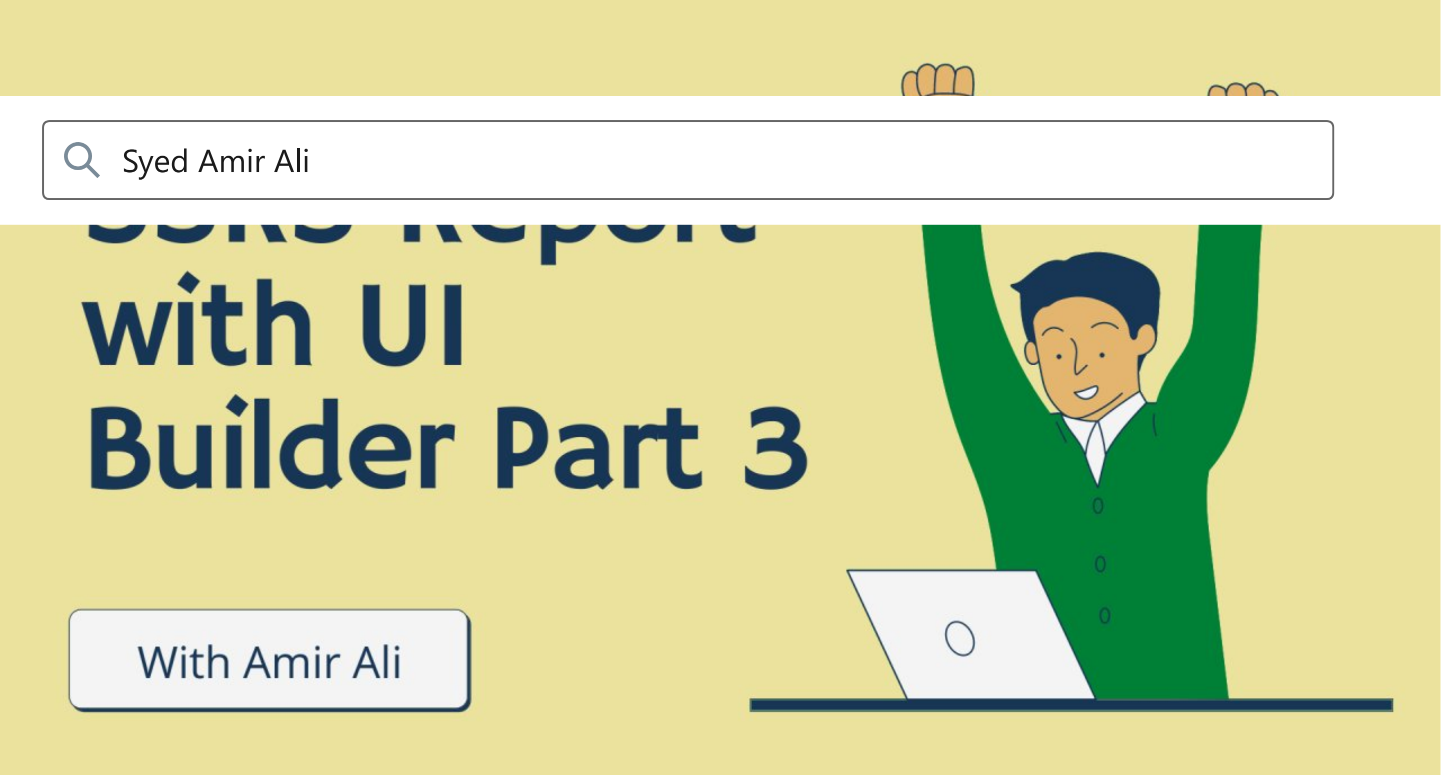
Syed Amir Ali



Create a Report using UI Builder class in Dynamics 365 Finance and Operation

Syed Amir Ali
Microsoft Certified Dynamics 365 Finance & Operations Solution Architect | MCT | Team lead | Senior Software Engineer at Confiz

+ Follow

Published Aug 14, 2021

Hello, I welcome you once again in learning SSRS Report development in Dynamics 365 F&O. Today, we do code together and work with UI Builder class and Multi-select lookup and use the **PurchTable**. The theoretical part is already done in part 2. In case of missed that, click the link here

Contract class

```
[DataContractAttribute,

SysOperationContractProcessingAttribute(classStr(PurchaseUIBuilder))

 ]
```

```
public class PurchaseContractClass

{

  TransDate        FromDate, ToDate;

  List             purchId;

 [DataMemberAttribute('From date')]

  public TransDate ParmFromDate(FromDate _FromDate=FromDate)

  {

     FromDate = _FromDate;

     return FromDate;

  }



  [DataMemberAttribute('To date')]

  public TransDate ParmToDate(ToDate _ToDate=ToDate)

 {

     ToDate = _ToDate;
```

```
    return ToDate;
  }
  [
    DataMemberAttribute('PurchaseId'),
    AifCollectionTypeAttribute('PurchaseId', Types::String),
    SysOperationLabelAttribute(literalStr("PurchaseId")),
    SysOperationDisplayOrderAttribute('3')

  ]
  public List parmPurchId(List _purchId = purchId)
  {
    purchId =_purchId;
    return purchId;
  }
}
```

Explanation:

Now create a contract class, which is responsible for getting and setting the data, in which use the fields to set parameters in the form to get the condition from the user.

Like in the above contract class we define three attributes, two is of type TransDate and one is of List type to save the multiple purchase Ids's as the purchase ids are MultiSelect and define the getter setter for the fields, one point to remember is to mention the “DataContractAttribute” which differentiates the Contract class from other classes. Like this

One more parameter is to declare on Contract class, is that as we are making a report using UI Builder. So we have to add the reference for the UI Builder class. Like in this example, I add the reference

SysOperationContractProcessingAttribute (classStr (PurchaseUIBuilder))

So that, the functionality we add in the UI Builder class has to reference there.

Now create a temp table for the report to display and create fields in it which you want to use in the Report. Like in the figure below, I use two fields Purchase Id, Purchase Name. One point to remember is that change the Table Type to **TempDB** in the Table Properties.

RDP class

```
[

   SRSReportParameterAttribute(classstr(PurchaseContractClass))
```

```
]

public class PurchaseDpClass extends SrsReportDataProviderPreProcessTempDB

{

  ListEnumerator            PurchListIterator;



  PurchaseContractClass      Contract;

  PurchaseOrderTmp           purchaseOrderTmp;

  TransDate                  _FromDate, _ToDate;

  List                       _purchId;

  PurchTable                 purchTable;

  PurchId                    purchId;

  [SrsReportDataSetAttribute('PurchaseOrderTmp')]

  public PurchaseOrderTmp GetData()

  {

    select * from purchaseOrderTmp;

    return purchaseOrderTmp;
```

```
}

public void processReport()

{

    Contract = this.parmDataContract();

    _FromDate = Contract.ParmFromDate();

    _ToDate = Contract.ParmToDate();

    _purchId=Contract.parmPurchId();

    if (_purchId != null)

    {

        PurchListIterator = _purchId.getEnumerator();

        while (PurchListIterator.moveNext())

        {

            purchId = PurchListIterator.current();

            while select purchTable where purchId ==purchTable.PurchId

            {

         purchaseOrderTmp.PurchaseID=purchTable.PurchId;
```

```
            purchaseOrderTmp.PurchaseName=purchTable.PurchName;

            purchaseOrderTmp.insert();



              }

            }

        }

    }

}
```

I am not going into details on the RDP class code, because it is already communicated in the last part.

Now create a class UI Builder class in which write all the functionality. As we are developing the report in which three-parameter are there and the **purchase Id** is dependant on the other two parameters: **From Date** and **To Date**. As the drop-down is filled according to **From Date** and **To Date**. So we have to write the build-in functionality according to our requirements.

For this, we add the Dialog fields for all the parameters and add a reference for the contract class. There are three main functions which we have to write are: **Build**, **postBuild**, **postRun**. There is one **lookup function** to fill the Purchase Ids' according to **From Date** and **To Date** and we write the leave function for it as well.

```
public class PurchaseUIBuilder extends SysOperationAutomaticUIBuilder{

  PurchaseContractClass             _contract;



  DialogField                    DialogFromDate;

  DialogField                    DialogToDate;

  DialogField                    DialogpurchId;

  date                         dateFrom;

  date                         dateTo;

  public void build(){

     _contract   = this.dataContractObject();

     dialogFromdate = this.addDialogField(methodStr(PurchaseContractClass, parmFromDate), _contract);

     dialogTodate   = this.addDialogField(methodStr(PurchaseContractClass, parmToDate), _contract);

     DialogpurchId   = this.addDialogField(methodStr(PurchaseContractClass, parmPurchId), _contract);
  }

  public void postBuild(){

    super();

    _contract = this.dataContractObject();

    DialogpurchId= this.bindInfo().getDialogField(_contract,
methodStr(PurchaseContractClass, parmPurchId));

    DialogpurchId.registerOverrideMethod(methodStr(FormStringControl,
lookup),methodStr(PurchaseUIBuilder, purchIdLookup), this);

    DialogFromDate = this.bindInfo().getDialogField(_contract,
methodStr(PurchaseContractClass, ParmFromDate));

    DialogFromDate.registerOverrideMethod(methodStr(FormDateControl, leave),
methodStr(PurchaseUIBuilder, dateFromLeave), this);

    DialogToDate = this.bindInfo().getDialogField(_contract,
methodStr(PurchaseContractClass, ParmToDate));

    DialogToDate.registerOverrideMethod(methodStr(FormDateControl, leave),
methodStr(PurchaseUIBuilder, dateToLeave), this);

  }

  public void postRun()
  {

  }
```

Lookup and Leave method

Now we explain the lookup method in which we write the lookup for purchase table in which render the purchase Ids according to the date range and get the data from **Purch Table** according to **CreatedDateandTime.**

In the Lookup method, there is a DataSource reference to get the table values and set the value from that DataSource.

For **the leave method**, therefore, To Date and From Date. We save the value from the FromDate and ToDate parameter and get their value and save it in the fields.

Now use these values to get the purchase Ids and render them on the dropdown. At the end MultiSelect the Purchase ids, we write the line

SysLookupMultiSelectGrid::lookup (query,_control, _control, _control, conNull());

```
  public boolean dateFromLeave(FormDateControl _control)

  {

     if (_control.valueStr() != '')

     {
        dateFrom = _control.dateValue();
    }
    return true;
}
public boolean dateToLeave(FormDateControl _control)
{
    if (_control.valueStr() != '')
    {
        dateTo = _control.dateValue();
    }
    return true;
}
private void purchIdLookup(FormStringControl _control)
{
    ListEnumerator          enum;
    Query               query = new query();
    QueryBuildDataSource          queryBuildDataSource;

    queryBuildDataSource = query.addDataSource(tableNum(PurchTable));

    queryBuildDataSource.fields().dynamic(false);

    queryBuildDataSource.fields().clearFieldList();

    queryBuildDataSource.addSelectionField(fieldNum(PurchTable, PurchId));

    if (!dateFrom)

    {

        dateFrom = DialogFromDate.value();

    }

    if (!dateTo)

    {

        dateTo = DialogToDate.value();

    }

    queryBuildDataSource.addRange(fieldNum(PurchTable,
CreatedDateTime)).value(SysQuery::range(DateTimeUtil::newDateTime(dateFrom,0),D

    SysLookupMultiSelectGrid::lookup(query,_control, _control, _control,
conNull());
}
```

Now, add the new item > Report and create the design as shown in below figure.

Add the menu item and menu extension and attach it as we done in part 1. The final output will show like below fig.

Happy Learning

Syed Amir Ali

Like Comment Share

To view or add a comment, sign in

More articles by this author

See all

Fixing the SQL server bug after upgrading…
Aug 1, 2022

Use of JumpRef() method in Dynamics…
Jul 26, 2022

Delete Global Number Sequence in x++…
Jun 15, 2022

Others also viewed

Custom SSRS REPORT IN DYNAMICS 365 F&O || AX 2012 R3
Syed Amir Ali · 1y

Custom SSRS Report in Dynamics 365 Finance and Operation (Part II)
Syed Amir Ali · 1y

Create multi select lookup using enums in AX 2012
Udhaya Chandran R. · 2y

Dynamics 365 option is not showing in visual studio
Syed Amir Ali · 1y

Excel import in Dynamics AX 365 or AX7 using Dialog
Shashi Kant Yadav · 6y

Some tips for filtering dates in D365FO
Hylke Britstra · 2mo

Integrations with Dynamics 365 for Finance and Operations

P V Paparao Kancharapu · 3y

Add range/filter to queries in D365 Fin&Ops

Tazeen Zaidi · 1y

Troubleshooting "The network path was not found"AX Reports Deployment

Shafeaa' AlLabadi, MCP · 7y

How to add a pre-filter to SSRS Report using Fetch-XML

Mohamed Elgharably · 1y

Explore topics

Workplace

Job Search

Careers

Interviewing

Salary and Compensation

Internships

Employee Benefits

See All