Exercises

Unit: Data Interface API

Topic: Establish a Connection to SAP Business One

At the conclusion of this exercise, you will be able to:

- Connect to an SAP Business One database

You want to develop additional functionality for SAP Business One.

In a first step, you want to create a simple program to connect to an existing SAP Business One database.

1-1 Log on to SAP Business One.

1-1-1 Note the name of one database you want to log on to.

1-1-2 Note one user in that database and the user's password.

1-2 Create a new Visual Studio project.

1-2-1 Within this project, create a form with two buttons on it. One of the buttons should be used to connect to the SAP Business One database, the other to disconnect from it.

1-2-2 Add a reference to the SAP Business One DI API COM library…

1-3 Code the connection to the SAP Business One database.

1-3-1 Define a variable for the Company object – ensure it is defined as a member of the add-on application class or globally.

1-3-2 Create a new Company object.

1-3-3 Set the properties needed to connect to the SAP Business One database.

1-3-4 Call connect on the Company object

1-4 Implement error handling and success handling.

  1-4-1 If the connection succeeds, display a message box displaying a corresponding message.

  1-4-2 If the connection failed, display the error message provided by the Company object.

1-5 Code the disconnection from the SAP Business One database.

Exercises

Unit: Data Interface API

Topic: Documents Object

At the conclusion of this exercise, you will be able to:

- Work with Documents objects

Create a Sales Order in Business One. Via the DI create an Invoice based on this Sales Order and later create an Incoming Payment for that Invoice

2-1 On your Visual Studio project create a new button called "Invoice and Payment"

2-2 In Business One create an Order for a particular customer and a particular item.

2-2-1 First you must create a new Document object instance for the Invoice. Then you set the properties of the Documents object and the Documents_Lines ensuring the BaseEntry, BaseLine and BaseType are set.

2-2-2 Add the whole document. In the case of success, you should bring up a message box telling the user the number of the newly added Sales Invoice using the method GetNewObjectCode. In case of any error, you should display a message box with an error message.

The Documents object must be created with the GetBusinessObject method of the company object you are connected to. Look in the online help of the GetBusinessObject method for the correct object type. Which one must be used?

To access the Documents_Lines object, look at the properties of the Documents object.

To create a document based on a document you need to utilize the properties BaseEntry (DocEntry of Base document), BaseType (in this case Sales Order), BaseLine (line you wish to copy to target document)

2-2-3 Finally you should release the document object variables.

2-3 Create the Incoming Payment for this Invoice

2-3-1 Create a new Payments object instance for the Incoming Payment. Then you set the properties for the CardCode, Invoice DocEntry, and we will pay via cash so we will use the properties CashAccount and CashSum.

2-3-2 Add the whole document. In the case of success, you should bring up a message box telling the user the number of the newly added Payment using the method GetNewObjectCode. In case of any error, you should display a message box with an error message.

2-3-3 Finally you should release the document object variables.

Exercises

Unit: Data Interface API

Topic: XML

At the conclusion of this exercise, you will be able to:

- Work with XML

Create data as XML and checkout how to use this process to add new data to the SAP Business One database.

3-1 On your Visual Studio project create a new button called “Working with XML”

3-2 Save the Invoice created in the Documents exercise as XML.

3-2-1 Try all settings for XmlExportType property on the Company object and find the differences.

Have a look at the DI-API Help file

3-2-2 Save the Invoice document created in the previous exercise in Xml format

Use the GetAsXml or SaveXml methods of the Documents object (verify all business objects have the same methods)

3-2-3 Test also the method GetBusinessObjectXmlSchema of the Company object. What kind of information does it save?

3-3 Modify the XML data obtained before and add it to the SAP Business One database.

Use the method GetBusinessObjectFromXML of the Company object

3-3-1 Try all files generated above and check the errors (exceptions) for details.

Exercises

Unit: Data Interface API

Topic: Transactions

At the conclusion of this exercise, you will be able to:

- Work with transactions

Create an Order via DI API and later create an Invoice that is based in that Order, Documents exercise done before.

This time open a transaction before and close it afterwards.

4-1 Log on to a SAP Business One Company as shown in the first exercise.

4-2 Open a transaction (StartTransaction of the Company Object).

4-3 Perform the same actions as you did in the Documents exercise.

4-4 Close the transaction (EndTransaction of the Company Object).

4-5 Play e.g with the TaxCode (or VatGroup – depending on the localization!) property to see if and how the transaction fails. Also use wrong data (e.g. non-existing CardCode etc.) to see the reaction (as discussed in the presentation).

Exercises

Unit: Data Interface API

Topic: Using General Objects

At the conclusion of this exercise, you will be able to:

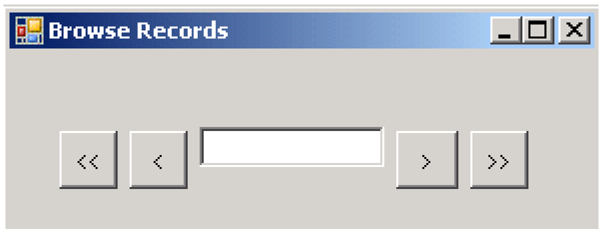
- Use the data browser object to browse through a set of data
- Use the record set object

Create an application to navigate through all customers.

You will use the Browser property of the BusinessPartners object. Add the navigation buttons to your form and provide the coding so that the user can browse through the customers.

5-1 On your Visual Studio project create a new button called “General Objects”

5-2 Create a new form in your Visual Studio application containing a text box where you will show the Business Partners Card Code and four buttons: first, previous, next and last.



5-3 Create a Recordset object and set the Browser property of the BusinessPartners object to this Recordset.

There is a code sample in the DI-API Help documentation.

5-3-1 Add the code to all four of the buttons so that the user can navigate backwards and forward through the customers. Be sure that your application only includes customers, not Leads or Vendors (Suppliers).

Use the DoQuery method of the RecordSet object with the appropriate SQL query.

5-4 Test your changes. Be sure to include the following scenarios:

5-4-1 Click the "First Record" button (|<<), then click it again. Try the same thing with the "Last Record" button (>>|).

5-4-2 Click the "First Record" button (|<<), then click the "Previous Record" button (<).

5-4-3 Click the "Last Record" button (>>|), then click the "Next Record" button (>).

5-4-4 If any of these scenarios raises an error, add code that will fix the error. Then test the application again.

Exercises

Unit: Data Interface API

Topic: Meta Data

At the conclusion of this exercise, you will be able to:

- Work with Meta Data objects in the DI API

Create user-fields and user-tables in the SAP Business One database.

- Use the UserTableMD Object to create User Tables
- Use the UserFieldMD Object to create User Fields
- Use the specifications for the User-Defined Table and the User-Defined Fields within from the “Course Project Exercise” (see end of the course’s “Introduction” section

6-1 As a first small exercise add a User-Defined Field to the item table (OITM) through DI API. On your Visual Studio project create a new button called “UDF and UDT”

6-1-1 Use namespace “TB1_” as a prefix…

6-2 Add a User-Defined Table (use namespace “TB1_” as a prefix…), but do not add any fields to the table yet.

Table name: TB1_VIDS

Table description: Video Management

You will need to create an instance of the UserTablesMD object in order to add a field to the User Table. It is recommended that after you create your table you set this object variable to “Nothing” so that its properties do not inadvertently carry forward to the next table or field you are creating.

6-3 Test your application by opening the “Manage User Fields” window in SAP Business One. Check to see that the table was added.

6-4 Remove the User-Defined Table (in the SAP Business One application) you just created before. Enhance your application with the capability to remove the User-Defined Table through DI API – and then test your application to see that you can also add and delete the User-Defined Table in SAP Business One.

6-5 Add the following User-Defined Fields to your new User-Defined Table:

You will need to create an instance of the UserFieldsMD object in order to add a field to the User Table. It is recommended that after you create each field, you set this object variable to "Nothing" so that its properties do not inadvertently carry forward to the next field you are creating. Do the same thing at the end of the last user field added.

Aisle Number – Indicates in which aisle the movie is stored.

- Field Name: AISLE
- Field Description: Aisle Number
- Field Type: db_Numeric
- Field EditSize: 2

Section – Indicates the section the movie is store in the aisle.

- Field Name: SECTION
- Field Description: Section Number
- Field Type: db_Alpha
- Field EditSize: 20

Rented – Indicates weather the movie is rented or not.
Holds 2 "valid values": Y/N.

- Field Name: RENTED
- Field Description: Rented/Available
- Field Type: db_Alpha
- Field EditSize: 1

CardCode – In case the movie is "Rented"
This field will hold the CardCode of the customer who rented it
. otherwise it will be empty.

- Field Name: CARDCODE
- Field Description: Card Code
- Field Type: db_Alpha
- Field EditSize: 20

6-6 Test your application and make sure all your fields were added successfully.

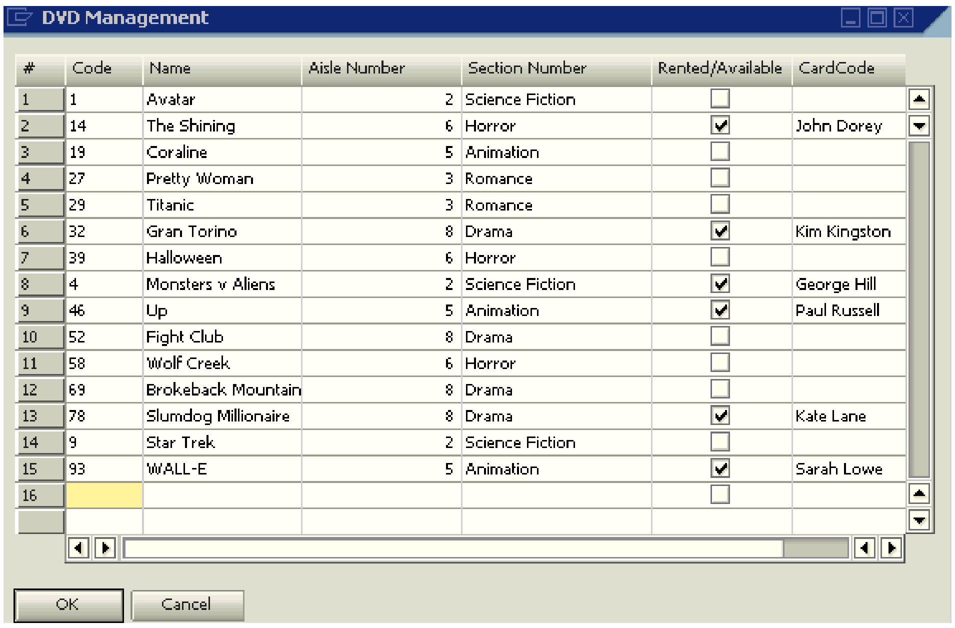
6-7 Write data into the User-Defined Table.

6-7-1 Add about 15 records to your new User-Defined Table.

- In order to add a record, you will need to use the UserTable object. The name of this object is a bit misleading – the UserTable object actually corresponds to a ***record*** within a user table.
- When referring to specific fields within a User Table record, you must prefix the fieldname with "U_". For example, if you have created a User-Defined Table object variable called pRecord, you could set the value of the "Make" field by adding this line of code: **pRecord.UserFields("U_Make").Value = "Ford"**

The "Code" and "Name" must each be unique within the User Table. The "Code" is the Primary Key used to retrieve a record.

6-7-2 Your User-Defined Table could look like this:


DVD Management

#	Code	Name	Aisle Number	Section Number	Rented/Available	CardCode
1	1	Avatar	2	Science Fiction	☐	
2	14	The Shining	6	Horror	☑	John Dorey
3	19	Coraline	5	Animation	☐	
4	27	Pretty Woman	3	Romance	☐	
5	29	Titanic	3	Romance	☐	
6	32	Gran Torino	8	Drama	☑	Kim Kingston
7	39	Halloween	6	Horror	☐	
8	4	Monsters v Aliens	2	Science Fiction	☑	George Hill
9	46	Up	5	Animation	☑	Paul Russell
10	52	Fight Club	8	Drama	☐	
11	58	Wolf Creek	6	Horror	☐	
12	69	Brokeback Mountain	8	Drama	☐	
13	78	Slumdog Millionaire	8	Drama	☑	Kate Lane
14	9	Star Trek	2	Science Fiction	☐	
15	93	WALL-E	5	Animation	☑	Sarah Lowe
16					☐	

Optional Exercise

Unit: Data Interface API

Topic: Services

At the conclusion of this exercise, you will be able to:

- Work with Service Type objects

Use CompanyService to change the backgound color of forms for a particular company…

7-1 On your Visual Studio project create a new button called “Service Object”

7-2 Get CompanyServices object.

7-3 Get structure which reflects information in table OADM.

7-4 Set the background color to purple.

7-5 Call the method which updates the information in the SAP Business One database. See the effect in the SAP Business One application.

Please note that only forms opened after changing the bachground color will reflect this change.