MB-500: Microsoft Dynamics 365: Finance and Operations Apps Developer
Lab 5 – Code Extension & Development

Contents

Lab Environment

In order to run this lab, you will need:

- An all-in-one demo data VM with
    - Visual Studio installed, and a Visual Studio subscription
    - A browser to run the user interface
    - Lab 4 – Metadata Extension & Development completed

Lab Overview

- Dependency: Lab 4 – Metadata Extension & Development should be completed
- Develop & Extend Class, Methods etc.

Estimated time to complete this lab: 85+ minutes

Scenario 1 (Exercises 1-2)

- You need to add validation in the Customer Fly Details table to ensure From and To Airports are not the same
- You need to add a find() method on the DDTTierRange table
- You need to add a new method on the DDTTierRange table named getTier(), which will return the corresponding Tier once you send the Miles as parameter into the method
- You need to add a new method on CustTable table named getTotalMiles(), which will return the total miles of a customer
- You need to add a new method called getAirportCode() on the DDTAirport table, which will return the airport code once you send the table recId as parameter
- You need to add a validation in the DDTTierRange form datasource that will check that ToMiles is greater than FromMiles
- You need to write code in the DDTCustFlyDetails form datasource that will auto increase the value of the Counter column for each Customer
- You need to create a new runnable class that will update the Customer tier status reading data from Customer Flying Details

Scenario 2 (Exercises 3-4)

- You need to create a new package (MyLabAirlines) that will have the DynamicsDevTraining package as a reference
- You need to create a new table MLAAirportMilesChart that will capture Miles between two Airports
- You need to create a new form MLAAirportMilesChart using the table created above
- You need to create a menu item and place it under Accounts Receivables > Setup to call MLAAirportMilesChart form
- You need to add a new field to capture Loyalty Points in the extension of DDTCustFlyDetails table
- You need to add the new Loyalty Points field in the extension of DDTCustFlyDetails form
- You need to add a new Loyalty Percent field in the extension of DDTTierRange table. This field will calculate the Loyalty points of the customer based on the percentage of the Miles
- You need to add the new Loyalty Points field in the extension of DDTTierRange form
- You need to add a validation routine in the DDTTierRange form datasource to avoid entering an overlapping range
- You need to develop code to update the Flying miles and Loyalty points in the DDTCustFlyDetails table based on data entered in the Airport Miles Chart

Exercise 1: Develop Table Methods

Task 1: Table DDTCustFlyDetails > validateField Method

1. Open DynamicsDevProject in Solution Explorer
2. Right click the project and select **Add > New Item**
3. Select Class under **Dynamics 365 Items > Code**
4. Create a new class DDTCustFlyDetailsEventHandler
5. Open table DDTCustFlyDetails in the designer pane
6. Under the events node, find onValidatedField event; right-click and select **Copy event handler method**
7. Paste the method signature in the new class DDTCustFlyDetailsEventHandler within the class's brackets

```
/// <summary>
///
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
[DataEventHandler(tableStr(DDTCustFlyDetails), DataEventType::ValidatedField)]
public static void DDTCustFlyDetails_onValidatedField(Common sender, DataEventArgs e)
{
}
```

8. Paste the following piece of code within the method

```
ValidateFieldEventArgs  event = e as ValidateFieldEventArgs ;
DDTCustFlyDetails custFlyDetails = sender as DDTCustFlyDetails;
```

```
    boolean result = event.parmValidateResult();


if(result)
{
   switch(event.parmFieldId())
   {
      case fieldNum(DDTCustFlyDetails, FlyFrom):
      case fieldNum(DDTCustFlyDetails, FlyTo):
         result = result && (custFlyDetails.FlyFrom != custFlyDetails.FlyTo);
         break;
   }
}
event.parmValidateResult(result);
```

Task 2: Table DDTTierRange > find Method

1. Open DynamicsDevProject in Solution Explorer and search for DDTTierRange table
2. Double click the table to open it in Designer
3. Right click the Methods node and add a find() method as follows

```
   static DDTTierRange find(DDTCustomerTier   _custTier,
                            boolean      _forUpdate = false)
     {
        DDTTierRange tierRange;
        if (_custTier)
        {
```

```
            if (_forUpdate)
            {
               tierRange.selectForUpdate(_forUpdate);
            }
            select firstonly tierRange
               index hint CustTierIdx
               where tierRange.CustTier == _custTier;
         }
         return tierRange;
      }
```

Task 3: Table DDTTierRange > new Method

1. Open DynamicsDevProject in Solution Explorer and search for DDTTierRange table
2. Right click the method and add a new method getTier() as follows

```
public static DDTCustomerTier getTier(int _miles)
{
   DDTTierRange    tierRange;
   DDTCustomerTier ret;
   while select tierRange
   {
      if(tierRange.FromMiles <= _miles && tierRange.ToMiles >= _miles)
         {
               ret = tierRange.CustTier;
         }
   }
```

```
    return ret;

}
```

Task 4: Table CustTable > new Method

1. Open DynamicsDevProject in Solution Explorer
2. Right click project and select **Add > New Item**
3. Select Class under **Dynamics 365 Items > Code**
4. Create a new class DDTCustTableTable_Extension
5. This should be the signature of the class to ensure it is a class extension:

```
[ExtensionOf(tableStr(CustTable))]

final class DDTCustTableTable_Extension

{

}
```

6. Add new method getTotalMiles() in the class

```
public static DDTFlyingMiles getTotalMiles(CustAccount _cust)

  {

    DDTCustFlyDetails   custFlyDetails;

    select sum(FlyingMiles)

      from custFlyDetails

      where custFlyDetails.custAccount == _cust;

    return custFlyDetails.FlyingMiles;

  }
```

Task 5: Table DDTAirport > new Method

1. Open DynamicsDevProject in Solution Explorer and search for DDTAirport table
2. Double click to open in Designer
3. Right click the Methods node and add a new method getAirportCode() as follows

```
public static DDTAirportCode getAirportCode(RefRecId _AirportRecID)
  {
     DDTAirport  airport;
     select firstOnly AirportCode from airport
        where airport.RecId == _AirportRecID;
     return airport.AirportCode;
  }
```

Exercise 2: Develop Form Methods

Task 1: Form DDTTierRange > Data Source > validateWrite method

1. Open DynamicsDevProject in Solution Explorer and search for **DDTTierRange** form
2. In the form **DDTTierRange,** open data source DDTTierRange and override validateWrite method with the following piece of code

Replace ret = super(); with

```
ret = super() && (DDTTierRange.ToMiles > DDTTierRange.FromMiles);
```

Task 2: Form DDTCustFlyDetails > Data Source > initValue method

1. Open DynamicsDevProject in Solution Explorer and search for **DDTCustFlyDetails** form

In the form **DDTCustFlyDetails,** open data source DDTCustFlyDetails and override initValue method with the following piece of code

```
DDTCustFlyDetails   custFlyDetailsMax;
super();
select FlyCount
  from custFlyDetailsMax
  order by CustAccount, FlyCount desc
  where custFlyDetailsMax.CustAccount == DDTCustFlyDetails.CustAccount;
DDTCustFlyDetails.FlyCount = custFlyDetailsMax.FlyCount + 1;
```

2. Open table DDTCustFlyDetails from Solution Explorer
3. Change the following properties of field FlyCount
    a. Allow Edit: No
    b. Allow Edit On Create: No

Exercise 3: Develop new Class

Task 1: Create a Runnable class DDTUpdateTier

1. Open DynamicsDevProject in Solution Explorer
2. Right click project and select **Add > New Item**
3. Select Runnable Class under **Dynamics 365 Items > Code**
4. Create a new runnable class DDTUpdateTier
5. A blank main() method will automatically be created

```
public static void main(Args _args)
{
}
```

6. Create a new static method update() as a new method to develop the business logic for updating customer tier as follows

```
public static int update()
{
    CustTable       custTable;
    ttsbegin;
    while select forupdate custTable
    {
        custTable.DDTCustomerTier =
DDTTierRange::getTier(CustTable::getTotalMiles(custTable.AccountNum));
        custTable.update();
    }
```

```
    ttscommit;

    return custTable.rowCount();

}
```

7. Now this method needs to be called from main() using the following code

```
public static void main(Args _args)

{

    int updateCount;

    updateCount = DDTUpdateTier::update();

    info(strFmt("Number of records updated is %1", updateCount));

}
```

Task 2: Execute class DDTUpdateTier from CustTable form

1. Open DynamicsDevProject in Solution Explorer
2. Right click project and select **Add > New Item**
3. Select Action menu Item under **Dynamics 365 Items > User Interface**
4. Create a new action menu item DDTCustTierUpdateAction
    a. Object Type: Class
    b. Object: DDTUpdateTier
    c. Label: *Update Customer Tier*
5. In Solution Explorer, search for form extension CustTable.DynamicsDevTraining and open in the designer
6. Navigate to **Design | Pattern > ActionPaneHeader > aptabGeneral**
7. Add a new button Group DDTCustTierButtonGroup
    a. Caption: Update Customer Tier
8. Drag Action menu item DDTCustTierUpdateAction and drop it under the newly created button group DDTCustTierButtonGroup

Check Output

Scenario - 1

- Build solution
- Open a browser, then **Modules > Accounts Receivable > Customers > All Customers**. Select customer **DE-001** and go to **Flying Details** fast tab. Create a new record with the same value in From & To City. The system should not allow entering such data
- Open the **Miles wise Tier Range** form under **Accounts Receivable > Setup**. Try to enter a 'To Miles' less than 'From Miles', which shouldn't be allowed. For a future modification, we should throw an informative error message.
- The Count field in the Fly Details tab of the Customer form should get auto-incremented as soon as a new record is created
- Click in the "**Update Customer Tier**" button in the General Action Pane of the Customer form. A process will get executed that will update the Customer Tier field.

Exercise 4: New package creation

Task 1: Create a new package extending DynamicsDevTraining

1. From **Dynamics 365 > Model Management > Create Model**, create a new model MyLabAirlines

    a. Model name: MyLabAirlines
    b. Model publisher: D365F&O developer
    c. Layer usr
    d. Version 1.0.0.0
    e. Model description: Dynamics 365 F&O development training - Airlines
    f. Model display name: My Lab Airlines

2. Create a new package
3. Select the following packages as references:
    a. ApplicationFoundation
    b. ApplicationPlatform
    c. ApplicationSuite
    d. ContactPerson
    e. Directory
    f. DynamicsDevTraining
4. Create a new project and make this my default model
5. ~~Exit and re-enter Visual Studio to go into the MyLabAirlines solution~~
6. Create a new Project MyLabAirlines
    a. Model: MyLabAirlines
    b. Company: USMF

Task 2: Create a table: MLAAirportMilesChart

1. Open MyLabAirlines in Solution Explorer
2. Right click project and select **Add > New Item**
3. Select Table under **Dynamics 365 Items > Data Model**
4. Create a new table MLAAirportMilesChart
    a. Label: Airport Miles Chart

b. Cache Lookup: Found

5. Add the following fields
    a. FromAirport
        i. EDT: DDTAirportCode
        ii. Label: From Airport
        iii. Mandatory: Yes
    b. ToAirport
        i. EDT: DDTAirportCode
        ii. Label: To Airport
        iii. Mandatory: Yes
    c. FlyingMiles
        i. EDT: DDTFlyingMiles
        ii. Mandatory: Yes
6. Add the unique index AirportIdx with following fields
    a. FromAirport
    b. ToAirport
7. Create a new Relation: AirportFrom
    a. Related Table: DDTAirport
    b. RelationshipType: Association
    c. Cardinality: ZeroMore
    d. Related Table Cardinality: ZeroOne
    e. On Delete: Restricted
    f. New Normal: MLAAirportMilesChart.FromAirport = DDTAirport.AirportCode
8. Create a new Relation: AirportTo
    a. Related Table: DDTAirport
    b. RelationshipType: Association
    c. Cardinality: ZeroMore
    d. Related Table Cardinality: ZeroOne
    e. On Delete: Restricted
    f. New Normal: MLAAirportMilesChart.ToAirport = DDTAirport.AirportCode

9. Override the validateWrite() method of the table and write the following code

```
public boolean validateWrite()
  {
    boolean ret;
    ret = super();
    MLAAirportMilesChart    milesChart;
    select firstonly milesChart
      where milesChart.FromAirport == this.ToAirport &&
      milesChart.ToAirport == this.FromAirport;
    ret = ret && !milesChart && !(this.FromAirport == this.ToAirport);
    return ret;
  }
```

10. Add a new method getMiles() in the table

```
public static DDTFlyingMiles getMiles(DDTAirportCode _fromAirport, DDTAirportCode _toAirport)
  {
    MLAAirportMilesChart    airportMilesChart;
    select firstonly FlyingMiles
      from airportMilesChart
      where (airportMilesChart.FromAirport == _fromAirport &&
          airportMilesChart.ToAirport == _toAirport) ||
          (airportMilesChart.FromAirport == _toAirport &&
          airportMilesChart.ToAirport == _fromAirport);
    return airportMilesChart.FlyingMiles;
  }
```

Task 3: Create a form: MLAAirportMilesChart

1. Open MyLabAirlines in Solution Explorer
2. Right click project and select **Add > New Item**
3. Select Form under **Dynamics 365 Items > User Interface**
4. Create a new form MLAAirportMilesChart
5. Add table MLAAirportMilesChart as the data source of the form
    a. Property: Index: AirportIdx
    b. Property: Insert If Empty: No
6. Right click **Design | Pattern** and select **Simple List** as form pattern
7. Right click **Design | Pattern** and select **New** and add a new Action Pane
    a. Name: MilesActionPane
8. Right click **Design | Pattern** and select New and add a new Group
    a. Name: MilesFilterGroup
    b. Sub-Pattern: Custom & Quick Filters
    c. Right click **FilterGroup (Group) | Pattern** and add a new QuickFilter
        i. Name: MilesQuickFilter
9. Right click **Design | Pattern** and select **New** and add a new Grid
    a. Name: MilesGridControl
    b. Drag following fields from data source and add in the grid
        i. FromAirport
        ii. ToAirport
        iii. FlyingMiles

Task 4: Create a Display Menu Item: Airport Miles Chart

1. Open MyLabAirlines in Solution Explorer
2. Right click project and select **Add > New Item**
3. Select Display Menu Item under **Dynamics 365 Items > User Interface**

4. Create a new Display menu item MLAAirportMilesChartDisplay
    a. Object Type: Form
    b. Object: MLAAirportMilesChart
    c. Label: *Airport Miles Chart*

Task 5: Menu Extension: AccountsReceivable

1. Open MyLabAirlines in Solution Explorer
2. In the Application Explorer under **User Interface > Menus**, search for AccountsReceivable menu
3. Right click AccountsReceivable and select **Create Extension**
4. A new element will be created in Solution Explorer under Menu Extensions folder named AccountsReceivable.MyLabAirlines
5. Open AccountsReceivable.MyLabAirlines in the designer to drag the display Menu item MLAAirportMilesChartDisplay and drop it under **Setup** submenu

Task 6: EDT: Loyalty Points

1. Open MyLabAirlines in Solution Explorer
2. Right click project and select **Add > New Item**
3. Select EDT Integer under **Dynamics 365 Items > Data Types**
4. Create a new EDT MLALoyaltyPoints (Label: *Loyalty Points*)

Task 7: EDT: Loyalty Percent

1. Open MyLabAirlines in Solution Explorer
2. Right click project and select **Add > New Item**
3. Select EDT Real under **Dynamics 365 Items > Data Types**
4. Create a new EDT MLALoyaltyPercent (Label: *Loyalty Percent*)
    a. Extends: Percent

Task 8: Table Extension: DDTCustFlyDetails

1. Open MyLabAirlines in Solution Explorer
2. In the Application Explorer, search for DDTCustFlyDetails table under Data Model > Tables
3. Right click DDTCustFlyDetails and select **Create Extension**
4. A new element will be created in Solution Explorer under Table Extensions folder named DDTCustFlyDetails.MyLabAirlines
5. Add new field MLALoyaltyPoints in DDTCustFlyDetails.MyLabAirlines by dragging the EDT MLALoyaltyPoints and dropping on the field node of the extended table

Task 9: Form Extension: DDTCustFlyDetails

1. Save all.
2. Open MyLabAirlines in Solution Explorer
3. In the Application Explorer under User Interface > Forms, search for DDTCustFlyDetails form
4. Right click DDTCustFlyDetails and select **Create Extension**
5. A new element will be created in Solution Explorer under Form Extensions folder named DDTCustFlyDetails.MyLabAirlines
6. Under the datasource, find the field MLALoyaltyPoints; drag and add the field on the FlyInfoGrid.
7. Select the FlyingMiles field in the FlyInfoGrid and change the AllowEdit property to No

Task 10: Table Extension: DDTTierRange

1. Open MyLabAirlines in Solution Explorer
2. In the Application Explorer, search for DDTTierRange table
3. Right click DDTTierRange and select **Create Extension**

4. A new element will be created in Solution Explorer under Table Extensions folder named DDTTierRange.MyLabAirlines
5. Add new field MLALoyaltyPercent in DDTTierRange.MyLabAirlines by dragging the EDT MLALoyaltyPercent and dropping it on the fields node of the extended table

Task 11: Form Extension: DDTTierRange

1. Save all.
2. Open MyLabAirlines in Solution Explorer
3. In the Application Explorer, search for DDTTierRange form
4. Right click DDTTierRange and select **Create Extension**
5. A new element will be created in Solution Explorer under Form Extensions folder named DDTTierRange.MyLabAirlines
6. Under the datasource, find the field MLALoyaltyPercent; drag and add the field on the CustTierGrid

Task 12: Chain of Command: Datasource of DDTTierRange form

1. Open MyLabAirlines in Solution Explorer
2. Right click project and select **Add > New Item**
3. Select Class under **Dynamics 365 Items > Code**
4. Create a new class MLATierRangeFormDataSource_Extension with the following signature

```
[ExtensionOf(formdatasourcestr(DDTTierRange, DDTTierRange))]
final class MLATierRangeFormDataSource_Extension
{
}
```

5. Add new validation within that class by creating a chain of command for the validateWrite() method of the datasource, which is already overridden in the base class. Following is the code:

```
public boolean validateWrite()
 {
    boolean ret;
    ret = next validateWrite();
    FormDataSource  formDS = this;
    DDTTierRange    tierRangeBase = formDS.cursor();
    DDTTierRange    tierRange;
  if(tierRangeBase)
  {
            select RecId from tierRange
            where (tierRange.FromMiles > tierRangeBase.FromMiles &&
                tierRange.FromMiles > tierRangeBase.ToMiles) ||
                (tierRange.ToMiles < tierRangeBase.FromMiles &&
                tierRange.ToMiles < tierRangeBase.ToMiles);
            ret = ret && tierRange.RecId;
    }
    return ret;
 }
```

Task 13: Table DDTCustFlyDetails > modifiedField Method

1. Open MyLabAirlines in Solution Explorer
2. Right click project and select **Add > New Item**
3. Select Class under **Dynamics 365 Items > Code**
4. Create a new class MLACustFlyDetailsEventHandler
5. Open table DDTCustFlyDetails.MyLabAirlines in the designer pane

6. Under the Events node, find the onModifiedField event; right-click and select Copy event handler method
7. Paste the method signature in the new class MLACustFlyDetailsEventHandler

```
/// <summary>
///
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
[DataEventHandler(tableStr(DDTCustFlyDetails), DataEventType::ModifiedField)]
public static void DDTCustFlyDetails_onModifiedField(Common sender, DataEventArgs e)
{
}
```

8. Paste the following piece of code within the method:

```
ModifyFieldEventArgs    event = e as DataEventArgs;
    DDTCustFlyDetails      custFlyDetails = sender as DDTCustFlyDetails;
    FieldId                fieldId = event.parmFieldId();
    switch(fieldId)
    {
       case fieldNum(DDTCustFlyDetails, FlyFrom):
       case fieldNum(DDTCustFlyDetails, FlyTo):
          if(custFlyDetails.FlyFrom && custFlyDetails.FlyTo)
          {
             custFlyDetails.FlyingMiles = MLAAirportMilesChart::getMiles(DDTAirport::getAirportCode(custFlyDetails.FlyFrom),
DDTAirport::getAirportCode(custFlyDetails.FlyTo));
```

```
                custFlyDetails.MLALoyaltyPoints = custFlyDetails.FlyingMiles *
(DDTTierRange::find(CustTable::find(custFlyDetails.CustAccount).DDTCustomerTier).MLALoyaltyPercent /100);

            }

            break;

        }
```

Check Output

Scenario - 2

- Save all and build.
- Under **Accounts Receivables > Setup**, find the **Airport Miles Chart** form. Enter the following data in that form:

From Airport	To Airport	Flying Miles
DEL	BOM	400
DEL	HYD	500
DEL	BLR	1000
DEL	MAS	1000
BOM	HYD	300
BOM	BLR	400
BOM	MAS	500
HYD	BLR	200
HYD	MAS	300
BLR	MAS	200

- Open the Flying Details fast tab under **Accounts Receivables > Customers > All Customers** and find the new column “Loyalty Points”
- In the Miles wise Tier Range form find the new field Loyalty Percent. This percentage value will be multiplied by the Miles to yield the Loyalty Points for the customer. Enter values between 1 and 100.
- In the Customer Flying Miles tab of the Customer form, create a new record and enter travel date, From Airport and To Airport. Flying Miles should automatically populate form the Airport Miles Chart along with the Loyalty Points. Use the table above.