**Dynamic query :**

1. **To Select All Customers from Cust table –**

**static** **void** Job4(Args \_args)

{

Query q; // declare query/primary method

QueryRun qr; // to run the query

QueryBuildDataSource qbd; // to add table to data source

Account acc; // declare table with buffer name

q = **new** Query(); // create obj. for query

qbd = q.addDataSource(**tableNum**(Account)); // adding table to ds

qbd.addSortField(**fieldNum**(Account, //records will be sorted in-

AccountNum),SortOrder::Ascending); // -ascending order

qr = **new** queryRun(q);

**while**(qr.next()) // to fetch next record

{

acc = qr.get(**tableNum**(Account)); // assigning selected record-

info(**strFmt**("%1", acc.AccountNum)); // -to buffer-table

} // AccountNo. Will store in %1

}

**Output** :- It will show all the accountnum data in ascending order

1. **To Select Customers Between Range -:**

static void Job4(Args \_args)

{

Query q;

QueryRun qr;

QueryBuildDataSource qbds;

QueryBuildRange qbr; // to give range

Account account;

q = new Query();

qbds = q.addDataSource(tableNum(account));

qbds.addSortField(fieldNum(account ,AccountNum),SortOrder::Ascending);

qbr = qbds.addRange(fieldNum(account, AccountNum));

qbr.value(queryRange('542','1245')); // we gave 2 ranges

qr = new queryRun(q);

while(qr.next())

{

account = qr.get(tableNum(account));

info(strFmt("%1", account.AccountNum));

}

}

Output :- it will show accountnum between range 542 to 1245

1. **Where Condition - same as above code**
2. **CrossCompany --** Suppose you are login with Company 'A' and you want to fetch the data for company 'B'

**static** **void** Job4(Args \_args)

{

Query q;

QueryRun qr;

QueryBuildDataSource qbds;

Account account;

q = **new** Query();

q.allowCrossCompany (**true**); //setting the prpty savedatapercompany to yes

qbds = q.addDataSource(**tableNum**(account));

qr = **new** queryRun(q);

**while**(qr.next())

{

account = qr.get(**tableNum**(account));

info(**strFmt**("%1-%2", account.AccountNum,account.dataAreaId));

}

}

Output – it will show accountnum data with 542-usmf

1. **Aggregate Function -- min, max, sum, count, computedcolumn, database**

**static** **void** Job4(Args \_args)

{

Query q;

QueryRun qr;

QueryBuildDataSource qbds;

QueryBuildRange qbr;

Account account;

q = **new** Query();

qbds = q.addDataSource(**tableNum**(account));

qbds.addSelectionField(**fieldNum**(account, // min accoNo. Will calculated AccountNum),Selectionfield::**Min**);

*//qbds.addSelectionField(fieldNum(account, RecId)SelectionField::Count);*

qr = **new** queryRun(q);

**while**(qr.next())

{

account = qr.get(**tableNum**(account));

info(**strFmt**("%1", account.AccountNum));

}

}

Output -: it will show minimum accountnum Ex:- min:- 542 , max:- 216546 , sum:- 233217 , count:- 6(records) , computedcolumn , database :- show all data

1. **first only,first fast :--**

static void Job4(Args \_args)

{

int i;

Query q;

QueryRun qr;

QueryBuildDataSource qbds;

QueryBuildRange qbr;

Account account;

*//Table12 ttwel;*

*//Table11 telew;*

q = new Query();

qbds = q.addDataSource(tableNum(account));

qbds.addSortField(fieldNum(account, AccountNum),SortOrder::Descending);

qbds.firstOnly(true); *// here we can add firstOnly, firstFast*

qr = new queryRun(q);

while(qr.next())

{

account = qr.get(tableNum(account));

info(strFmt("%1", account.AccountNum));

}

}

Output :- only first account number will show 216546 , first Fast :- all record will show

1. **forUPdate :- for updating names Ex:-ved = “shivakumar”**

**static** **void** Job4(Args \_args)

{

**int** i;

Query q;

QueryRun qr;

QueryBuildDataSource qbds;

QueryBuildRange qbr;

Account account;

q = **new** Query();

qbds = q.addDataSource(**tableNum**(Account));

qbds.addSortField(**fieldNum**(Account, Name));

qr = **new** queryRun(q);

**while**(qr.next())

{

account = qr.get(**tableNum**(Account));

account.selectForUpdate(**true**);

**ttsBegin**;

**if**(account.Name == "ved")

{

account.Name = "Sivakumar";

account.update();

}

**ttsCommit**;

info(**strFmt**("%1", account.Name));

}

}

**Output:- it will update name ved to shivakumar**

1. **GroupBy :- will group the rows that have same values in a row**

**static** **void** Job4(Args \_args)

{

Query q;

QueryRun qr;

QueryBuildDataSource qbds;

QueryBuildRange qbr;

Account account;

q = **new** Query();

qbds = q.addDataSource(**tableNum**(Account));

qbds.addSelectionField(**fieldNum**(Account, Name),SelectionField::**Max**);

qbds.addGroupByField(**fieldNum**(Account, Name),OrderMode::GroupBy);

qr = **new** queryRun(q);

**while**(qr.next())

{

account = qr.get(**tableNum**(Account));

info(**strFmt**("%1-%2", account.Name, account.AccountNum));

}

}

Output :- it will perform groupby here i.e if there are two same name then it wills count as once

1. **Joins :- inner join with two tables (by taking 4 tables)**

static void Job2(Args \_args)

{

Query q;

QueryRun qr;

QueryBuildDataSource qbdsAccount, qbdsEmployeeT;

QueryBuildRange qbr;

Account account;

EmployeeT empt;

Table12 ttwl;

Table11 tel;

q = new Query();

qbdsAccount = q.addDataSource(tableNum(Table12)); // add main datasource

qbdsEmployeeT = qbdsAccount.addDataSource(tableNum(Table11));

qbdsEmployeeT.relations(false);

qbdsEmployeeT.joinMode(JoinMode::InnerJoin);

qbdsEmployeeT.addLink(fieldNum(Table12, Emp\_Id),fieldNum(Table11, Emp\_Id)); // link between parent field and child field

qr = new QueryRun(q);

while(qr.next())

{

ttwl = qr.get(tableNum(Table12));

tel = qr.get(tableNum(Table11));

info(strFmt("%1-%2", ttwl.Emp\_Id, tel.Emp\_Id));

}

}

Output :- it will show all common emp\_id Ex:- 1-1 , 2-2 (we can perform left join , right join, outer join)

1. **Multiple Tables Join:- we are joining 3 tables with each other(2nd with 1st & 2nd with 3rd)**

**static** **void** Job2(Args \_args)

{

Query q;

QueryRun qr;

QueryBuildDataSource ttwlds;

QueryBuildDataSource telds;

QueryBuildDataSource emptds;

EmployeeT empt;

Table12 ttwl;

Table11 tel;

q = **new** Query();

ttwlds = q.addDataSource(**tableNum**(Table12));

telds = ttwlds.addDataSource(**tableNum**(Table11));

telds.relations(**false**);

telds.joinMode(JoinMode::InnerJoin);

telds.addLink(**fieldNum**(Table12, Emp\_Id),**fieldNum**(Table11, Emp\_Id));

emptds = ttwlds.addDataSource(**tableNum**(EmployeeT));

emptds.relations(**false**);

emptds.joinMode(JoinMode::InnerJoin);

emptds.addLink(**fieldNum**(Table12, Emp\_Id),**fieldNum**(EmployeeT, Emp\_Id));

qr = **new** QueryRun(q);

**while**(qr.next())

{

ttwl = qr.get(**tableNum**(Table12));

tel = qr.get(**tableNum**(Table11));

empt = qr.get(**tableNum**(EmployeeT));

info(**strFmt**("%1-%2-%3", ttwl.Emp\_Id, tel.Emp\_Id, empt.Emp\_Id));

}

}

Output :- it will join table with their Emp\_Id Ex:- 1-0-0, 1-1-0, 1-0-1, 2-0-0, 2-2-0, 2-0-2

1. **Validtimestate :- didn’t try this code**

static void QueryCurrent(Args \_args)  
{  
     CustInterestVersion interestVersion;  
     CustInterest interest;  
     Date asOfDate = 1\1\2002;  
  
 while select validtimestate(asOfDate) \* from interestVersion join interest  
       where interestVersion.CustInterest == interest.RecID  
      {  
          info(strFmt("%1, %2, %3, %4", interest.InterestCode, interestVersion.GraceDays,  
        interestVersion.ValidFrom, interestVersion.ValidTo));  
      }  
}

1. **Full Text Index :-** we have to change table property to main, and create full text index

it will find how many same name (shiva) are there in the table

**static** **void** Job2(Args \_args)

{

Query q;

QueryRun qr;

QueryBuildDataSource qbds;

QueryBuildRange qbr;

Account account;

q = **new** Query();

qbds = q.addDataSource(**tableNum**(Account));

qbr = qbds.addRange(**fieldNum**(Account,Name));

qbr.rangeType(QueryRangeType::FullText);

qbr.value("shiva");

qr = **new** QueryRun(q);

**while**(qr.next())

{

account = qr.get(**tableNum**(Account));

info(account.Name);

}

}

Output :- search in table how many same name are there in table

* **QueryFilter** :- to filter the records from database

**static** **void** Job2(Args \_args)

{

Query q;

QueryRun qr;

QueryBuildDataSource Table12DS;

QueryBuildDataSource Table11DS;

QueryBuildRange qbr;

QueryFilter qf;

Table12 ttwel;

Table11 tel;

q = **new** Query();

Table11DS = Table12DS.addDataSource(**tableNum**(Table11));

Table11DS.joinMode(JoinMode::InnerJoin);

Table11DS.addLink(**fieldNum**(Table12, Emp\_Id),**fieldNum**(Table11, Emp\_Id));

qf = q.addQueryFilter(Table11DS, **fieldStr**(Table11,Emp\_Id));

qf.value('1');

qr = **new** QueryRun(q);

**while**(qr.next())

{

ttwel = qr.get(**tableNum**(Account));

tel = qr.get(**tableNum**(EmployeeT));

info(**strFmt**("%1-%2", ttwel.Emp\_Id, tel.Emp\_Id));

}

}

Output :- match record will be display Ex: 00006 --- 00006

* **QueryHavingFilter :** it will display how many group of same name are there in a table (datasource)

**static** **void** Raj\_Queryhavingfilter(Args \_args)

{

Query q;

QueryBuildDataSource qbds;

QueryBuildRange qbr;

QueryHavingFilter havingFilter;

QueryRun qr;

**int** counter = **0**, totalCounter = **0**;

EmployeeT empt;

q = **new** Query();

qbds = q.addDataSource(**tableNum**(EmployeeT));

qbds.addSelectionField(**fieldNum**(EmployeeT, RecId),

SelectionField::Count);

qbds.orderMode(OrderMode::GroupBy);

qbds.addGroupByField(**fieldNum**(EmployeeT, FirstName));

havingFilter = q.addHavingFilter(qbds, **fieldStr**(EmployeeT, RecId),

AggregateFunction::Count);

havingFilter.value('< 3');

qr = **new** QueryRun(q);

**while** (qr.next())

{

empt = qr.getNo(**1**);

info(**strFmt**("Group %1: %2", empt.FirstName, empt.RecId));

}

}

**Output :- Group kavya -1**

**Group mira -1**

**Group ziva -2 ( because 2 ziva name present in the table)**

#### Query Group By Field : it will count how many no. of rows are there in table

**static** **void** Job2(Args \_args)

{

query q;

QueryBuildDataSource qbds;

QueryGroupByField qgf;

QueryOrderByField qof;

QueryRun qr;

EmployeeT empt;

q = **new** Query();

qbds = q.addDataSource(**tableNum**(EmployeeT));

qbds.addSelectionField(**fieldNum**(EmployeeT, Emp\_Id),SelectionField::Count);

qgf = qbds.addGroupByField(**fieldNum**(EmployeeT, FirstName));

qr = **new** QueryRun(q);

info(q.toString());

**while**(qr.next())

{

empt = qr.get(**tableNum**(EmployeeT));

info(**strFmt**("%1 - count %2",empt.FirstName, empt.Emp\_Id));

}

}

#### Query Order By Field : it will ordered the records in ascending /descending order

static void Job2(Args \_args)

{

query q;

QueryBuildDataSource qbds;

queryBuildFieldList qbfl1, qbfl2;

QueryRun qr;

Table12 ttwel;

Table11 tel;

EmployeeT empt;

q = new Query();

qbds = q.addDataSource(tableNum(EmployeeT));

qbds.addOrderByField(fieldNum(EmployeeT,FirstName),SortOrder::Ascending);

qr = new QueryRun(q);

info(q.toString());

while(qr.next())

{

empt = qr.get(tableNum(EmployeeT));

info(strFmt("%1",empt.FirstName));

}

}

Output : perform order by in ascending order

#### QueryBuildFieldList :-   define which field are return from database

static void Job2(Args \_args)

{

query q;

QueryBuildDataSource qbds;

queryBuildFieldList qbfl1, qbfl2;

QueryRun qr;

Table12 ttwel;

Table11 tel;

EmployeeT empt;

q = new Query();

qbds = q.addDataSource(tableNum(EmployeeT));

qbds.fields().dynamic(NoYes::Yes);

qbfl1 = qbds.fields().addField(fieldNum(EmployeeT, Emp\_Id));

qbfl2 = qbds.fields().addfield(fieldNum(EmployeeT, FirstName));

qr = new QueryRun(q);

info(q.toString());

while(qr.next())

{

empt = qr.get(tableNum(EmployeeT));

info(strFmt("%1-%2",empt.Emp\_Id, empt.FirstName));

}

}

#### Output : it will come like normal we choose fields from DB Ex: 1 - kavya

#### Query Build link :- define relation between 2 datasources in the join

static void Job2(Args \_args)

{

query q;

QueryBuildDataSource qbds1, qbds2;

QueryBuildRange qbr;

QueryBuildlink qbl;

QueryRun qr;

Table12 ttwel;

Table11 tel;

q = new Query();

qbds1 = q.addDataSource(tableNum(Table12));

qbr = qbds1.addRange(fieldNum(Table12, Emp\_Id));

qbr.Value('2');

qbds2 = qbds1.addDataSource(tableNum(Table11));

qbl = qbds2.addLink(fieldNum(Table12, Emp\_Id),fieldNum(Table11, Emp\_Id));

qr = new QueryRun(q);

info(q.toString());

while(qr.next())

{

ttwel = qr.get(tableNum(Table12));

tel = qr.get(tableNum(Table11));

info(strFmt("%1-%2",ttwel.Emp\_Id, tel.Emp\_Id));

}

}

#### Output :- it will join 2 datasources and then will show common from both table which we give range Ex:- 2 – 2

### **QueryBuildDynaLink** :- contain information regarding relation to an external record

static void Job2(Args \_args)

{

query q;

QueryBuildDataSource qbds;

QueryBuildDynalink qbdl;

QueryRun qr;

Account account;

EmployeeT empt;

q = new Query();

qbds = q.addDataSource(tableNum(Account));

select firstOnly empt where empt.FirstName =="mira";

qbds.addDynalink(fieldNum(Account, Name),empt,fieldNum(EmployeeT, FirstName));

qr = new QueryRun(q);

while(qr.next())

{

account = qr.get(tableNum(Account));

setPrefix('Customer -' + account.Name);

info(strFmt("%1",account.Emp\_Id));

info(strFmt("%1",empt.FirstName));

}

}

**Output** :- Customer mira

-1

-mira

#### Container :- Containers are used to store different datatypes   - Max columns in a container - 50

1. **Conpeek() :** used to retrieve a specific element from a container

static void Job2(Args \_args)

{

container c = ['Test',43,"shravya",20.000];

info(strFmt("%1-%2", conPeek(C, 1),conPeek(C, 2)));

info(strFmt("%1-%2", conPeek(C, 1),conPeek(C, 3)));

info(strFmt("%1-%2", conPeek(C, 1),conPeek(C, 4)));

}

**Output :- first it will add info. In container Ex: c=[‘Test’,43,”shravya”,20.000]**

**& then with help of conpeek function we are retrieving same data from container**

**Ex:- 43 shravya 20.00 🡨 will display on the form**

**2)Conpoke(): used To replace any value in container**

**static** **void** Job2(Args \_args)

{

**container** c = ["shravya", **123**];

c = **conPoke**(C, **2**, **143143**);

info(**strFmt**("%1-%2", **conPeek**(C, **1**),**conPeek**(C, **2**)));

}

**Output :- it will replace 123 to 143143 EX:-** shravya – 143143

**3)ConIns ():** used to insert one or more elements into a container

**static** **void** Job2(Args \_args)

{

**container** c = ["shravya", **111**];

c = **conIns**(C, **3**, **4444**);

info(**strFmt**("%1-%2-%3", **conPeek**(C, **1**),**conPeek**(C, **2**),**conPeek**(c, **3**)));

}

**Output : it will add another element in container Ex:** shravya 111 4444

4) **ConFind()**: it will find the position of element

**static** **void** Job2(Args \_args)

{

**container** c = ["shrvya", **111**];

info(**strFmt**("111 is found at Position %1 - shravya is found at Position %2",

**conFind**(c,**111**),**conFind**(c, "shravya")));

}

**Output** :- 111 is found at Position 1 shravya is found at Position 2

**5) Conlen ():** it will find how many element are there in the container

**static** **void** Job2(Args \_args)

{

**container** c = ["shravya", **111**, 'G', **34.98**];

info(**strFmt**("Length of container is %1",**conlen**(c)));

}

Output :- length of container is 4 (how many total elements)

6) **ConDel():** use to delete specific number from container

**static** **void** Job2(Args \_args)

{

**container** c = ["shravya", **1234**];

c = **conDel**(c, **2**, **1**);

info(**strFmt**("%1-%2",**conpeek**(c, **1**),**conPeek**(C, **2**)));

}

**Output :** it will delete element 12345 from position 2nd

**7) Connull ():** it will empty the container

**static** **void** Job2(Args \_args)

{

**container** con = ["Hello","Welcome","To","Ax",**2012**,"R3"];

info(**strFmt**("Container conNull() : %1 - %2 - %3 - %4- %5 -%6",

**conPeek**(con,**1**),**conPeek**(con,**2**),**conPeek**(con,**3**),**conPeek**(con,**4**),**conpeek**(con,**5**),**conPeek**(con,**6**)));

con = **conNull**();

info(**strFmt**("Container conNull() after : %1 - %2 - %3 - %4 - %5 - %6",**conPeek**(con,**1**),**conPeek**(con,**2**),**conPeek**(con,**3**),**conPeek**(con,**4**),

**conpeek**(con,**5**),**conPeek**(con,**6**)));

}

Output : it will empty the container means it will delete all elements from the container Ex: 0-0-0-0-0-0

8) Conview : to view the element in container

**static** **void** Job2(Args \_args)

{

**container** c = ["Hello","Welcome","To","Ax",**2012**,"R3"];

conView(C);

}

Output :- Hello

Welcome

To

Ax

* **Method’s:-**
* **Insert():-**it willinsert info. in the table

**static** **void** Job2(Args \_args)

{

EmployeeT empt;

empt.Emp\_Id = **5**;

empt.FirstName = "kashmira";

empt.LastName = "shah";

empt.insert();

info(**strFmt**("%1-%2-%3", empt.Emp\_Id,empt.FirstName,empt.LastName));

}

**Output**: (5 Kashmira shah ) will add in the table at last

* **Update():-** it will change 1 id with other id

static void Job2(Args \_args)

{

EmployeeT empt;

ttsBegin;

select forUpdate empt

where empt.Emp\_Id == 5;

empt.Emp\_Id = 4;

empt.update();

ttsCommit;

}

**Output**: update id 5 with 4

* **Update with while :-**used for multiple updates

static void Job2(Args \_args)

{

EmployeeT empt;

ttsBegin;

select forUpdate empt

where empt.Emp\_Id == 5;

{

empt.Emp\_Id = 7;

empt.update();

}

select forUpdate empt

where empt.Emp\_Id == 4;

{

empt.Emp\_Id = 9;

empt.update();

}

}

**Output** : it will change 5 to 7 and 4 to 9

* **Delete():-** it will delete the whole record from table

static void Job2(Args \_args)

{

EmployeeT empt;

ttsBegin;

select forUpdate empt

where empt.Emp\_Id == 4;

{

empt.delete();

}

ttsCommit;

}

Output: delete 4th record from the table

* **Delete with while():** used for multiple delete

static void Job2(Args \_args)

{

EmployeeT empt;

ttsBegin;

while select forUpdate empt

where empt.Emp\_Id == 4

{

empt.delete();

}

while select forUpdate empt

where empt.Emp\_Id == 5

{

empt.delete();

}

ttsCommit;

}

Output : record 4th and 5th will be deleted from table

* **Insert recordset:-** it will copies data from table1 and inserted to another table

static void Job2(Args \_args)

{

Employee emp;

EmpDetails empd;

ttsBegin;

insert\_recordset emp(Emp\_Id, Name)

select Emp\_Id, Name

from empd

where empd.Emp\_Id > 0;

ttsCommit;

}

Output: copy data from detail table to emp table (add at last of first tables records)

* **Update recordset:-** used for multiple updates at a time then

static void Job2(Args \_args)

{

Employee emp;

update\_recordSet emp

setting

Emp\_Id = 1

where emp.Emp\_Id <= 2;

}

Output: change all id-2 to id-1

* **Delete recordset:-** if you want to delete multiple records at a time then

static void Job2(Args \_args)

{

Employee emp;

delete\_from emp

where emp.Emp\_Id <= 2;

}

Output: it will delete the record 2 from emp table

* **Modified field:-** if we change value of field 1 another field will automatically change (or we can set default name)

public void Job2(FieldId \_fieldId)

{

switch(\_fieldId)

{

case fieldnum(EmpDetails, Emp\_Id):

this.Name = "gargi";

break;

default:

super(\_fieldId);

}

}

Output: whenever we change id field, name field will automatically set to gargi

* **Validate field:-**used for validation if we enter <2 char. It will show warning msg to re-enter fieldname

public boolean validateField(FieldId \_fieldIdToCheck)

{

boolean ret;

ret = super( \_fieldIdToCheck);

if(ret)

{

switch ( \_fieldIdToCheck)

{

case fieldNum(Employee, Name):

if(strLen(this.Name)<=3)

ret = checkFailed("Customer Name must be longer than 3 character");

}

}

return ret;

}

Output: if we enter only <3 character in name field it will show warning msg to(Customer Name must be longer than 3 character)

* **Validate Write:-**it will check mandatory (if we didn’t write anything in the name field it will show warning msg)

public boolean validateWrite()

{

boolean ret;

if (this.Name != "")

ret = super();

else

warning("Please fill the name field");

return ret;

}

Output: if we didn’t write anything (keep blank) in name field it will show msg (plz fill name field)

* **Validate Delete:-**it will shows info. About what you delete (if we want to put any validation we can put here)

public boolean validateDelete()

{

boolean ret;

ret = super();

info(this.Name);

return ret;

}

Output: if we delete any record it will show what info we delete Ex: Noah

* **Display:-**used to display the records on the form/reports

We will write only on Forms methods

Display Name getNameMethod()

{

Employee emp;

select Name from emp

where emp.Name == this.Name;

return emp.Name;

}

Output: it will display the table (if we try to edit record msg will throw by system

* **Display method with 2 tables:-**

Display Name LookupBankGroup()

{

Employee emp;

EmpDetails empd;

select Name from empd

join emp

where empd.Emp\_Id == emp.Emp\_Id && emp.Name == this.Emp\_Id;

return empd.Name;

}

Output : 1 error

* **Edit:-**we can edit(change)in this method

We will write only on Forms methods

Edit Name EditName(boolean set, Name names)

{

Name name = names;

Employee emp;

if(Set)

{

if(Name)

{

ttsBegin;

select forUpdate Name from emp

where emp.Name == this.Name;

emp.Name = names;

emp.update();

ttsCommit;

}

else

{

select Name from emp where emp.Name == this.Name;

names = emp.Name;

}

}

return name;

}

Output:we can edit the record in this method

* **Find method:-** write code in Table methods--first then in job

**static** Table12 find(Name name, **boolean** \_forupdate = **false**)

{

Table12 tt;

**if**(name)

{

tt.selectForUpdate(**false**);

}

**select** **firstOnly** tt

**where** tt.Name == name;

**return** tt;

}

**Write this code in job(find method):-**

**static** **void** Job18(Args \_args)

{

Table12 tt;

tt = Table12::find('mm');

**if**(tt)

{

info("Record Exist");

}

**else**

{

info("Not Exist");

}

}

Output:-Record Exist (the name value present in the table)

* **Exist method:-** Write this code in table method

**static** **boolean** exist(Name name)

{

**return** name && (**select** **firstOnly** RecId **from** Table12

**where** Table12.Name == name).RecId!=**0**;

}

**Write this code in Job(Exist Method):-**

**static** **void** Exist2(Args \_args)

{

Table12 tt;

**boolean** b;

b = Table12::exist("mm");

**if**(b)

{

info("Record Exist");

}

**else**

{

info("Not Exist");

}

}

Output:- Record Exist (mm is present in the table)

**HA Task 1:** print accountnumber in CustTable which are match with SalesTable CustAccount

static void Job1(Args \_args)

{

Query query = new Query();

QueryBuildDataSource qbds1;

QueryBuildDataSource qbds2;

QueryRun qr;

CustTable cstt;

SalesTable sltbl;

qbds1 = query.addDataSource(tableNum(CustTable));

qbds2 = qbds1.addDataSource(tableNum(SalesTable));

qbds2.joinMode(JoinMode::InnerJoin);

qbds2.addLink(fieldNum(CustTable, AccountNum),fieldNum(SalesTable, CustAccount));

qr = new QueryRun(query);

while(qr.next())

{

cstt = qr.get(tableNum(CustTable));

sltbl = qr.get(tableNum(SalesTable));

info(strFmt("%1",cstt.AccountNum));

}

}

**Output :** 9968 (coz there was only 1 record in custTable)

**HA Task 2:** join 3 tables without join and find common data from these 3 tables

static void Job1(Args \_args)

{

Query q;

QueryRun qr;

QueryBuildDataSource csttbds;

QueryBuildDataSource csttrnsds;

QueryBuildDataSource saltbds;

CustTable csttb;

CustTrans csttrns;

SalesTable saltb;

q = new Query();

csttbds = q.addDataSource(tableNum(CustTable));

csttrnsds = csttbds.addDataSource(tableNum(CustTrans));

csttrnsds.relations(false);

csttrnsds.joinMode(JoinMode::InnerJoin);

csttrnsds.addLink(fieldNum(CustTable, AccountNum), fieldNum(CustTrans, AccountNum));

saltbds = csttbds.addDataSource(tableNum(SalesTable));

saltbds.relations(false);

saltbds.joinMode(JoinMode::InnerJoin);

saltbds.addLink(fieldNum(CustTable, AccountNum),fieldNum(SalesTable, CustAccount));

qr = new QueryRun(q);

while(qr.next())

{

csttb = qr.get(tableNum(CustTable));

csttrns = qr.get(tableNum(CustTrans));

saltb = qr.get(tableNum(SalesTable));

info(strFmt("%1-%2-%3", csttb.AccountNum, csttrns.AccountNum, saltb.CustAccount));

}

}

**Output :** 888 c—(common data will show) (coz there was only 1 record in custTable)

**@ Use view to test a query:-**

**static** **void** Job1(Args \_args)

{

Query q;

QueryRun qr;

CustTable cstt;

CustTrans cstr;

qr = **new** QueryRun(**queryStr**(CustTable));

**if**(qr.prompt())

{

**while**(qr.next())

{

cstt = qr.get(**tableNum**(CustTable));

cstr = qr.get(**tableNum**(CustTrans));

info(cstt.AccountNum + "-" + **num2str**(cstr.amountCur,**0**,**2**,**1**,**2**));

}

}

}

Output: view table will prompt

**Task 3:** create new Customer through code

static void Job1(Args \_args)

{

CustTable custTable;

AccountNum accountNum = 'GBL-789';

CustGroupId custGroupId = '30';

Name name = 'salvator';

DirParty dirParty;

DirPartyPostalAddressView dirPartyPostalAddressView;

custTable = CustTable::find(accountNum);

if(!custTable)

{

ttsBegin;

custTable.clear();

custTable.initValue();

custTable.AccountNum = accountNum;

custTable.CustGroup = custGroupId;

custTable.insert(DirPartyType::Organization, name);

dirParty = DirParty::constructFromCommon(custTable);

dirPartyPostalAddressView.LocationName = 'GBL-auto';

dirPartyPostalAddressView.City = 'Germany';

dirPartyPostalAddressView.Street = 'jsnjks';

dirPartyPostalAddressView.StreetNumber = '456';

dirPartyPostalAddressView.CountryRegionId = 'rth';

dirParty.createOrUpdatePostalAddress(dirPartyPostalAddressView);

ttsCommit;

}

else

{

ttsBegin;

dirParty = DirParty::constructFromCommon(custTable);

dirPartyPostalAddressView.LocationName = 'GBL-auto';

dirPartyPostalAddressView.City = 'UK';

dirPartyPostalAddressView.Street = 'uk';

dirPartyPostalAddressView.StreetNumber = '345';

dirPartyPostalAddressView.CountryRegionId = 'abc';

dirPartyPostalAddressView.IsPrimary=3;

dirParty.createOrUpdatePostalAddress(dirPartyPostalAddressView);

ttsCommit;

}

info('Done!');

}

**Output:** (error)The number sequence for party records is not set

/ otherwise new customer will create – to check 🡪 goto acc.receivable all cust and search that customer name

static void SK\_CreateCust(Args \_args)

{

CustTable custTable;

AccountNum accountNum = 'XYZ-123';

CustGroupId custGroupId = '10';

Name name = 'DEF';

DirParty dirParty;

DirPartyPostalAddressView dirPartyPostalAddressView;

custTable = CustTable::find(accountNum);

if(!custTable) */\*if customer not exits\*/*

{

ttsBegin;

custTable.clear();

custTable.initValue();

custTable.AccountNum = accountNum;

custTable.CustGroup = custGroupId;

custTable.insert(DirPartyType::Organization, name);

dirParty = DirParty::constructFromCommon(custTable);

dirPartyPostalAddressView.LocationName = "XYZ-AUTO";

dirPartyPostalAddressView.City = "Washingtan";

dirPartyPostalAddressView.Street = "salvator Street";

dirPartyPostalAddressView.StreetNumber = "123";

dirPartyPostalAddressView.CountryRegionId = "SBR";

dirParty.createOrUpdatePostalAddress(dirPartyPostalAddressView);

ttsCommit;

}

else */\* if customer exits updates the address of the customer\*/*

{

ttsBegin;

dirParty = DirParty::constructFromCommon(custTable);

dirPartyPostalAddressView.LocationName = "XYZ-AUTO'";

dirPartyPostalAddressView.City = "Japan";

dirPartyPostalAddressView.Street = "Gemini Street";

dirPartyPostalAddressView.StreetNumber = "123";

dirPartyPostalAddressView.CountryRegionId = "SBR";

dirPartyPostalAddressView.IsPrimary=1;

dirParty.createOrUpdatePostalAddress(dirPartyPostalAddressView);

ttsCommit;

}

info('Done!');

}

Output: done..

**Select Statements**:-

1. **Count**:-how many records

**static** **void** Count22(Args \_args)

{

Stu\_Table student;

**int** a = **0**;

**while** **select** **count**(RecId) **from** student

{

a = student.RecId;

}

info(**strFmt**("%1", a));

}

Output: 5

1. **Select all:-**

**static** **void** Selectstatement22(Args \_args)

{

Stu\_Table student;

;

**while** **select** \* **from** student

info(**strFmt**("%1----%2", student.Name, student.RollNumber));

}

Output: all data will be displayed

\*\* **Select only 1 record from the table**

**Static** **Void** Con\_CrossCompany22(Args \_args)

{

Stu\_Table student;

**select** \* **FROM** student;

info(**strFmt**("%1",student.RollNumber));

}

1. **Arithmetic operation**: add sub mul div

**static** **void** ArithmeticJob22(Args \_args)

{

ArithmeticOperations A1 = **new** ArithmeticOperations (); *// ‘new’ method is inherited from the Object class.*

A1.addition1();

A1.Subtarction1();

A1.Multiplication1();

A1.Division1();

}

1. **Order by:-**

**static** **void** Job6(Args \_args)

{

Stu\_Table student;

**while** **select** student

**order** **by** student.RollNumber **desc**

info(**strFmt**("%1, %2", student.Name,student.RollNumber));

}

Output: stuti, 5

Meera, 1

1. **Group by**:-

**static** **void** Groupby(Args \_args)

{

Stu\_Table student;

**while** **select** student

**group** **by** student.Name

**where** student.Name == "meera"

info(**strFmt**("%1, %2", student.Name,student.RollNumber));

}

Output: meera, 0

1. **Where clause**:-

**static** **void** Where22(Args \_args)

{

Stu\_Table student;

*//SalesTable SalesTable;*

**while** **select** \* **from** student

**where** student.Name == "meera"

info(**strFmt**("%1", student.Name));

}

Output:- Meera

Meera

1. **AND OR NOT**: && , || ,

**static** **void** Relationoperation(Args \_args)

{

Stu\_Table student;

**while** **select** student **where** student.Name == "meera" || student.Name == "stuti”

{

info(**strFmt**("%1", student.Name));

}

}

OUTPUT :- Meera, Meera

1. **Forupdate**:-

**static** **void** forupdate1(Args \_args)

{

Stu\_Table student;

**ttsBegin**;

**while** **select** **forUpdate** student **where** student.Name == "meera"

{

student.Name = "meera joe";

student.update();

*//info(strFmt("%1", student.Name));*

}

**ttsCommit**;

}

Output:- “meera” change to “meera joe”

1. **FirstFast**:- The first row appears more quickly but the total return time for this option might be slower

**Static** **Void** Firstfast22(Args \_args)

{

Stu\_Table student;

*//While select firstFast student <--- (we can write like this also) (Firstonly)*

**While** **select** **firstFast** \* **from** student

{

info(student.Name);

}

}

Output:- all data will be displayed

1. **Cross Company**:- Returns data for all companies that the user is authorized to read from
2. **……………….**

**Static** **Void** Con\_CrossCompany22(Args \_args)

{

Stu\_Table student;

**Container** Companies=['IBM','HCL'];

**while** **select** **CrossCompany**:Companies \* **from** student // with crosscompany

{

info(**strFmt**("%1",student.RollNumber));

}

}

Output:- it will get both companies data

**B) ……………**

**Static** **Void** Con\_CrossCompany22(Args \_args)

{

Stu\_Table student;

*//Container Companies=['USMF','BRMF']; change to two companies*

**while** **select** \* **from** student // Without crosscompany

{

info(**strFmt**("%1",student.RollNumber));

}

}

Output:- it will get only 1 companies data

**c) ……………**

**Static** **Void** Con\_CrossCompany22(Args \_args)

{

Stu\_Table student;

**select** \* **FROM** student;

info(**strFmt**("%1",student.Name));

}

Output: it will select only 1 record (not whole data)

1. **Change company**:-

**static** **void** changeComp(Args \_args)

{

Stu\_Table student;

info(student.dataAreaId);

**changeCompany**('USMF')

{

student = **null**;

**while** **select** student

{

info(**strFmt**("%1", student.dataAreaId));

}

}

}

Output:- it will change company to usmf (usmf, usmf)

1. **Firstonly**:-

**Static** **Void** Firstonly22(Args \_args)

{

Stu\_Table student;

**while** **select** **firstOnly** student

{

info(**strFmt**("%1",student.Name));

}

}

Output:- meera trib ( only first record)

1. **Joins**:-
2. **Inner-join**:-

**Static** **Void** join22(Args \_args)

{

Stu\_Table student;

DetalsTable dtable;

**while** **select** student **join** dtable **where** student.RollNumber==dtable.RollNumber

{

info(**strFmt**("%1.%2",student.Name, student.RollNumber));

}

}

Output:- common record will print (1,2,3 records)

1. **Outer-join**:-

**Static** **Void** join22(Args \_args)

{

Stu\_Table student;

DetalsTable dtable;

**while** **select** student **outer** **join** dtable **where** student.RollNumber==dtable.RollNumber

{

info(**strFmt**("%1.%2",student.Name, student.RollNumber));

}

}

Output:- all record will print (1,2,3,4,5 records)

1. **Exists**:-

**Static** **Void** join22(Args \_args)

{

Stu\_Table student;

DetalsTable dtable;

**while** **select** student **exists** **join** dtable **where** student.RollNumber==dtable.RollNumber

{

info(**strFmt**("%1.%2",student.Name, student.RollNumber));

}

}

Output:- will print 3 records (coz that is exists in the both tables)

1. **Not- Exists**:-

**Static** **Void** join22(Args \_args)

{

Stu\_Table student;

DetalsTable dtable;

**while** **select** Name, RollNumber **from** student **order** **by** RollNumber **notExists** **join** \* **from** dtable **where**(student.RollNumber==dtable.RollNumber)

{

info(**strFmt**("%1.%2",student.Name, student.RollNumber));

}

}

Output:- It will print only 2 records(which is not exists in both the tables)

1. **Dynamic Query for 4 tables**:-

**static** **void** DynamicQuery22(Args \_args)

{

Query q;

QueryBuildDataSource qbds;

QueryBuildDataSource tbl2;

QueryBuildDataSource tbl3;

QueryBuildDataSource tbl4;

QueryBuildRange qbr;

QueryRun qr;

Table11 t11;

Table22 t12;

Table33 t13;

Table44 t44;

q = **new** Query();

qbds = q.addDataSource(**tableNum**(Table11));

tbl2 = qbds.addDataSource(**tableNum**(Table22));

tbl3 = tbl2.addDataSource(**tableNum**(Table33));

tbl4 = qbds.addDataSource(**tableNum**(Table44));

qbr = qbds.addRange(**fieldNum**(Table11,Name));

qbr.Value();

qr = **new** QueryRun(q);

**while**(qr.next())

{

t11 = qr.get(**tableNum**(Table11));

info(**strFmt**("%1,%2",t11.Name,t11.Id));

}

}

Output:- siya,2

Siya,2

Siya,2

**Task:-**

1. **Join 4 tables through relations**:- 1-2, 2-3, 3-4

**static** **void** join\_Tble(Args \_args)

{

query q;

QueryBuildDataSource qbds, tb12, tb13, tb14;

QueryBuildRange qbr;

QueryBuildlink qbl;

QueryRun qr;

Table11 t11;

Table22 t22;

Table33 t33;

Table44 t44;

q = **new** Query();

qbds = q.addDataSource(**tableNum**(Table22)); // (1 table - 2 table)

tb12 = qbds.addDataSource(**tableNum**(Table11));

tb12.relations(**false**);

tb12.joinMode(JoinMode::InnerJoin);

tb12.addLink(**fieldNum**(Table22, Id),**fieldNum**(Table11, Id));

qbds = q.addDataSource(**tableNum**(Table33));

tb13 = qbds.addDataSource(**tableNum**(Table22)); // (2 table – 3 table)

tb13.relations(**false**);

tb13.joinMode(JoinMode::InnerJoin);

tb13.addLink(**fieldNum**(Table22, Id),**fieldNum**(Table11, Id));

qbds = q.addDataSource(**tableNum**(Table44));

tb14 = qbds.addDataSource(**tableNum**(Table33)); // (3 table – 4 table)

tb14.relations(**false**);

tb14.joinMode(JoinMode::InnerJoin);

tb14.addLink(**fieldNum**(Table22, Id),**fieldNum**(Table11, Id));

qr = **new** QueryRun(q);

**while**(qr.next())

{

t22 = qr.get(**tableNum**(Table22));

t11 = qr.get(**tableNum**(Table11));

t33 = qr.get(**tableNum**(Table33));

t44 = qr.get(**tableNum**(Table44));

info(**strFmt**("%1-%2-%3-%4",t22.Id, t11.Id, t33.Id, t44.Id));

}

}

Output:- common data from all tables 1-1-0-0 2-2-0-0 0-0-0-1 0-0-0-2

1. **Reverse order**:- same like desc order by

**static** **void** Job10(Args \_args)

{

Stu\_Table student;

**while** **select** **reverse** student

**order** **by** student.RollNumber

info(**strFmt**("%1",student.RollNumber));

}

Output:- 105,104,103,102,101

**Task 3a**:

**Passing arguments from(1 form to another form)**:-

**void** clicked() // method on forms button

{

Args args;

FormRun formRun;

**super**();

args = **new** Args();

args.record(Stu\_Table);

args.parmEnum(Stu\_Table.Id);

args.name(**formstr**(TaskForm1));

formRun = ClassFactory.formRunClass(args);

formRun.init();

formRun.run();

formRun.wait();

}

**public** **void** init() // method on second form (override)

{

Query query;

QueryBuildRange qbr;

Stu\_Table student;

**str** parmid;

;

**super**();

student = element.args().record();

parmid = element.args().parm();

query = **new** Query();

qbr = query.addDataSource(**tablenum**(DetalsTable)).addRange(**fieldnum**(DetalsTable,Id));

qbr.value(SysQuery::value(student.Id));

DetalsTable\_ds.query(query);

}

Output:- if we click on 1st form data it will shows same data on 2nd form

**Task 3b**:

**Passing arguments from (form to class)**:-

**Void** clicked () // method on forms button

{

Args args;

MenuFunction menuFunction;

**super**();

args = **new** Args();

args.record(Table44);

args.parmEnum(Table44.Id);

menuFunction = **new** MenuFunction(**identifierStr**(MenuItem2), MenuItemType::Action);

menuFunction.run(args);

}

**public** **static** **void** main(Args \_args) // method on class

{

Table44 t44;

ClassForm clsform;

**int** Id;

;

clsform = **new** ClassForm();

Id = \_args.parmEnum();

t44 = \_args.record();

info(**strfmt**("%1",t44.Id));

}

Output:- if we click data on 1st form it will get same info on message form

**Task 3c**:

**Passing arguments from (class to form)**:-

**public** **static** **void** main(Args \_args) // method on the class

{

FormRun formRun;

**container** con;

**anytype** value1;

\_args = **new** Args();

con = ["no\_001","vihan","4563873456"];

value1 = con2Str(con);

\_args.parm(value1);

\_args.Name(**formstr**(class\_to\_Form88));

formRun = classFactory.formRunClass(\_args);

formRun.init();

formRun.run();

formRun.wait();

}

**public** **void** init() // method on form(override)

{

Cust\_Table tbl88;

**str** value1, value2, value3, value4, value5;

**int** i;

**container** con;

**super**();

value1 = element.args().parm();

con = str2con(value1);

**for**(i=**0**; i<=**conLen**(con);i++)

{

value2 = **conPeek**(con,i);

info(**strFmt**("%1", value2));

**if**(i == **1**)

{

value3 = value2;

}

**if**(i == **2**)

{

value4 = value2;

}

**if**(i == **3**)

{

value5 = value2;

}

}

tbl88.Id = value3;

tbl88.Name = value4;

tbl88.phone = value5;

tbl88.insert();

}

Output:- data will be add on the table(no\_001, vihan, 4563873456)

\*) **Lookup through code**:- EDT, Display method

**public** **void** lookup()

{

Query query = **new** Query();

QueryBuildDataSource queryBuildDataSource;

SysTableLookup sysTableLookup;

queryBuildDataSource = query.addDataSource(**tableNum**(CustTable));

sysTableLookup = SysTableLookup::newParameters(**tableNum**(CustTable), this);

sysTableLookup.addLookupField(**fieldNum**(CustTable,AccountNum));

sysTableLookup.parmQuery(query);

sysTableLookup.performFormLookup();

}

**Lookup through display method:- not exists**

1. Write display method in table(custtable) , on form drag custtbl , create new string Edit- here give display method name ,lebel,

**display** CustName myCustName()

{

CustTable custtable;

**select** AccountNum **from** custtable **where** custtable.RecId == this.RecId;

**return** custtable.AccountNum;

}

1. Write lookup code in string Edits override (method)

**public** **void** lookup()

{

Query q = **new** Query();

QueryBuildDataSource qbds;

SysTableLookup sysTableLookup;

**super**();

sysTableLookup = SysTableLookup::newParameters(**tableNum**(CustTable),this);

sysTableLookup.addLookupfield(**fieldNum**(CustTable,AccountNum),**true**);

sysTableLookup.addLookupMethod(**tableMethodStr**(CustTable,myCustName));

qbds = q.addDataSource(**tableNum**(CustTable));

sysTableLookup.parmQuery(q);

sysTableLookup.performFormLookup();

}

Output:- u will get lookup on AccountNum’s field (dropdown Box) if u don’t give datasource – u can select

**Edit method**:-

**Edit** CustAccount editNames(**boolean** set,CustAccount names)

{

CustAccount name=names;

CustTable custdetails;

**if**(set)

{

**if**(name)

{

**ttsBegin**;

**select** **forupdate** AccountNum **from** custdetails

**where** custdetails.AccountNum==this.AccountNum;

custdetails.AccountNum=names;

custdetails.update();

**ttsCommit**;

}

}

**else**

{

**select** AccountNum **from** custdetails **where** custdetails.AccountNum==this.AccountNum;

names=custdetails.AccountNum;

}

**return** names;

}

**Task 2 :- when you select record from the lookup and click ok button the selected record should appear in the below table**

* Take 4 tbls (1 custom table created by us with fields [salesId, custAccount, CustNmae] and 3 tbls drag from AOT salestbl,custtable,dirpartytable)
* Take 1 from drag custom table in , Grid take String Edits for salesId custaccount, Name (give datasource , field)
* Create another string Edit outside the grid (give- name, label, ExtendedDataType) autodeclrton yes
* Create button (ok) write click method

**void** clicked()

{

SalesTable st;

CustTable ct;

DirPartyTable dpt;

DAX\_StudTable \_StudTable;

**ttsBegin**;

\_StudTable.clear();

**select** \* **from** ct **join** st

**where** ct.AccountNum == st.custaccount && ct.AccountNum == CustomAccount.valuestr();

\_StudTable.SalesId = st.salesId;

\_StudTable.accountnum = CustomAccount.valuestr();

**select** \* **from** ct **join** dpt **where** ct.AccountNum == CustomAccount.valuestr()

&& dpt.RecId == ct.party;

\_StudTable.name = dpt.Name;

\_StudTable.insert();

**ttsCommit**;

DAX\_StudTable\_ds.refresh(); // for refresh the table

DAX\_StudTable\_ds.research(**true**);

}

**Task 3:- if we select 1 record cancel button should be disable**

\*) create listpage — query(drag table, change dynamic query-yes), class(extends ListPageInteraction and override selectionChanged method), form(add query)

\*) create menuitem button (Cancel) give same name in text

\*) select interaction class (AListPageInteraction)

**public** **class** AListPageInteraction **extends** ListPageInteraction

{

}

**Public** **void** selectionChanged()

{

*// Emp\_Table emp\_Table;*

Emp\_Table emp\_Table = this.listPage().activeRecord(**queryDataSourceStr**(RootQuery3, Emp\_Table));

**super**();

**if**(emp\_Table.Emp\_Id > **3**)

{

this.listPage().actionPaneControlEnabled(**formControlStr**(DAX\_ListPage, Cancel), **false**);

}

**else**

{

this.listPage().actionPaneControlEnabled(**formControlStr**(DAX\_ListPage, Cancel), **true**);

}

}

Output:- if we select record 4,5,6 button will be disable but for 1,2,3 record button will be enable

**Task1:- Form-to-form**

\*) create 2 tables(StudTable, StudDetails),2 forms(from11,form22)

while creating fields take --**string type id**--

\*) create relation in child table(StudDetails)

\*) in form11 on button write click method/ form22 in ds override init method

**void** clicked()

{

Args args;

FormRun formRun;

**super**();

args = **new** Args();

args.record(StudTable);

args.parmEnum(StudTable.Stud\_Id);

args.name(**formStr**(DAX\_Form22));

formRun = ClassFactory.formRunClass(args);

formRun.init();

formRun.run();

formRun.wait();

}

**public** **void** init()

{

Query query;

QueryBuildRange qbr;

StudTable studTable;

**str** permid;

;

**super**();

studTable = element.args().record();

permid = element.args().parm();

query = **new** Query();

qbr = query.addDataSource(**tableNum**(StudDetails)).addRange(**fieldnum**(StudDetails,Stud\_Id));

*//qbr.value(SysQuery::value(StudDetails.Stud\_Id));*

qbr.value(permid);

StudDetails\_Ds.query(query);

}

**Task2 :- Form\_to\_form (select multiple records)**

\*) create 1 tbale (StudTable) field- string-one(Stud\_Id)

\*) create 2 forms in 1- write clicked method in 2- write init method

\*) button property multiselect- yes

**void** clicked()

{

**int** recordsCount;

StudTable \_studTable;

**container** con;

Args args;

**str** multiSelectString;

;

args = **new** Args();

recordsCount = StudTable\_ds.recordsMarked().lastIndex(); *// gets the total records selected*

\_studTable = StudTable\_ds.getFirst(**1**);

**while** (\_studTable)

{

*// storing recid of selected field in container*

con = **conIns**(con,**1**,\_studTable.RecId);

*// converting container to string with comma separated*

multiSelectString = con2Str(con,",");

\_studTable = StudTable\_ds.getNext(); *// moves to next record*

}

*// passing string*

args.parm(multiSelectString);

*// calling menu item*

**new** MenuFunction(**menuitemDisplayStr**(MenuItem3), MenuItemType::Display).run(args);

}

**public** **void** init()

{

**container** con;

**int** i;

**str** multipleRecords;

**super**();

*// getting string value from caller*

multipleRecords = element.args().parm();

*// string to container*

con = str2con(multipleRecords,",");

*// for sorting*

**for**(i = **1**;i<= **conLen**(con) ;i++)

{

StudTable\_ds.query().dataSourceTable(**Tablenum**(StudTable)).addRange(**fieldNum**(StudTable,RecId)).value(SysQuery::value(**conPeek**(con,i)));

}

}

Output:- select multi record same will display on second form

**Args Types:-**

Caller – used to get/set info(value) for object i.e form

Name - used to pass form name

Record – to pass table

Parm - to pass string values

ParmEnum – to pass enumtype methods

ParmEnumType – to pass enum values

ParmObject – used to pass instance of obj. (buffer table)

/ to get the value from buffertable

**Task3:- Form-to-class**

\*) create 1 table, 1 form(button-click method), class(write main method),

menuitem(in action mode)give objType-class obj-class Name

**Void** clicked ()

{

Args args;

MenuFunction menuFunction;

**super**();

args = **new** Args();

args.record(DAX\_CstTable1);

args.parm(DAX\_CstTable1.Cust\_Id);

menuFunction = **new** MenuFunction(**identifierStr**(MenuItem2), MenuItemType::Action);

menuFunction.run(args);

**super**();

}

**public** **static** **void** main(Args \_args)

{

DAX\_CstTable1 dAX\_CstTable1;

ClassArgs classArgs;

**str** parmid;

;

classArgs = **new** ClassArgs();

parmid = \_args.parm();

dAX\_CstTable1 = \_args.record();

info(parmid);

}

**Task4:- Form-To-Class (Multiple select)**

\*) create table, Form, class(ClassArgs11), menuitem (menuitem3)

\*) give objType-class obj-class Name button multipleselect property-yes

**void** clicked()

{

**int** recordsCount;

DAX\_CstTable1 \_dAX\_CstTable1;

**container** con;

Args args;

**str** multiSelectString;

;

args = **new** Args();

recordsCount = DAX\_CstTable1\_ds.recordsMarked().lastIndex(); *// gets the total records selected*

\_dAX\_CstTable1 = DAX\_CstTable1\_ds.getFirst(**1**);

**while** (\_dAX\_CstTable1)

{

*// storing recid of selected field in container*

con = **conIns**(con,**1**,\_dAX\_CstTable1.RecId);

*// converting container to string with comma separated*

multiSelectString = con2Str(con,", ");

\_dAX\_CstTable1 = DAX\_CstTable1\_ds.getNext(); *// moves to next record*

}

*// passing string*

args.parm(multiSelectString);

*// calling menu item*

**new** MenuFunction(**identifierstr**(MenuItem3), MenuItemType::Action).run(args);

}

**public** **static** **void** main(Args args)

{

DAX\_CstTable1 \_dAX\_CstTable1;

**container** con;

**int** i;

**str** multipleRecords;

**anytype** values;

;

multipleRecords = args.parm();

con = str2con(multipleRecords);

**for**(i=**1**; i<=**conLen**(con); i++)

{

values = **conPeek**(con,i);

**select** \_dAX\_CstTable1 **where** \_dAX\_CstTable1.RecId == values;

**if**(\_dAX\_CstTable1.RecId)

{

info(**strFmt**("%1 %2",\_dAX\_CstTable1.Cust\_Id, \_dAX\_CstTable1.Name));

}

}

*// i am just here an infolog of recids but you can do as per requirement once you got multiple records*

}

**Task5:- Class-To-Form**

\*) create table (DAX\_CustomerTable), Form(init method), Class(main method)

\*) Don’t give data in table It will inserted by this (class\_To\_form)

**public** **static** **void** main(Args \_args)

{

FormRun formRun;

**container** con;

**anytype** value1;

\_args = **new** Args();

con = ["d\_01","Riya"];

value1 = con2Str(con);

\_args.parm(value1);

\_args.name(**formStr**(DAX\_Class\_To\_FormAA));

formRun = classFactory.formRunClass(\_args);

formRun.init();

formRun.run();

formRun.wait();

}

**public** **void** init()

{

DAX\_CustomerTable \_dAX\_CustomerTable;

**str** value1, value2, value3, value4;

**int** i;

**container** con;

**super**();

value1 = element.args().parm();

con = str2con(value1);

**for**(i=**1**; i<=**conLen**(con); i++)

{

value2 = **conPeek**(con,i);

info(**strFmt**("%1", value2));

**if**(i == **1**)

{

value3 = value2;

}

**if**(i == **2**)

{

value4 = value2;

}

}

\_dAX\_CustomerTable.Cust\_Id = value3;

\_dAX\_CustomerTable.Name = value4;

\_dAX\_CustomerTable.insert();

}

Output:- ["d\_01","Riya"] will add on form

**Task6:- Class-To-Form (Multiple select)**

----

---

**Task7:- Class\_To\_Class**

-> Create 2 classes.(ClassA, ClassB)

-> Create menuitem for 2nd Class. (MenuItem4)

-> Write Main method in 2 classes.

**ClassA** :



**ClassB** :



**Main method of 1st class** :

**public** **static** **void** main(Args args)

{

MenuFunction menuFunction;

**str** dept = '1000';

args = **new** Args();

args.parm(dept);

menuFunction = **new** MenuFunction(**identifierStr**(MenuItem4), MenuItemType::Action);

menuFunction.run(args);

}

**Main method of 2nd class** :

**public** **static** **void** main(Args args)

{

**str** deptid;

deptid = args.parm();

info(deptid);

}

**Containers:-**

1. **Conpeek:-**

**static** void ConpeekJob1(Args \_args)

{

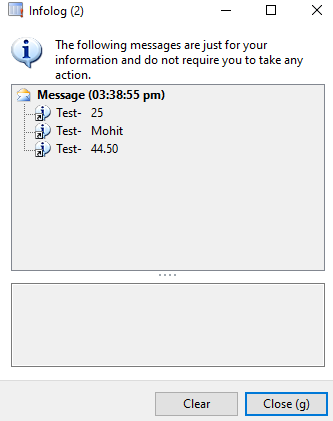
**container** con = ["Test-",**25**,"Mohit",**44.500**];

info(**strFmt**("%1 %2", **Conpeek**(con, **1**), **conPeek**(con, **2**)));

info(**strFmt**("%1 %2", **Conpeek**(con, **1**), **conPeek**(con, **3**)));

info(**strFmt**("%1 %2", **Conpeek**(con, **1**), **conPeek**(con, **4**)));

}



1. **Conpoke:-**

**static** **void** ConpokeJob1(Args \_args)

{

**container** con = ["Test-",**25**,"Mohit",**44.500**];

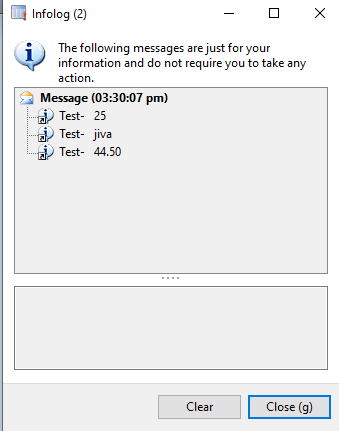
con = **conPoke**(con, **3**,"jiva");

info(**strFmt**("%1 %2", **Conpeek**(con, **1**), **conPeek**(con, **2**)));

info(**strFmt**("%1 %2", **Conpeek**(con, **1**), **conPeek**(con, **3**)));

info(**strFmt**("%1 %2", **Conpeek**(con, **1**), **conPeek**(con, **4**)));

}



1. **ConIns:-**

**static** **void** ConInsJob1(Args \_args)

{

**container** con = ["Test-",**25**,"Mohit",**44.500**];

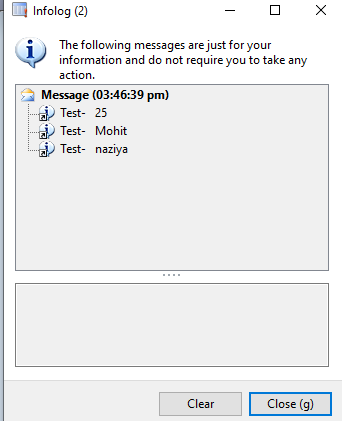
con = **conIns**(con,**4**,"naziya");

info(**strFmt**("%1 %2", **Conpeek**(con, **1**), **conPeek**(con, **2**)));

info(**strFmt**("%1 %2", **Conpeek**(con, **1**), **conPeek**(con, **3**)));

info(**strFmt**("%1 %2", **Conpeek**(con, **1**), **conPeek**(con, **4**)));

}



**4)ConFind:-**

**static** **void** ConFindJob1(Args \_args)

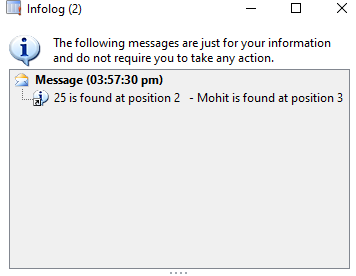
{

**container** con = ["Test-",**25**,"Mohit",**44.500**];

info(**strFmt**("25 is found at position %1 - Mohit is found at position %2",

**conFind**(con,**25**),**conFind**(con,"Mohit")));

}



**5)ConLen:-**

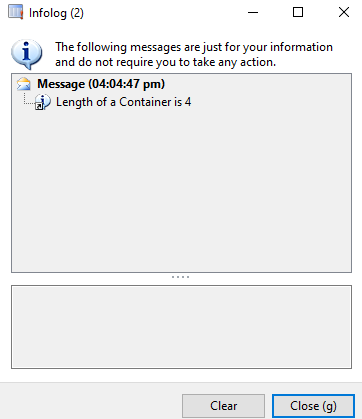
**static** **void** ConLenJob1(Args \_args)

{

**container** con = ["Test-",**25**,"Mohit",**44.500**];

info(**strFmt**("Length of a Container is %1",**conLen**(con)));

}



**6) ConDel:-**

**static** **void** ConDelJob1(Args \_args)

{

**container** con = ["Test-",**25**,"Mohit",**44.500**];

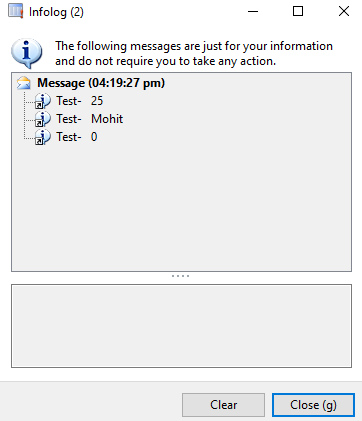
con = **conDel**(con,**4**,**1**);

info(**strFmt**("%1 %2", **Conpeek**(con, **1**), **conPeek**(con, **2**)));

info(**strFmt**("%1 %2", **Conpeek**(con, **1**), **conPeek**(con, **3**)));

info(**strFmt**("%1 %2", **Conpeek**(con, **1**), **conPeek**(con, **4**)));

}



**7) Connull:-**

**static** **void** ConNullJob1(Args \_args)

{

**container** con = ["Test-",**25**,"Mohit",**44.500**];

info(**strFmt**("Container conNull() after : %1 - %2 - %3 - %4",

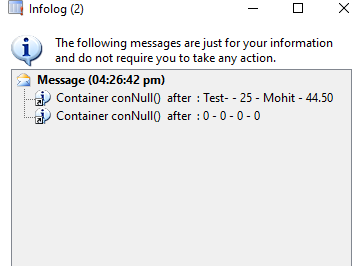
**conPeek**(con,**1**),**conPeek**(con,**2**),**conPeek**(con,**3**),**conPeek**(con,**4**)));

con = **conNull**();

info(**strFmt**("Container conNull() after : %1 - %2 - %3 - %4",

**conPeek**(con,**1**),**conPeek**(con,**2**),**conPeek**(con,**3**),**conPeek**(con,**4**)));

}



**\*) Collection classes**:- collection classes have been designed for storing objects

Collection classesTypes:- **Set , List, map, Struct, (Array) -------** write on jobs

### **Set**

A set is an unordered collection that contains elements of all the same type

A **Set**is used for the storage and retrieval of data from a collection i.e :- duplicates values are not allowed here

**static** **void** CollectionClass\_Set()

{

Set cars = **new** Set(Types::String); *// Create a new set of type String*

SetEnumerator setE;

cars.add("Toyota"); *// Add elements to the set*

cars.add("Ford");

cars.add("Mazda");

cars.add("Toyota");

**if** (cars.in("Toyota")) *// Check to see if an element*

*// exist in the set*

info ("Toyota is part of the set");

info (cars.toString()); *// Display the content of the set*

setE = cars.getEnumerator(); *// Get the enumerator of the set*

*// to loop through it*

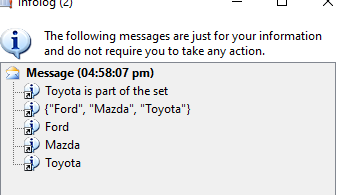
**while** (setE.moveNext())

{

info(setE.current());

}

}



### **List**

A list stores objects of all the same type in a sequential order

with ability to add values at the beginning or end of the list

**static** **void** CollectionClass\_List()

{

List L = **new** List(Types::Integer); *// Create a new list of type string*

ListEnumerator listE;

L.addEnd(**22**); *// Add elements to the list*

L.addEnd(**33**);

L.addStart(**11**); *// Start position*

*//L.addStart(11); // we can add duplicate value*

info (L.toString()); *// Display the content of the list*

listE = L.getEnumerator(); *// Get the enumerator of the list*

*// to loop through it*

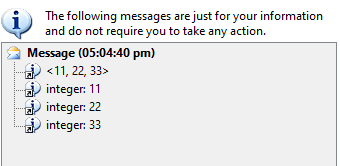
**while** (listE.moveNext())

{

info (**strfmt**("integer: %1", listE.current()));

}

}



### **Map**

A map stores key-value pairs. For each key, there is a corresponding value

The keys are unique and are all of the same type.

**static** **void** CollectionClass\_Map()

{

*// Create a new map with a key and value type*

Map brands = **new** Map(Types::Integer, Types::String);

MapEnumerator mapE;

brands.insert (**1**, "GVk"); *// Insert values to the map*

brands.insert (**2**, "BHV");

brands.insert (**3**, "SRI");

*//brands.insert (3, "SRI"); // We cannot add duplicate values*

info (brands.toString()); *// Display the content of the map*

mapE = brands.getEnumerator(); *// Get the enumerator to loop*

*// through the elements of the map*

**while** (mapE.moveNext())

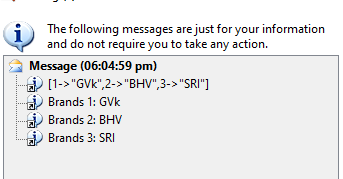
{

info(**strfmt**("Brands %1: %2", mapE.currentKey(),

mapE.currentValue()));

}

}



### **Struct**

A [**struct**](http://msdn.microsoft.com/en-us/library/struct.aspx) groups information into a single entity. Any of the types of the Types enum can be stored which includes containers, dates

**static** **void** CollectionClass\_Struct()

{

struct myCar = **new** struct ("int ModelYear; str Carbrand"); *// Create a struct with two fields*

**int** i;

myCar.value("ModelYear", **2015**); *// Set values to the fields*

myCar.value("Carbrand", "BMW");

*// Add a new field and give it a value*

myCar.add("Model", "320");

*// Loop through the fields of the struct*

**for** (i=**1**; i<=myCar.fields(); i++)

{

info(**strfmt**("FieldType: %1, FieldName: %2, Value: %3",

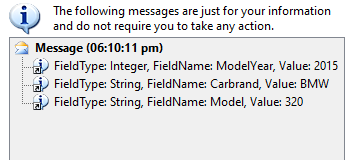
myCar.fieldType(i),

myCar.fieldName(i),

myCar.value(myCar.fieldName(i))));

}

}



* **Array**

It is accepting the single data type only it is not a collection type

you **cannot** **store** objects in **arrays** or **containers**

**static** **void** CollectionClass\_Array(Args \_args)

{

**int** i;

Array a1 = **new** Array(Types::Integer);

;

a1.value(**1**,**30**);

a1.value(**2**,**50**);

a1.value(**3**,**10**);

a1.value(**4**,**60**);

a1.value(**5**,**40**);

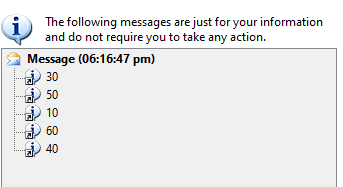
**for**(i = **1**; i<= a1.lastIndex(); i++)

{

info(**strFmt**("%1", a1.value(i)));

}

}



**\*) Creating sales order line through code:-** in job write this code

**static** **void** CreateSalesOrder(Args \_args)

{

SalesTable salesTable; *//declaring variables*

SalesLine salesLine;

NumberSeq numberSeq;

SalesFormLetter salesFormLetter;;

**ttsBegin**;

numberSeq = NumberSeq::newGetNum(SalesParameters::numRefSalesId()); *//creating sales order header*

numberSeq.used(); *//getting sales order id from number sequence*

salesTable.SalesId = numberSeq.num();

salesTable.initValue();

salesTable.CustAccount = "US-001";

salesTable.initFromCustTable();

**if** (!salesTable.validateWrite()) *//validate*

{

**throw** Exception::Error;

}

salesTable.insert();

salesLine.SalesId = salesTable.SalesId; *//creating sales order line*

salesLine.ItemId = "D0001";

salesLine.SalesQty = **2**;

salesLine.LinePercent = **1**;

salesLine.createLine(**true**, *// Validate*

**true**, *// initFromSalesTable*

**true**, *// initFromInventTable*

**true**, *// calcInventQty*

**true**, *// searchMarkup*

**true** *// searchPrice*

);

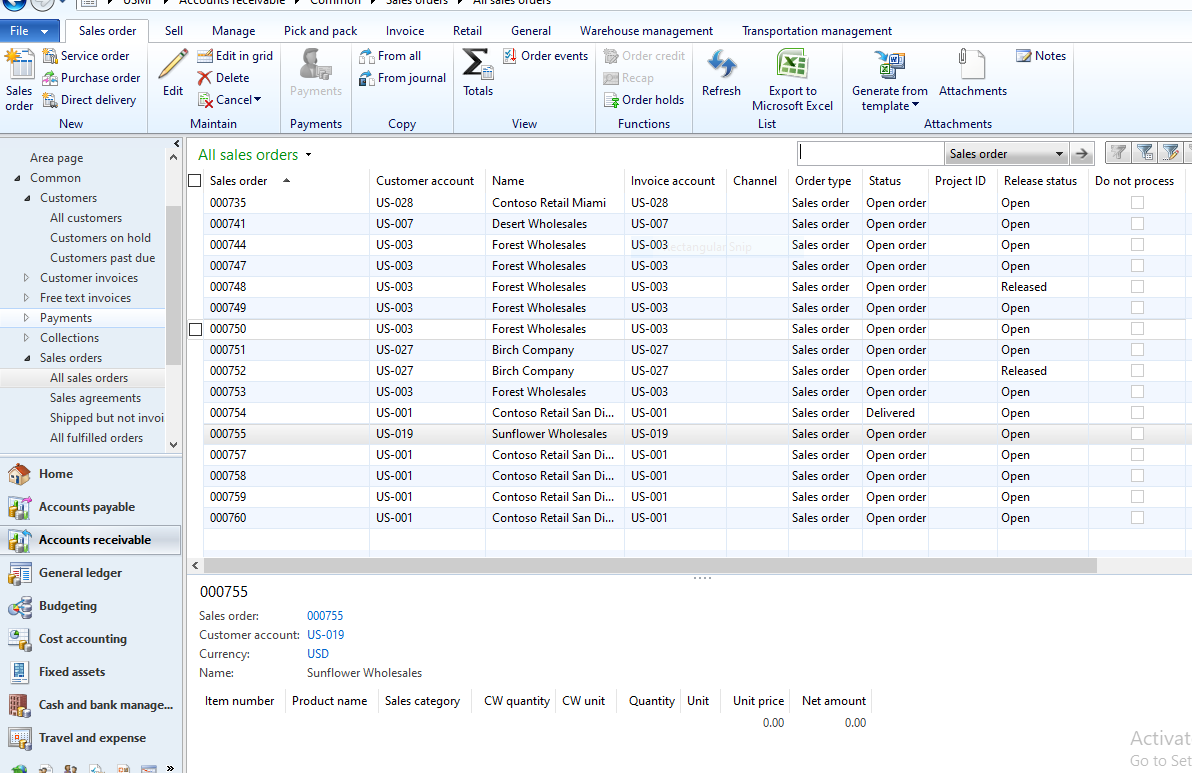
**ttsCommit**;

salesFormLetter = SalesFormLetter::construct(DocumentStatus::Confirmation); *//confirm sales order*

salesFormLetter.update(salesTable);

info(salesTable.SalesId); *//displaying sales order id*

}



### Creating a simple batch operation

\*) Create 2 classes :- **Controller Class, Service Class**

MenuItem (Action):- with controller class

[Batch Processing in Dynamics AX 2012 - SysOperation *Framework* in AX 2012 | Stoneridge Software](https://stoneridgesoftware.com/batch-processing-in-dynamics-ax-2012-using-sysoperation-framework/)

**public** **class** SysOperationControllerClass **extends** SysOperationServiceController

{

}

New method:-

**public** ClassDescription caption()

{

**return** "Batch Operation Task Description";

}

New method:-

**public** **void** **new**()

{

**super**();

this.parmClassName(**classStr**(SysOperationServiceClass));

this.parmMethodName(**methodStr**(SysOperationServiceClass, processOperation));

this.parmDialogCaption("Batch Operation Dialog Title");

}

New method:-

**public** **static** **void** main(Args args)

{

SysOperationControllerClass controller;

controller = **new** SysOperationControllerClass();

controller.startOperation();

}

Class declaration:-

**public** **class** SysOperationServiceClass **extends** SysOperationServiceBase

{

}

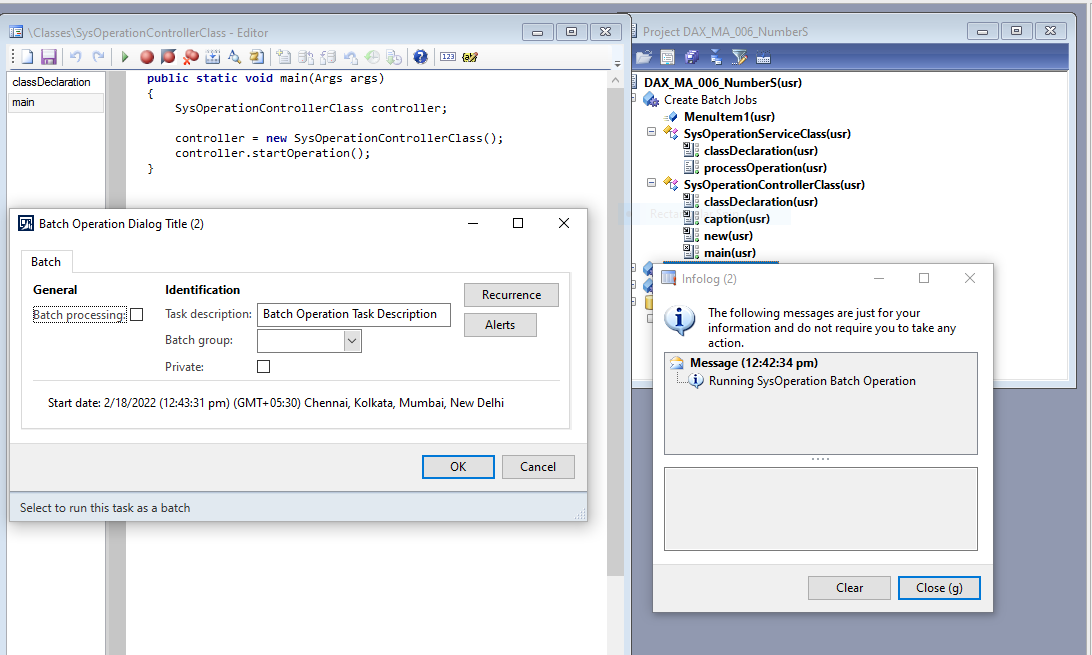
New method:-

**public** **void** processOperation()

{

info ("Running SysOperation Batch Operation");

}



### Add user defined parameters for batch operation

\*) Create 3 classes :- **Controller Class,** **Service Class, Data Contract Class**

\*) Menu Item :- with Controller class

-----In class declaration:-

[DataContractAttribute]

**public** **class** SysOperationDataContractClass

{

TransDate dateValue;

Description255 textValue;

}

----New method:-

[DataMemberAttribute,

SysOperationLabelAttribute('Date Property'),

SysOperationHelpTextAttribute('Enter a date in past'),

SysOperationDisplayOrderAttribute('1')]

**public** TransDate parmDate(TransDate \_dateValue = dateValue)

{

dateValue = \_dateValue;

**return** dateValue;

}

----New method:-

[DataMemberAttribute,

SysOperationLabelAttribute('Text Property'),

SysOperationHelpTextAttribute('Type some text'),

SysOperationDisplayOrderAttribute('2')]

**public** Description255 parmText(Description255 \_textValue = textValue)

{

textValue = \_textValue;

**return** textValue;

}

-----In class declaration:-

**public** **class** SysOperationServiceClassTwo **extends** SysOperationServiceBase

{

}

----New method:-

**public** **void** processOperation(SysOperationDataContractClass \_contract)

{

info ("Running SysOperation Batch Operation");

info ("Date Property: " + **date2Str**

(\_contract.parmDate(),

**213**,

DateDay::Digits2,

DateSeparator::Dot,

DateMonth::Long,

DateSeparator::Dot,

DateYear::Digits4));

info ("Text Property: " + \_contract.parmText());

}

-----In class declaration:-

**public** **class** SysOperationControllerClassMy **extends** SysOperationServiceController

{

}

----New method:-

**public** ClassDescription caption()

{

**return** "Batch Operation Task Description";

}

----New method:-

**public** **void** **new**()

{

**super**();

this.parmClassName(**classStr**(SysOperationServiceClassTwo));

this.parmMethodName(**methodStr**(SysOperationServiceClassTwo, processOperation));

this.parmDialogCaption("Batch Operation Dialog Title");

}

----New method:-

**public** **static** **void** main(Args args)

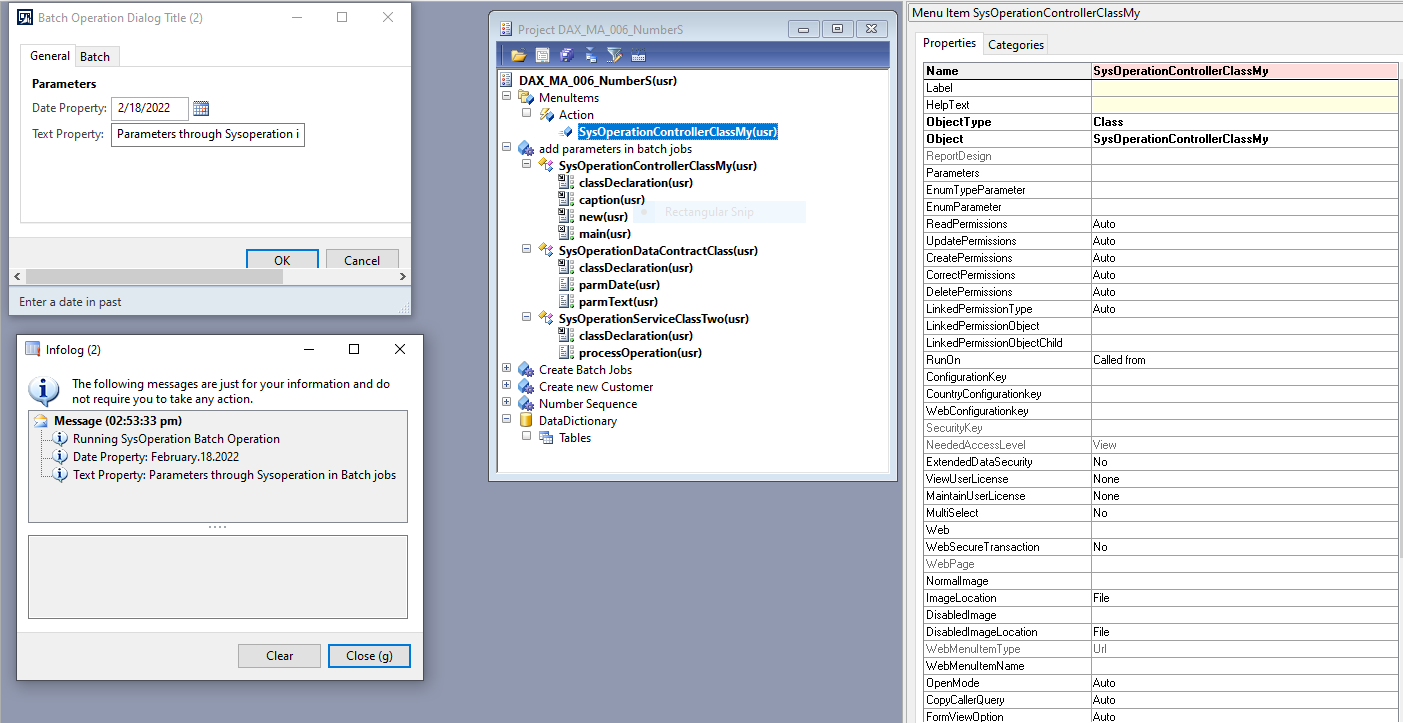
{

SysOperationControllerClassMy controller;

controller = **new** SysOperationControllerClassMy();

controller.startOperation();

}



### Add custom validation to dialog fields

\*) **Create 4 classes** :-- **Controller Class,** **Service Class,** **Data Contract Class,** **UI Builder Class**

1. **Validation Failed:-** if we select future date (more than current date) it will shows validation message ”please select past date”

[DataContractAttribute,

SysOperationContractProcessingAttribute(**classStr**(SysOperationUIBuilderClass))]

**public** **class** SysOperationDataContractClassTPL

{

TransDate dateValue;

Description255 textValue;

}

[DataMemberAttribute,

SysOperationLabelAttribute('Date Property'),

SysOperationHelpTextAttribute('Enter a date in past'),

SysOperationDisplayOrderAttribute('1')]

**public** TransDate parmDate(TransDate \_dateValue = dateValue)

{

dateValue = \_dateValue;

**return** dateValue;

}

[DataMemberAttribute,

SysOperationLabelAttribute('Text Property'),

SysOperationHelpTextAttribute('Type some text'),

SysOperationDisplayOrderAttribute('2')]

**public** Description255 parmText(Description255 \_textValue = textValue)

{

textValue = \_textValue;

**return** textValue;

}

**public** **class** SysOperationServiceClassPPL **extends** SysOperationServiceBase

{

}

**public** **void** processOperation(SysOperationDataContractClass \_contract)

{

info ("Running SysOperation Batch Operation");

info ("Date Property: " + **date2Str**

(\_contract.parmDate(),

**213**,

DateDay::Digits2,

DateSeparator::Dot,

DateMonth::Long,

DateSeparator::Dot,

DateYear::Digits4));

info ("Text Property: " + \_contract.parmText());

}

**public** **class** SysOperationControllerClassTT **extends** SysOperationServiceController

{

}

**public** ClassDescription caption()

{

**return** "Batch Operation Task Description";

}

**public** **void** **new**()

{

**super**();

this.parmClassName(**classStr**(SysOperationServiceClassPPL));

this.parmMethodName(**methodStr**(SysOperationServiceClassPPL, processOperation));

this.parmDialogCaption("Batch Operation Dialog Title");

}

**public** **static** **void** main(Args args)

{

SysOperationControllerClassTT controller;

controller = **new** SysOperationControllerClassTT();

controller.startOperation();

}

**public** **class** SysOperationUIBuilderClass **extends** SysOperationAutomaticUIBuilder

{

DialogField dateField;

}

**public** **void** postBuild()

{

**super**();

*// get references to dialog controls after creation*

dateField = this.bindInfo().getDialogField(this.dataContractObject(),

**methodStr**(SysOperationDataContractClassTPL, parmDate));

}

**public** **void** postRun()

{

**super**();

*// register overrides for form control events*

dateField.registerOverrideMethod(**methodstr**(FormDateControl,dateValue),

**methodstr**(SysOperationUIBuilderClass, validateDate), this);

}

**public** **boolean** validateDate(FormDateControl \_dateControl)

{

**if** (\_dateControl.dateValue() > **today**())

{

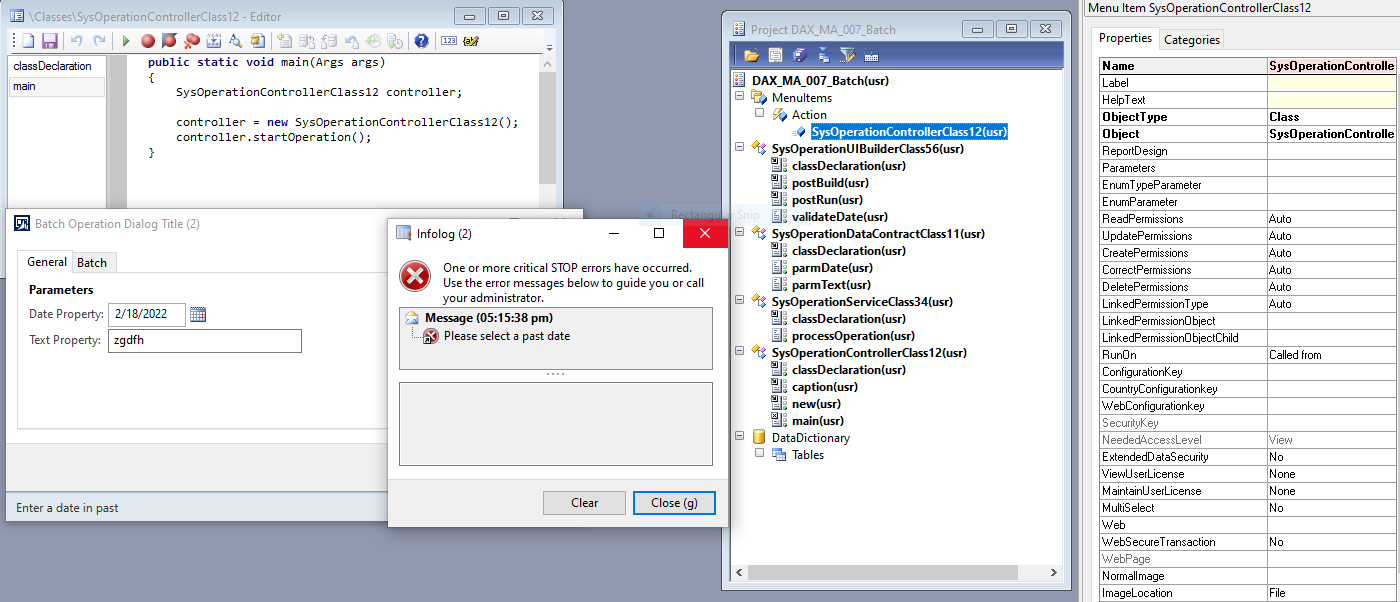
info("Please select a past date");

**return** **false**;

}

**return** **true**;

}

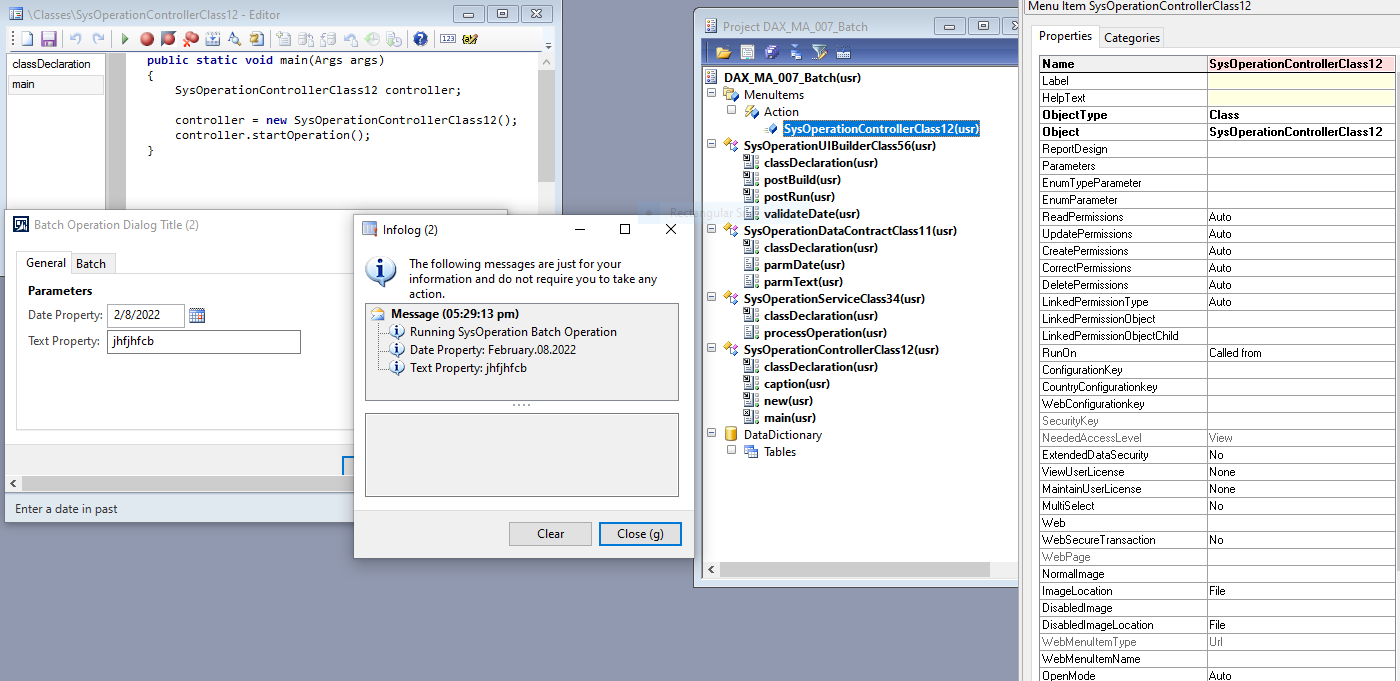


**Validation passed:-** if we select past date it will shows data

(“Running SysOperation Batch Operation

Date Property: February.17.2022

Text Property: jhfjhfcb”)



**Task:-**Select info from (FromDate,ToDate) and click ok it will display only those info dates

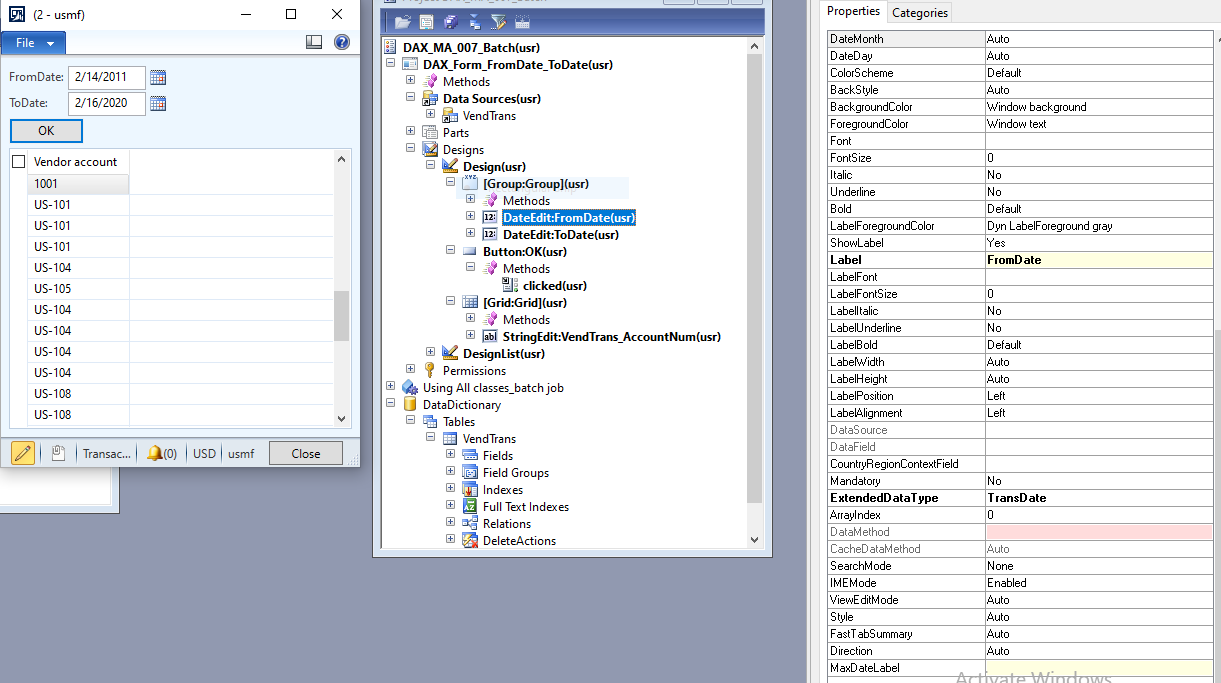
\*) create form , dateEdit(in a group—**take EDT TransDate**),button, AccountNum (field in the grid) , **--**after clicking press F5 you will get data**--**

**void** clicked()

{

VendTrans\_ds.executeQuery();

}



* **RDP Class Report in D365:-**

Create medel:- name/projectName and create // or take **fleet management**

Refrences:- ApplicationFoundation,-platform,-suite, ContactPerson, Directory,

fiscalbooks,SourceDocumentation,-type, Tax -Suitand-

2 tables:-Set property(TableType-TempDB)(headerTable,lineTable),create fields

3 classes :- Contract, DP, Controller classes and **build** the project

MenuItem(Output), set obj-name of **report**, objType-**SSRSReport**

Create Report (with 2 dataset)give datasourceType-ReportDataProvider,Query-

PurchaseOrderDP and create Design(drop above datasets to the

design) and give name to that design

Open CMD (Run As Administrator) sc queryex DynamicsAxBatch , taskkill /f /pid 5352(🡨|)

Go to Menu from (view,Appli Explorer,UserInterface,menus,AccountPayable)

Create Extentation,drop output menuitem–vendors,Deploy Report

Opem MicrosoftEdge:-(enter user,pwd),Account payable selest Your Report,ok

\_\_\_\_ You will see the Report created\_\_\_\_\_

**Contract Class**:-

**‌**[DataContractAttribute]

class PurchaseOrderContract

{

   PurchId purchId;

   [DataMemberAttribute('Purchase Id')]

   public PurchId parmPurchId(PurchId \_purchId = PurchId)

   {

       purchId = \_purchId;

       return purchId;

   }

}

**Controller class**:-

class PurchaseOrderController extends SrsReportRunController

{

   public static void main(Args \_args)

   {

       PurchaseOrderController controller = new PurchaseOrderController();

       controller.parmReportName(ssrsReportStr(PurchaseOrderReport,Report));

       controller.parmArgs(\_args);

       controller.parmShowDialog(false);

       controller.startOperation();

   }

}

**DP Class**:-

[SRSReportParameterAttribute(classstr(PurchaseOrderContract))]

class PurchaseOrderDP extends SRSReportDataProviderBase

{

   PurchaseOrderHeaderTmp  purchaseOrderHeaderTmp;

   PurchaseOrderLinesTmp   purchaseOrderLinesTmp;

   [SRSReportDataSetAttribute(tableStr(PurchaseOrderHeaderTmp))]

   public PurchaseOrderHeaderTmp getPurchaseOrderHeaderTmp()

   {

       select \* from purchaseOrderHeaderTmp;

       return purchaseOrderHeaderTmp;

   }

   [SRSReportDataSetAttribute(tableStr(PurchaseOrderLinesTmp))]

   public PurchaseOrderLinesTmp getPurchaseOrderLinesTmp()

   {

       select \* from purchaseOrderLinesTmp;

       return purchaseOrderLinesTmp;

   }

   public void processReport()

   {

       PurchTable  purchTable;

       PurchLine   purchLine;

       PurchaseOrderContract contract;

       PurchId     purchId;

       contract = this.parmDataContract() as PurchaseOrderContract;

       purchId = contract.parmPurchId();

       select \* from purchTable

           where purchTable.PurchId == purchId;

       purchaseOrderHeaderTmp.PurchId = purchTable.PurchId;

       purchaseOrderHeaderTmp.InvoiceAccount = purchTable.InvoiceAccount;

       purchaseOrderHeaderTmp.OrderAccount = purchTable.OrderAccount;

       purchaseOrderHeaderTmp.PurchName = purchTable.PurchName;

       purchaseOrderHeaderTmp.InclTax = purchTable.InclTax;

       purchaseOrderHeaderTmp.PaymMode = purchTable.PaymMode;

       purchaseOrderHeaderTmp.PurchStatus = purchTable.PurchStatus;

       purchaseOrderHeaderTmp.CurrencyCode = purchTable.CurrencyCode;

       purchaseOrderHeaderTmp.DeliveryDate = purchTable.DeliveryDate;

       purchaseOrderHeaderTmp.insert();

       while select \* from purchLine

           where purchLine.PurchId == purchTable.PurchId

       {

           purchaseOrderLinesTmp.LineNumber = purchLine.LineNumber;

           purchaseOrderLinesTmp.ItemId = purchLine.ItemId;

           purchaseOrderLinesTmp.PurchQty = purchLine.PurchQty;

           purchaseOrderLinesTmp.PurchPrice = purchLine.PurchPrice;

           purchaseOrderLinesTmp.PurchUnit = purchLine.PurchUnit;

           purchaseOrderLinesTmp.PriceUnit = purchLine.PriceUnit;

           purchaseOrderLinesTmp.LineAmount = purchLine.LineAmount;

           purchaseOrderLinesTmp.VendAccount = purchLine.VendAccount;

           purchaseOrderLinesTmp.VendGroup = purchLine.VendGroup;

           purchaseOrderLinesTmp.insert();

       }

   }

}

* **Query Based Report in D365:-**

Create **project** – update models (Extention) – create table , fields (add data)

Create **Query** (Drop Table into datasource) set Dynamics prop. Yes then No(delete some fields)

Create **Report** (in Datasets add Query) ---- build / Drop Query to Design

Create - **Output MenuItem** 🡪 set obj—Name of report , objtype – SSRS Report/ Extension

**CMD** – run 2 commands (SC queryex DynamicsAxBatch / taskkill /f /pid --) build & Deploy

------🡪 open in front end (Module) – USMF – in acc.payable (select your report) **menuitem** (ok)

* **Task Filter records:-**

In class declararion :-

**public** **class** FormRun **extends** ObjectRun

{

QueryBuildRange qbr;

}

@In Forms Datasource:-

**public** **void** init()

{

**super**();

qbr=this.query().dataSourceName('VendTrans').addRange(**fieldNum**(VendTrans,TransDate));

}

**public** **void** executeQuery()

{

**super**();

qbr.value(sysQuery::range(FromDate.dateValue(),

ToDate.dateValue()));

}

@On button:-

**void** clicked()

{

VendTrans\_ds.executeQuery();

}