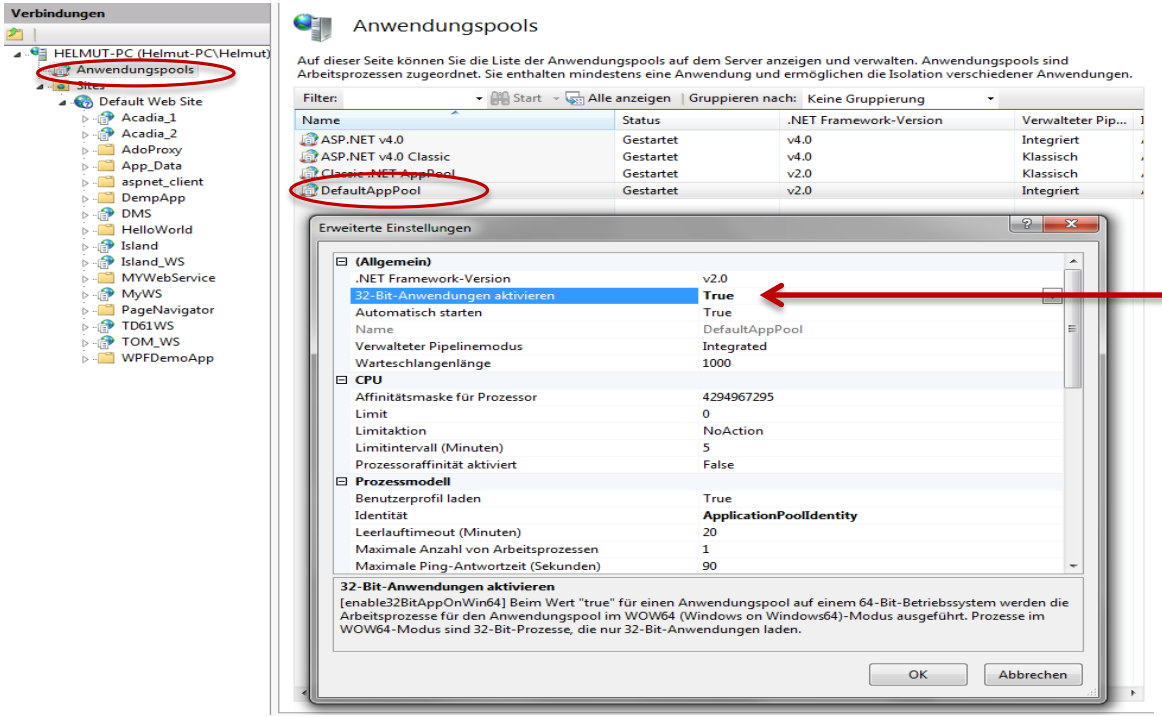
bin	08.10.2012 10:52	Dateiordner	
Island_WS.log	22.11.2012 10:26	Textdokument	1 KB
Web.config	30.08.2012 09:11	XML Configuration File	3 KB
WSIsland.asmx	08.10.2012 10:50	ASP.NET Web Service	1 KB

C:\inetpub\wwwroot\Island_WS\bin:

WS_Island.dll	09.10.2012 16:24	Anwendungserweiterung	36 KB

Hinweis: Island_WS.log = trace file für WS Fehler; generiert mit *SalWriteTrace*( )
WSIsland.asmx = TD Compiler generierte ASP.NET Web Service Beschreibung
Web.config = Konfigurationsfile mit Pfaden zu SQL.ini & TD .NET runtime
WS_Island.dll = TD-Compiler generierte Web Service DLL

Managing IIS imInetMgr.exe

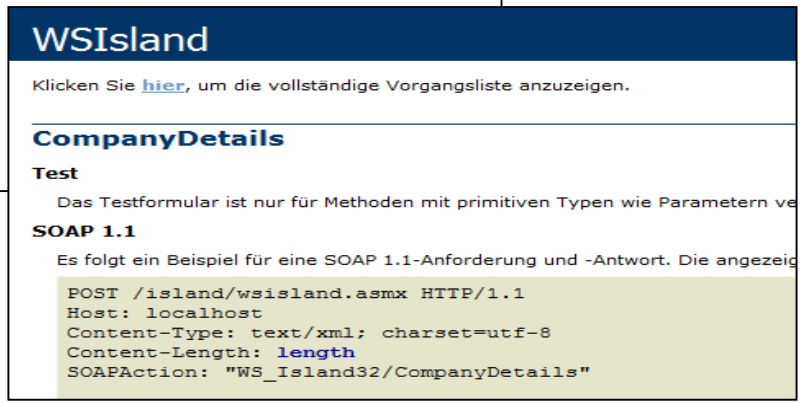


Diese Einstellung wird benötigt, um TD 6.3 Web Services im IIS verwenden zu können!

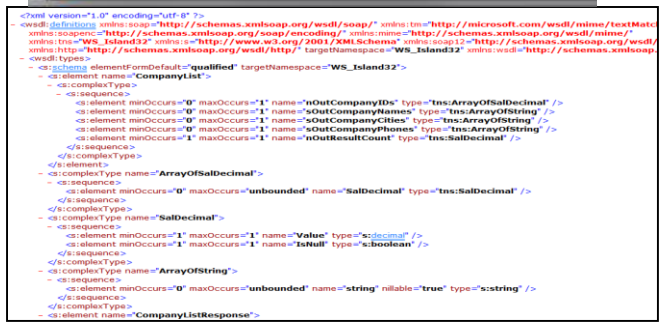
Test Web Service im Browser

- **Öffnen der URL in browser:**

Browser shows all Operations of WS



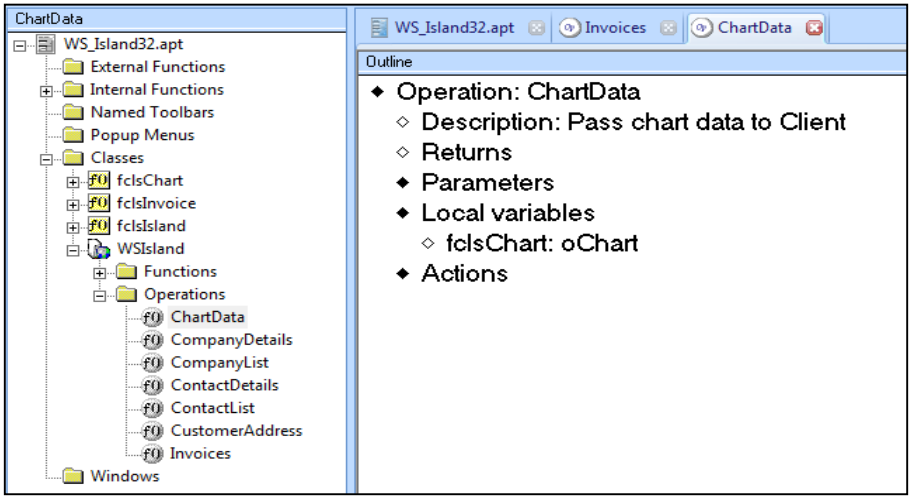
Select Operation to run



Besser: Testanwendung schreiben (Test von Exeption Handling, komplexen Datentypen usw.)

Debuggen von Web Services

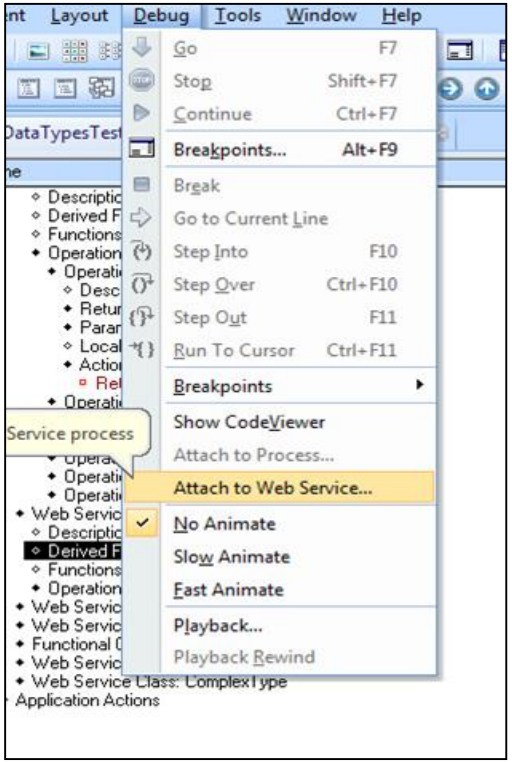
- **Beispiel: Debuggen eines .NET Web Service:**



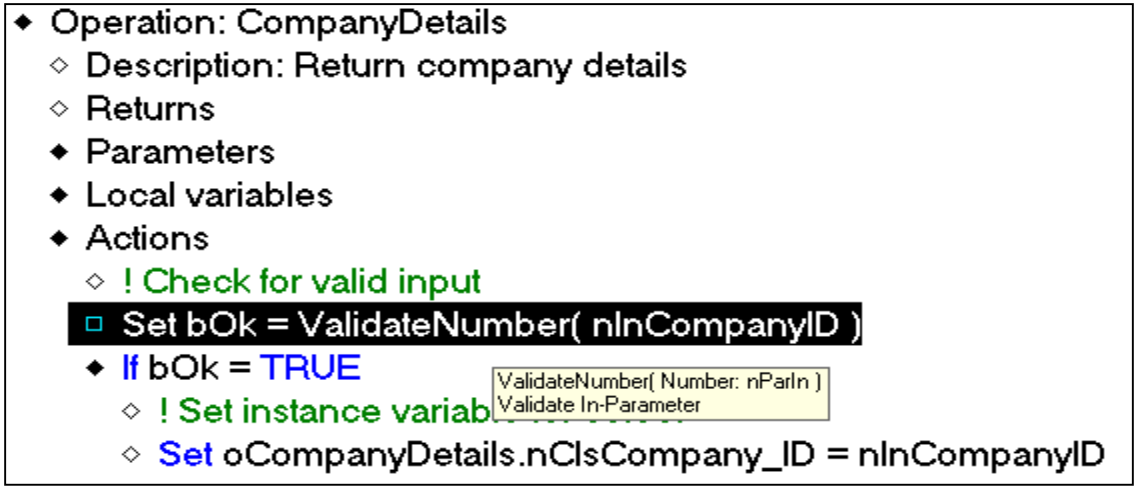
Starte TD6.3 als Administrator und öffne den Sourcecode in der Publish Directory innerhalb des IIS

Debuggen von Web Services

Attach den Sourcecode des Web Services zu dem laufenden WS im IIS

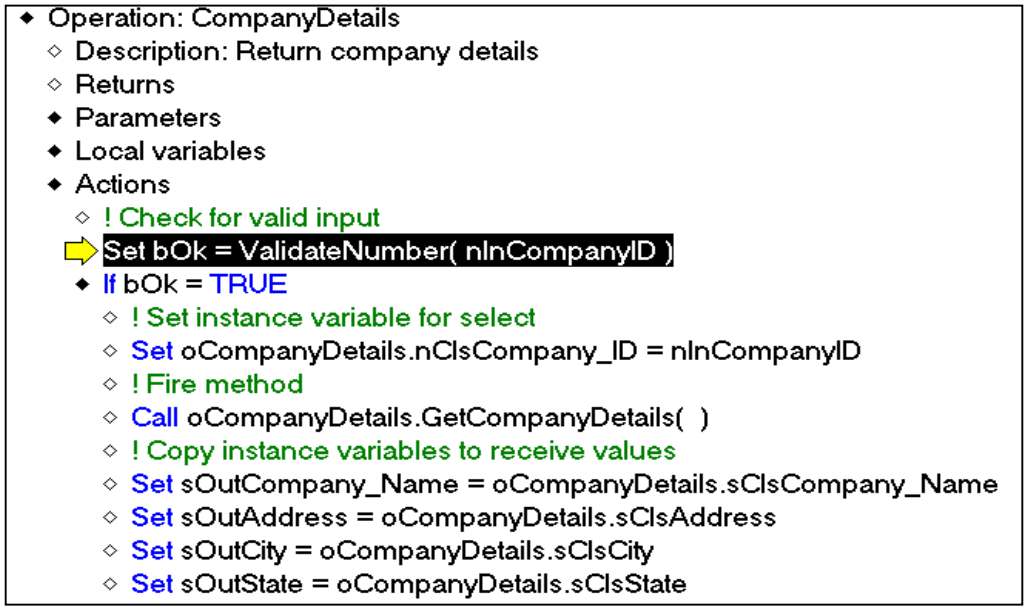


Debuggen von Web Services



Setze Breakpoint innerhalb der Source des WS

Debuggen von Web Services



Gehe schrittweise durch den Code des WS.....

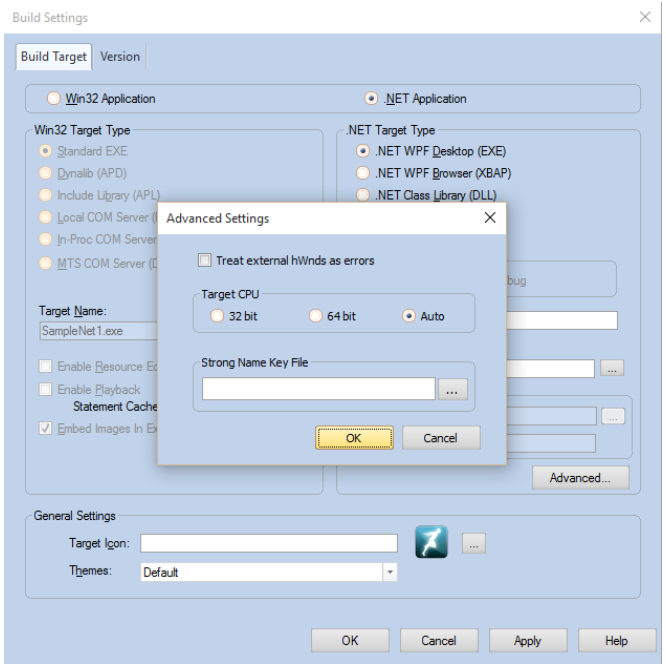
Exception handling Win32 Client

```
◆ Application Actions
  ◆ On SAM_NetException
    ◇ Set nButton =  SalMessageBox( '.NET Exception happends', 'Global Exception Handler', MB_RetryCancel | MB_IconStop )
    ◇ !
    ◇ ! Enter code to handel exception
    ◇ !
    ◇ ! Return TRUE => end error processing
    ◆ If nButton = IDRETRY
      ◇ Call SalMessageBox( 'Return TRUE: End error processing,
          do next steps...', 'Global Exception Handler', MB_Ok | MB_IconInformation )
      ◇ Return TRUE
    ◇ ! Return FALSE => continue error processing
    ◆ If nButton = IDCANCEL
      ◇ Call SalMessageBox( 'Return FALSE: Go ahead with error processing,
          do next steps...', 'Global Exception Handler', MB_Ok | MB_IconInformation )
      ◇ Return FALSE
```

Programmcode um eine .NET Exception in *SAM_NetException* abzufangen

.Net 64 Bit

- **Compiler Setting**



Achtung: Bitness muss übereinstimmen!

Team Developer 6.3 .Net Anwendungen

- **Ja, das ist realisierbar!**
  - **Es sind allerdings einige Vorarbeiten zu erledigen!**
  - **Auskunft gibt der Upgrade-Report des TD6.x**

Sal Web Reporting Funktionen

- Zwei Sal Funktionen stellen eine Web-Reporting Funktionalität innerhalb einer TD Mobile oder Team Developer .Net Anwendung zur Verfügung:
  - SalWebReportCreateToFile( )
  - SalWebReportCreate( )
- Anwendungsbeispiele
  - Generierung einer Rechnung und Versenden als Anhang einer Mail
  - Generieren eines Dokumentenarchivs für eine spätere Nutzung

SalWebReportCreateToFile( )

- Generiert einen Reportfile basierend auf Daten und einem Report Template
  - Ausgabe kann entweder PDF oder HTML sein, abhängig vom letzten Argument der entsprechenden Sal Funktion
- Das generierte Dokument verbleibt auf dem IIS Server
- Gibt dem Programmierer die Möglichkeit, Report zu erzeugen um sie dann SalFile* Funktionen zu verarbeiten

SalWebReportCreateToFile( ) Syntax

- bOk = SalWebReportCreateToFile(sRPXFile, sOutputFile, reportVars, reportData, nOutputType, sLang)

- Arguments:
  - sRPXFile: (String), the name of the RPX file which defines the report layout. This file is expected to be found on the web server and the path, if any, is relative to the home location of the web service which is running
  - sOutputFile: (String), the name of the output file which will be generated. This file will be generated on the web server and its path, if any, is relative to the home location of the web service which is running
  - reportVars: (UDV) an instance of a class whose member fields contain any “report variables” to be used by the report engine. The names of the class fields must match the names of the report variables as defined in the RPX file.
  - reportData: (UDV array) an array of class objects whose member fields contain the report “data items”. The names of the class fields must match the names of the report variables as defined in the RPX file.
  - nOutputType: (Number) a constant specifying the kind of report file to generate:
    - WRPT_OutputPDF: A PDF file
    - WRPT_OutputHTML: An HTML file
  - sLang: (String) The language to use when generating the report (e.g. “en-US”)

- Example:
  - Call SalWebReportCreateToFile("Invoice.rpx", “report.pdf", reportVars, inputItems, WRPT_OutputPDF, “en-US”)

SalWebReportCreate( )

- **Einfache Funktion zum Erzeugen eines serverseitigen Reports und direkter Übertragung als Binary Datenstream**
- **Sinnvoll, wenn der Anwender keine Schreibberechtigung im APP-Pool auf dem IIS hat**

SalWebReportCreate( ) Syntax

- bOk = SalWebReportCreate(sRPXFile, binOutputFile, reportVars, reportData, nOutputType, sLang)

- Arguments:
  - sRPXFile: (String), the name of the RPX file which defines the report layout. This file is expected to be found on the web server and the path, if any, is relative to the home location of the web service which is running
  - binOutputFile: (Receive Binary), the generated file, either PDF or HTML, in binary format
  - reportVars: (UDV) an instance of a class whose member fields contain any “report variables” to be used by the report engine. The names of the class fields must match the names of the report variables as defined in the RPX file.
  - reportData: (UDV array) an array of class objects whose member fields contain the report “data items”. The names of the class fields must match the names of the report variables as defined in the RPX file.
  - nOutputType: (Number) a constant specifying the kind of report file to generate:
    - WRPT_OutputPDF: A PDF file
    - WRPT_OutputHTML: An HTML file
  - sLang: (String) The language to use when generating the report (e.g. “en-US”)

- Example:
  - Call SalWebReportCreate("Invoice.rpx", bMyReport, reportVars, inputItems, WRPT_OutputPDF, “en-US”)

Fragen?