TASK 1

**Retrieve the field name using the entity**

Xrm.Page.data.entity.getId('name');

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Function creation**

function LearningJS()

{

var accountName=Xrm.Page.getAttribute('name').getValue();

if(accountName != null)

{

alert(accountName);

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 2

**Filter subgrid**

function subgridfilter()

{

debugger;

   var Phonebusiness = window.parent.document.getElementById("contacts");

if(Phonebusiness==null){

setTimeout(function () { subgridfilter(); }, 2000);

}

else{

if(Phonebusiness !=null){

debugger;

var fetchXml = "<fetch version='1.0' output-format='xml-platform' mapping='logical' distinct='false'>" +

      "<entity name='contact'>" +

        "<attribute name='telephone1' />" +

        "<attribute name='fullname' />"+

        "<order attribute='telephone1' descending='false' />" +

        "<filter type='and'>" +

          "<condition attribute='telephone1' operator='like' value='%22710%' />" +

        "</filter>"+

      "</entity>" +

    "</fetch>";

Phonebusiness.control.SetParameter('fetchXml',fetchXml);

      Phonebusiness.control.refresh();

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK3

**Subgrid lookup filter**

function subgridlookupfilter()

{

debugger;

   var cmpnyname = window.parent.document.getElementById("contacts");

if(cmpnyname==null){

setTimeout(function () { subgridfilter(); }, 2000);

}

else{

if(cmpnyname !=null){

var fetchXml = "<fetch distinct='false' mapping='logical' output-format='xml-platform' version='1.0'>"+"<entity name='contact'>"+

"<attribute name='fullname'/>"+

"<attribute name='telephone1'/>"+

"<attribute name='contactid'/>"+

"<order descending='false' attribute='fullname'/>"+

"<filter type='and'>"+

"<condition attribute='parentcustomerid' value='{A56B3F4B-1BE7-E611-8101-E0071B6AF231}' uitype='account' uiname='A Datum Fabrication' operator='eq'/>"+

"</filter>"+

"</entity>"+

"</fetch>";

cmpnyname.control.SetParameter('fetchXml',fetchXml);

      cmpnyname.control.refresh();

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 4

**Create record using webapi**

var data =

    {

        "name": "Sample Account",

        "creditonhold": false,

        "address1\_latitude": 47.639583,

        "description": "This is the description of the sample account",

        "revenue": 5000000,

        "accountcategorycode": 1

    }

Xrm.WebApi.createRecord("account", data).then(

    function success(result) {

        console.log("Account created with ID: " + result.id);

    },

    function (error) {

        console.log(error.message);

    }

);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 5

**Delete account using guid**

function DeleteRecord() {

Xrm.WebApi.deleteRecord("account", "E6636252-9642-EA11-A812-000D3A0A7526").then(

function success(result) {

Xrm.Utility.alertDialog("Account deleted", null);

},

function (error) {

Xrm.Utility.alertDialog(error.message, null);

});

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 6

**Updating the lookup field**

var data =

{

"[primarycontactid@odata.bind":"/contacts(63A0E5B9-88DF-E311-B8E5-6C3BE5A8B200](mailto:primarycontactid@odata.bind%22:%22/contacts(63A0E5B9-88DF-E311-B8E5-6C3BE5A8B200))"

}

Xrm.WebApi.updateRecord("account","A16B3F4B-1BE7-E611-8101-E0071B6AF231", data)

.then(function success(result)

{

console.log("updated");

},

function (error) {

console.log(error.message);

} );

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 7

**Make disable the field and invisible the field**

if(Xrm.Page.getAttribute('new\_customer').getValue()=='28'){

Xrm.Page.ui.controls.get("new\_customername").setDisabled(true);

Xrm.Page.ui.controls.get("new\_amount").setVisible(false);}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 8

**Plugins create**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Xrm.Sdk;

namespace ClassLibrary1

{

    public class Class1:IPlugin

    {

     public void Execute(IServiceProvider serviceProvider)

    {

        IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

        if (context.InputParameters.Contains("Target"))

        {

                if (context.InputParameters["Target"] is Entity)

                {

                    Entity entity = (Entity)context.InputParameters["Target"];

                    try {

                    Entity followup = new Entity("task");

                    followup["subject"] = "send mail";

                    followup["description"] = "this is to send an email to the customer";

                    followup["scheduledstart"] = DateTime.Now;

                    followup["scheduledend"] = DateTime.Now.AddDays(2);                    if (context.OutputParameters.Contains("id"))

                    {

                        Guid regardingobjectid = new Guid(context.OutputParameters["id"].ToString());

                        string regardingobjectidType = "contact";

                        followup["regardingobjectid"] = new EntityReference(regardingobjectidType, regardingobjectid);

                    }

                    IOrganizationServiceFactory servicefactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

                    IOrganizationService client = servicefactory.CreateOrganizationService(context.UserId);

                        client.Create(followup);

                }catch(Exception e)

                    {

                        throw  new InvalidPluginExecutionException

                           (e.Message);

                    }

                 }

                }

            }

        }

    }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 9

**Plugins update**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Xrm.Sdk;

namespace ClassLibrary1

{

    public class Class1:IPlugin

    {

     public void Execute(IServiceProvider serviceProvider)

    {

        IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

          if (context.InputParameters.Contains("Target"))

        {

                if (context.InputParameters["Target"] is Entity)

                {

                    Entity entity = (Entity)context.InputParameters["Target"];

                    try {

                    Entity followup = new Entity("task");

                    followup["subject"] = "send mail";

                    followup["description"] = "this is to send an email to the customer";

                    followup["scheduledstart"] = DateTime.Now;

                    followup["scheduledend"] = DateTime.Now.AddDays(2);

                    followup["contact"]=entity.Id;

                    if (context.OutputParameters.Contains("id"))

                    {

                        Guid regardingobjectid = new Guid(context.OutputParameters["id"].ToString());

                        string regardingobjectidType = "contact";

                        followup["regardingobjectid"] = new EntityReference(regardingobjectidType, regardingobjectid);

                    }

                    IOrganizationServiceFactory servicefactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

                    IOrganizationService client = servicefactory.CreateOrganizationService(context.UserId);

                        client.Update(followup);

                }catch(Exception e)

                    {

                        throw  new InvalidPluginExecutionException

                           (e.Message);

                    }

                    }

                }

            }

        }

    }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 10

**Retrieve records**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary1

{

    public class Retrieve : IPlugin

       {

        public void Execute(IServiceProvider serviceProvider)

           {

            ITracingService tracingService = (ITracingService serviceProvider.GetService(typeof(ITracingService));

            tracingService.Trace("Statred");

            IPluginExecutionContext context = (IPluginExecutionContext serviceProvider.GetService(typeof(IPluginExecutionContext));

         IOrganizationServiceFactory servicefactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService client = servicefactory.CreateOrganizationService(context.UserId);

            Entity entity = (Entity)context.InputParameters["Target"];

            var guid = entity.Id;

            Entity records = client.Retrieve("new\_account", new Guid("AC65B4A7-4E47-EA11-A812-000D3A0A82CB"), new ColumnSet(true));

            //Console.WriteLine(guid);

            //lookup

            if (records.Attributes.Contains("new\_accparentaccount"))

            {

                string parentacc = ((EntityReference)(records.Attributes["new\_accparentaccount"])).Name;

                Guid accid = ((EntityReference)(records.Attributes["new\_accparentaccount"])).Id;

                // Console.WriteLine(accid);

                tracingService.Trace(parentacc);

            }

            //dateandtime

            if (records.Attributes.Contains("new\_accpublished"))

            {

                DateTime publish =    Convert.ToDateTime(records.Attributes["new\_accpublished"]);

                tracingService.Trace(publish.ToString());

            }

            //currency

            if (records.Attributes.Contains("new\_accprice"))

            {

                Money price = (Money)(records.Attributes["new\_accprice"]);

                tracingService.Trace(price.ToString());

            }

            //default name

            if (records.Attributes.Contains("new\_name"))

            {

                string defaultname = Convert.ToString(records.Attributes["new\_name"]);

                tracingService.Trace(defaultname);

            }

            //two options

            if (records.Attributes.Contains("new\_accgender"))

            {

                string gender = Convert.ToString(records.FormattedValues["new\_accgender"]);

                tracingService.Trace(gender);

            }

            //single line text

            if (records.Attributes.Contains("new\_accname"))

            {

                string accname = Convert.ToString(records.Attributes["new\_accname"]);

                tracingService.Trace(accname);

            }

            //multiselect

            if (records.Attributes.Contains("new\_accnumberofnames"))

            {

                string multiselect = records.FormattedValues["new\_accnumberofnames"];

                ArrayList a = new ArrayList();

                for (int i = 0; i < ((OptionSetValueCollection)records.Attributes["new\_accnumberofnames"]).Count; i++)

                {

                    a.Add(((OptionSetValueCollection)records.Attributes["new\_accnumberofnames"])[i].Value);

                }

                tracingService.Trace(multiselect);

                }

         }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 11

**Locking all the fields**

function disableFormFields()

{

//debugger;

  Xrm.Page.ui.controls.forEach(function (control, index) {

           var controlType = control.getControlType();

           if ((controlType != "iframe") && (controlType != "webresource") && (controlType != "subgrid"))

           {

if(Xrm.Page.getAttribute('new\_accountid').getValue() != null)

           {

               control.setDisabled(true);

           }

else{

control.setDisabled(false);

}

Xrm.Page.ui.controls.get("new\_accountid").setDisabled(false);

          }

       });

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 12

**Locking particular field**

<using for>

function setFieldReadOnly(){

var fields = new Array("telephone1","tickersymbol","customertypecode","address1\_composite","industrycode","donotemail","creditonhold","address1\_shippingmethodcode","msdyn\_taxexempt","msdyn\_workorderinstructions");

for(var i in fields ) {

Xrm.Page.ui.controls.get(fields[i]).setDisabled(true);

}

}

//Xrm.Page.getControl("transactioncurrencyid").setDisabled(true);

<using foreach>

function call(){

    var index=0;

var fields = new Array("telephone1","tickersymbol","customertypecode","address1\_composite","industrycode","donotemail","creditonhold","address1\_shippingmethodcode","msdyn\_taxexempt","msdyn\_workorderinstructions");

for(var i in fields ) {

Xrm.Page.ui.controls.get(fields[i]).setDisabled(true);

}

function myFunction(item, index)

{

    console.log(item);

}

fields.forEach(myFunction);}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 13

**creating lookup and set values**

function create()

{

debugger;

var contact = {

firstname: "Rkeerthana",

lastname: "K",

"[parentcustomerid\_account@odata.bind":"/accounts("+Xrm.Page.data.entity.getId().slice(1,-1](mailto:parentcustomerid_account@odata.bind%22:%22/accounts(%22+Xrm.Page.data.entity.getId().slice(1,-1))+")"

}

Xrm.WebApi.createRecord("contact", contact).then(

function success(result) {

        alert("Account created with ID: " + result.id);

    },

    function (error) {

        alert(error.message);

    }

);

}

**fetching lookup from account to contact when the account is known**

function lookup(){

debugger;

var lookupValue = new Array();

lookupValue[0] = new Object();

lookupValue[0].name =Xrm.Page.getAttribute("name").getValue();// enter the value directly.

lookupValue[0].id =Xrm.Page.data.entity.getId();//enter id directly

lookupValue[0].entityType = "account";//entity type from where the lookup is to be fetched

if(Xrm.Page.getAttribute("parentcustomerid")==null)

{

//console.log(lookupValue);

Xrm.Page.getAttribute("parentcustomerid").setValue(lookupValue);

}}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 14

**retrieving the dateandtime records using plugins**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

namespace ClassLibrary1

{

    public class dateandtime: IPlugin

    {

        public void Execute(IServiceProvider serviceProvider)

        {

            ITracingService tracingService = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            tracingService.Trace("Started");

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationServiceFactory servicefactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService client = servicefactory.CreateOrganizationService(context.UserId);

            QueryExpression q = new QueryExpression("new\_account");

            //q.EntityName = "contacts";

            q.ColumnSet = new ColumnSet(true);

            q.Criteria.AddCondition("new\_accpublished", ConditionOperator.NotNull);

            EntityCollection res = client.RetrieveMultiple(q);

            for (int i = 0; i < res.Entities.Count; i++)

            {

                string s1 = res.Entities[i].GetAttributeValue<DateTime>("new\_accpublished").ToString();

                tracingService.Trace(s1);

                string s2 = res.Entities[i].GetAttributeValue<string>("new\_name");

                tracingService.Trace(s2);

            }

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 15

**retrieve all the records**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary1

{

    public class Retrieve : IPlugin

           {

        public void Execute(IServiceProvider serviceProvider)

        {

            ITracingService tracingService = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            tracingService.Trace("Statred");

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationServiceFactory servicefactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService client = servicefactory.CreateOrganizationService(context.UserId);

            Entity entity = (Entity)context.InputParameters["Target"];

            var guid = entity.Id;

                Entity records = client.Retrieve("new\_account", new Guid("AC65B4A7-4E47-EA11-A812-000D3A0A82CB"), new ColumnSet(true));

            //Console.WriteLine(guid);

            //lookup

            if (records.Attributes.Contains("new\_accparentaccount"))

            {

                string parentacc = ((EntityReference)(records.Attributes["new\_accparentaccount"])).Name;

                Guid accid = ((EntityReference)(records.Attributes["new\_accparentaccount"])).Id;

                // Console.WriteLine(accid);

                tracingService.Trace(parentacc);

            }

            //dateandtime

            if (records.Attributes.Contains("new\_accpublished"))

            {

                DateTime publish = Convert.ToDateTime(records.Attributes["new\_accpublished"]);

                tracingService.Trace(publish.ToString());

            }

            //currency

            if (records.Attributes.Contains("new\_accprice"))

            {

                Money price = (Money)(records.Attributes["new\_accprice"]);

                tracingService.Trace(price.ToString());

            }

            //default name

            if (records.Attributes.Contains("new\_name"))

            {

                string defaultname = Convert.ToString(records.Attributes["new\_name"]);

                tracingService.Trace(defaultname);

            }

            //two options

            if (records.Attributes.Contains("new\_accgender"))

            {

                string gender = Convert.ToString(records.FormattedValues["new\_accgender"]);

                tracingService.Trace(gender);

            }

            //single line text

            if (records.Attributes.Contains("new\_accname"))

            {

                string accname = Convert.ToString(records.Attributes["new\_accname"]);

                tracingService.Trace(accname);

            }

            //multiselect

            if (records.Attributes.Contains("new\_accnumberofnames"))

            {

                string multiselect = records.FormattedValues["new\_accnumberofnames"];

                ArrayList a = new ArrayList();

                for (int i = 0; i < ((OptionSetValueCollection)records.Attributes["new\_accnumberofnames"]).Count; i++)

                {

                    a.Add(((OptionSetValueCollection)records.Attributes["new\_accnumberofnames"])[i].Value);

                }

                tracingService.Trace(multiselect);

                }

         }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 16

**retrieve optionset**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

namespace ClassLibrary1

{

    public class optionsetretrieve : IPlugin

    {

        public void Execute(IServiceProvider serviceProvider)

        {

            ITracingService tracingService = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            tracingService.Trace("Started");

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationServiceFactory servicefactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService client = servicefactory.CreateOrganizationService(context.UserId);

            QueryExpression q = new QueryExpression("contact");

            //q.EntityName = "contacts";

            q.ColumnSet = new ColumnSet(true);

            q.Criteria.AddCondition("preferredcontactmethodcode", ConditionOperator.NotNull);

            EntityCollection res = client.RetrieveMultiple(q);

            for (int i = 0; i < res.Entities.Count; i++)

            {

                //string s1 = res.Entities[i].GetAttributeValue<>("preferredcontactmethodcode").ToString();

                //tracingService.Trace(s1);

                string s2 = res.Entities[i].GetAttributeValue<string>("fullname");

                tracingService.Trace(s2);

            }

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 17

**retrieve multiple records**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using System.Collections;

namespace ClassLibrary1

{

    public class Retrieve\_multiple : IPlugin

    {

        public void Execute(IServiceProvider serviceProvider)

        {

            ITracingService tracingService = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            tracingService.Trace("Started");

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationServiceFactory servicefactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService client = servicefactory.CreateOrganizationService(context.UserId);

            QueryExpression q = new QueryExpression("contact");

            //q.EntityName = "contacts";

            q.ColumnSet = new ColumnSet(true);

            q.Criteria.AddCondition("telephone1", ConditionOperator.BeginsWith, 768);

            EntityCollection res = client.RetrieveMultiple(q);

            for (int i = 0; i < res.Entities.Count; i++)

            {

                string s1 = res.Entities[i].GetAttributeValue<string>("telephone1").ToString();

                tracingService.Trace(s1);

                string s2 = res.Entities[i].GetAttributeValue<string>("fullname");

                tracingService.Trace(s2);

            }

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 18

**update the records**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Xrm.Sdk;

namespace ClassLibrary1

{

    public class Update:IPlugin

    {

     public void Execute(IServiceProvider serviceProvider)

    {

        IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

          if (context.InputParameters.Contains("Target"))

        {

                if (context.InputParameters["Target"] is Entity)

                {

                    Entity entity = (Entity)context.InputParameters["Target"];

                    try {

                    Entity followup = new Entity("task");

                    followup["subject"] = "send mail";

                    followup["description"] = "this is to send an email to the customer";

                    followup["scheduledstart"] = DateTime.Now;

                    followup["scheduledend"] = DateTime.Now.AddDays(2);

                    followup["contact"]=entity.Id;

                    if (context.OutputParameters.Contains("id"))

                    {

                        Guid regardingobjectid = new Guid(context.OutputParameters["id"].ToString());

                        string regardingobjectidType = "contact";

                        followup["regardingobjectid"] = new EntityReference(regardingobjectidType, regardingobjectid);

                    }

                    IOrganizationServiceFactory servicefactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

                    IOrganizationService client = servicefactory.CreateOrganizationService(context.UserId);

                        client.Update(followup);

                }catch(Exception e)

                    {

                        throw  new InvalidPluginExecutionException

                           (e.Message);

                    }

                    }

                }

            }

        }

    }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 19

**lookup retrieve**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary1

{

   public class Lookupretrieve : IPlugin

    {

        public void Execute(IServiceProvider serviceProvider)

        {

            ITracingService tracingService = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            tracingService.Trace("Started");

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationServiceFactory servicefactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService client = servicefactory.CreateOrganizationService(context.UserId);

            QueryExpression q = new QueryExpression("account");

            q.ColumnSet = new ColumnSet("primarycontactid");

            //q.Criteria.AddCondition("primarycontactid", ConditionOperator.NotNull);

            LinkEntity l = new LinkEntity()

            {

                LinkFromEntityName = "account",

                LinkFromAttributeName = "primarycontactid",

                LinkToAttributeName = "contactid",

                LinkToEntityName = "contact",

                JoinOperator = JoinOperator.Inner,

                Columns = new ColumnSet("telephone1"),

                //LinkCriteria =  new LinkCriteria(AddCondition("telephone1", ConditionOperator.Like, "%555%"))

                LinkCriteria =

                    {

                      Conditions =

                                {

                                    new ConditionExpression("telephone1", ConditionOperator.Like, "%555%")

                                }

                    }

            };

            q.LinkEntities.Add(l);

            EntityCollection res = client.RetrieveMultiple(q);

            foreach (var phone in res.Entities)

            {

                EntityReference lookuprec = (EntityReference)phone["primarycontactid"];

                var id = lookuprec.Id;

                var name = lookuprec.Name;

                var logicalname = lookuprec.LogicalName;

                //string s1 = res.Entities[i].GetAttributeValue<int>("telephone1").ToString();

                tracingService.Trace(id.ToString());

                tracingService.Trace(name.ToString());

                tracingService.Trace(logicalname.ToString());

                //QueryExpression child = new QueryExpression("contact");

                //child.ColumnSet = new ColumnSet(true);

                //child.Criteria.AddCondition("telephone1", ConditionOperator.Like, 555);

                //EntityCollection r = client.RetrieveMultiple(child);

                //for (int j = 0; j < r.Entities.Count; j++)

                //{

                //    string s1 = r.Entities[j].GetAttributeValue<string>("emailaddress1").ToString();

                //    tracingService.Trace(s1);

                //    string s2 = r.Entities[j].GetAttributeValue<string>("telephone1");

                //    tracingService.Trace(s2);

                //}

            }

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 20

**retrieve currency**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

namespace ClassLibrary1

{

    public class currency : IPlugin

    {

        public void Execute(IServiceProvider serviceProvider)

        {

            ITracingService tracingService = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            tracingService.Trace("Started");

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationServiceFactory servicefactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService client = servicefactory.CreateOrganizationService(context.UserId);

            QueryExpression q = new QueryExpression("new\_account");

            //q.EntityName = "contacts";

            q.ColumnSet = new ColumnSet(true);

            q.Criteria.AddCondition("new\_accprice", ConditionOperator.NotNull);

            EntityCollection res = client.RetrieveMultiple(q);

            for (int i = 0; i < res.Entities.Count; i++)

            {

                string s1 = res.Entities[i].GetAttributeValue<Money>("new\_accprice").ToString();

                tracingService.Trace(s1);

                string s2 = res.Entities[i].GetAttributeValue<string>("new\_name");

                tracingService.Trace(s2);

            }

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 21

**Dependent optionset using js**

function OptionSetAddRemove() {

    var type = Xrm.Page.getAttribute("new\_catagory").getValue();

    var optionSet = Xrm.Page.ui.controls.get("new\_catagorysub");

    var optionSetValues = optionSet.getAttribute().getOptions();

    if(type = "100000001")

    {

        optionSet.clearOptions();

        optionSetValues.forEach(function (element) {

            if (element.value == "100000002" || element.value == "100000001" )

            {

                optionSet.addOption(element);

            }

            if (element.value == "100000003" || element.value == "100000000")

            {

               optionSet.removeOption(element);

            }

        });

    }

    else if(type = "100000000")

    {

        optionSet.clearOptions();

        optionSetValues.forEach(function (element) {

            if (element.value == "100000003" || element.value == "100000000" )

            {

                optionSet.addOption(element);

            }

            if (element.value == "100000001" || element.value == "100000002" )

            {

                optionSet.removeOption(element);

            }

        });

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 22

**Console application standard code for connecting CRM and plugins**

string url = "<https://syamsundarinfotech.crm8.dynamics.com/>";

// e.g. [you@yourorg.onmicrosoft.com](mailto:you@yourorg.onmicrosoft.com)

string userName = "[Edwin@SyamSundarinfotech.onmicrosoft.com](mailto:Edwin@SyamSundarinfotech.onmicrosoft.com)";

// e.g. y0urp455w0rd

string password = "Fernando@1345";

string conn = $@"

Url = {url};

AuthType = Office365;

UserName = {userName};

Password = {password};

RequireNewInstance = True";

service = new CrmServiceClient(conn);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 22-1

**another method**

IOrganizationService organizationService = null;

try

{

    ClientCredentials clientCredentials = new ClientCredentials();

    clientCredentials.UserName.UserName = "[logi@techlogi.onmicrosoft.com](mailto:logi@techlogi.onmicrosoft.com)";

    clientCredentials.UserName.Password = "pass@123";

    // For Dynamics 365 Customer Engagement V9.X, set Security Protocol as TLS12

    ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;

    // Get the URL from CRM, Navigate to Settings -> Customizations -> Developer Resources

   // Copy and Paste Organization Service Endpoint Address URL

   organizationService = (IOrganizationService)new OrganizationServiceProxy(new   Uri("https://techlogi.api.crm8.dynamics.com/XRMServices/2011/Organization.svc"),                 null, clientCredentials, null);

        if (organizationService != null)

        {

            Console.Write("connected successfully")

        }

        else

        {

            Console.WriteLine("Failed to Established Connection!!!");

        }

}

catch (Exception ex)

{

     Console.WriteLine("Exception caught - " + ex.Message);

}

Console.ReadKey();

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 23

**Console connections**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Client;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Reflection;

using System.ServiceModel.Description;

using System.Text;

using System.Threading.Tasks;

namespace newstep

{

    class Program :IPlugin

    {

        static void Main(string[] args)

        {

            IOrganizationService organizationService = null;

            try

            {

                string AssemblyName = typeof(Program).Assembly.GetName().Name;

                Console.WriteLine(AssemblyName);

                Console.ReadKey();

                ClientCredentials clientCredentials = new ClientCredentials();

                clientCredentials.UserName.UserName = "[john@infocarter.onmicrosoft.com](mailto:john@infocarter.onmicrosoft.com)";

                clientCredentials.UserName.Password = "pass@123";

                // For Dynamics 365 Customer Engagement V9.X, set Security Protocol as TLS12

                ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;

                // Get the URL from CRM, Navigate to Settings -> Customizations -> Developer Resources

                // Copy and Paste Organization Service Endpoint Address URL

                organizationService = (IOrganizationService)new OrganizationServiceProxy(new Uri("https://selva.api.crm8.dynamics.com/XRMServices/2011/Organization.svc"),                 null, clientCredentials, null);

                if (organizationService != null)

                {

                    Console.Write("connected successfully");

                }

                else

                {

                    Console.WriteLine("Failed to Established Connection!!!");

                }

            }

            catch (Exception ex)

            {

                Console.WriteLine("Exception caught - " + ex.Message);

            }

            Console.ReadKey();

            int choice;

            String account\_name;

            Console.WriteLine("Press 1 to Create a new account");

            choice = Convert.ToInt32(Console.ReadLine());

            Entity acc = new Entity("account");

            switch (choice)

            {

                case 1:

                    Console.WriteLine("Enter Name of Account to Create ?");

                    account\_name = Console.ReadLine();

                    acc["name"] = account\_name;

                    organizationService.Create(acc);

                    Console.WriteLine("An account with name {0} is created successfully", account\_name);

                    Console.WriteLine();

                    Console.WriteLine("Press any key to exit..");

                    Console.ReadKey();

                    break;

                default:

                    Console.WriteLine("Wrong input...");

                    Console.ReadKey();

                    break;

            }

        }

        public void Execute(IServiceProvider serviceProvider)

        {

            throw new NotImplementedException();

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 24

**Connection plugins and crm**

IOrganizationService organizationService = null;

            try

            {

                ClientCredentials clientCredentials = new ClientCredentials();

                clientCredentials.UserName.UserName = "[john@infocarter.onmicrosoft.com](mailto:john@infocarter.onmicrosoft.com)";

                clientCredentials.UserName.Password = "pass@123";

                ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;

                organizationService = (IOrganizationService)new OrganizationServiceProxy(new Uri("https://selva.api.crm8.dynamics.com/XRMServices/2011/Organization.svc"),                null, clientCredentials, null);

                if (organizationService != null)

                {

                    QueryExpression qe = new QueryExpression { EntityName = "plugintype", ColumnSet = new ColumnSet("plugintypeid") };

                    qe.Criteria.AddCondition("typename", ConditionOperator.Equal, "ConsoleApplication7.Program");

                    EntityCollection ec=organizationService.RetrieveMultiple(qe);

                    QueryExpression query = new QueryExpression("sdkmessage");

                    query.Criteria.AddCondition("name", ConditionOperator.Equal, "Create");

                    EntityCollection entities = organizationService.RetrieveMultiple(query);

                    var s = entities.Entities.Select(e => e.Id).SingleOrDefault();

                    string StepName = "create accountss";

                    var step = new Entity("sdkmessageprocessingstep");

                    step["name"] = StepName;

                    step["configuration"] = "";

                    step["mode"] = new OptionSetValue(0);

                    step["rank"] = 1;

                    step["stage"] = new OptionSetValue(40);

                    step["supporteddeployment"] = new OptionSetValue(0); ;

                    step["invocationsource"] = new OptionSetValue(0);//0-parent 1-child

// step["plugintypeid"] = new EntityReference("plugintype", pluginTypeId);

// step["assemblyid"] = "1fb173ec-445b-42af-86bf-7c01d4d34840";                                                                     //step["isolationMode"] = "sandbox";                                                                   //step["entityName"] = "account";                                                                     //step["asyncAutoDelete"] = false;

                    step["sdkmessageid"] = new EntityReference("sdkmessage", s);

                    step["plugintypeid"] = new EntityReference("plugintype", ec[0].Id);

                    organizationService.Create(step);

                    Console.Write("connected successfully");

                    }

                else

                {

                    Console.WriteLine("Failed to Established Connection!!!");

                }

            }

            catch (Exception ex)

            {

                Console.WriteLine("Exception caught - " + ex.Message);

            }

            Console.ReadKey();

        }

        public void Execute(IServiceProvider serviceProvider)

        {

            throw new NotImplementedException();

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 25

**Preoperation**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace postpre

{

    public class preoperation : IPlugin

    {

        public void Execute(IServiceProvider serviceProvider)

        {

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationServiceFactory serviceFactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService service = serviceFactory.CreateOrganizationService(context.UserId);

            ITracingService trace = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            trace.Trace("test");

            Entity entity = (Entity)context.InputParameters["Target"];

            Entity records = service.Retrieve("account", entity.Id, new ColumnSet(true));

            if (records.Attributes.Contains("name"))

            {

                entity.Attributes["new\_desc"] = records.Attributes["name"];

                trace.Trace(records.Attributes["name"].ToString());

            }

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 26

**prevalidation**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace postpre

{

    public class prevalidation : IPlugin

    {

        public void Execute(IServiceProvider serviceProvider)

        {

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationServiceFactory serviceFactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService service = serviceFactory.CreateOrganizationService(context.UserId);

            ITracingService trace = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            if (context.InputParameters.Contains("Target"))

            //{

                if (context.InputParameters["Target"] is Entity)

                {

                    Entity entity = (Entity)context.InputParameters["Target"];

                    try

                    {

                        QueryExpression query = new QueryExpression("account");

                        query.ColumnSet = new ColumnSet(new string[] { "name" });

                        query.Criteria.AddCondition(new ConditionExpression("name", ConditionOperator.Equal, "dfg"));

                        EntityCollection events = service.RetrieveMultiple(query);

                        foreach (var event1 in events.Entities)

                        {

                                service.Delete("account", event1.Id);

                        }

                    }

                    catch(Exception)

                    {

                    }

                }

            }

        }

    }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 27

**Autonumbergeneration**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary1

{

    public class Autonumbergeneraton : IPlugin

    {

        public void Execute(IServiceProvider serviceProvider)

        {

            ITracingService tracingService = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            int autoNumber = 0;

            string prefix = string.Empty;

            string suffix = string.Empty;

            if (context.InputParameters.Contains("Target") && context.InputParameters["Target"] is Entity)

            {

                Entity entity = (Entity)context.InputParameters["Target"];

                IOrganizationServiceFactory serviceFactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

                IOrganizationService service = serviceFactory.CreateOrganizationService(context.UserId);

                if (context.MessageName == "Create")

                {

                   QueryExpression query = new QueryExpression("new\_countersettings");

                    FilterExpression childFilter = query.Criteria.AddFilter(LogicalOperator.And);

                    childFilter.AddCondition("new\_entityname", ConditionOperator.Equal, entity.LogicalName);

                    query.ColumnSet = new ColumnSet(true);

                    EntityCollection entitycollection = service.RetrieveMultiple(query);

                    if (entitycollection.Entities.Count > 0)

                    {

                        Entity \_dynamicsEntity = entitycollection.Entities[0];                        if (\_dynamicsEntity.Attributes.Contains("new\_currentvalue"))

                        {

                            autoNumber = Convert.ToInt32(\_dynamicsEntity.Attributes["new\_currentvalue"]) + 1;

                        }

                        #region Prefix Handling

                        if (\_dynamicsEntity.Attributes.Contains("new\_prefix"))

                        {

                            prefix = \_dynamicsEntity.Attributes["new\_prefix"].ToString();

                        }

                        #endregion

                        #region Updating Autonumber

                        if (\_dynamicsEntity.Attributes.Contains("new\_targetattribute"))

                        {

                            string \_attributeName = \_dynamicsEntity.Attributes["new\_targetattribute"].ToString();                            \_dynamicsEntity.Attributes["new\_prefix"] = prefix;

                            \_dynamicsEntity.Attributes["new\_currentvalue"] = autoNumber;

                            entity.Attributes[\_attributeName] = prefix + autoNumber.ToString();

                            service.Update(entity);

                        }

                        #endregion

                    }

                }

            }

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 28

**WEBSERVICE**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Services;

namespace WebApplication1

{

    [WebService(Namespace = "<http://tempuri.org/>")]

    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

    [System.ComponentModel.ToolboxItem(false)]

    public class WebService1 : System.Web.Services.WebService

    {

        [WebMethod]

        public string HelloWorld()

        {

            return "Hello World";

        }

        [WebMethod]

        public string Addition(int value1, int value2)

        {

            int result = value1 + value2;

            return "Addition= " + result.ToString();

        }

    }

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Net;

using System.IO;

using System.Web.Services.Description;

using System.Web.Services;

namespace ConsoleApplication1

{

    class Program

    {

        static void Main(String[] args)

        {

            localhost.WebService1 obj = new localhost.WebService1();

            obj.Url = "<http://localhost:51062/WebService1.asmx>";

            Console.Write("enter the value:");

            int value1 = Convert.ToInt32(Console.ReadLine());

            Console.Write("enter the value:");

        int value2 = Convert.ToInt32(Console.ReadLine());

            string result = obj.Addition(value1, value2);

        Console.WriteLine(result);

            Console.ReadKey();

    }

  }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 29

**plugin registration tool**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Client;

using Microsoft.Xrm.Sdk.Query;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.ServiceModel.Description;

using System.Text;

using System.Threading.Tasks;

namespace pluginregistrationtool

{

    public class Class1:IPlugin

    {

        enum CrmPluginStepDeployment

        {

            ServerOnly = 0,

            OfflineOnly = 1,

            Both = 2

        }

        enum CrmPluginStepMode

        {

            Asynchronous = 1,

            Synchronous = 0

        }

        enum CrmPluginStepStage

        {

            PreValidation = 10,

            PreOperation = 20,

            PostOperation = 40,

        }

        enum SdkMessageName

        {

            Create,

            Update,

            Delete,

            Retrieve,

            Assign,

            GrantAccess,

            ModifyAccess,

            RetrieveMultiple,

            RetrievePrincipalAccess,

            RetrieveSharedPrincipalsAndAccess,

            RevokeAccess,

            SetState,

            SetStateDynamicEntity,

        }

        public void Execute(IServiceProvider serviceProvider)

        {

            IOrganizationService organizationService = null;

            try

            {

                ClientCredentials clientCredentials = new ClientCredentials();

                clientCredentials.UserName.UserName = "[john@infocarter.onmicrosoft.com](mailto:john@infocarter.onmicrosoft.com)";

                clientCredentials.UserName.Password = "pass@123";

                ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;

                organizationService = (IOrganizationService)new OrganizationServiceProxy(new Uri("https://selva.api.crm8.dynamics.com/XRMServices/2011/Organization.svc"),

                null, clientCredentials, null);

                if (organizationService != null)

                {

                    QueryExpression qe = new QueryExpression { EntityName = "plugintype", ColumnSet = new ColumnSet("plugintypeid") };

                    qe.Criteria.AddCondition("typename", ConditionOperator.Equal, "ConsoleApplication7.Program");

                    EntityCollection ec = organizationService.RetrieveMultiple(qe);

                    QueryExpression query = new QueryExpression("sdkmessage");

                    query.Criteria.AddCondition("name", ConditionOperator.Equal, "Create");

                    EntityCollection entities = organizationService.RetrieveMultiple(query);

                    var s = entities.Entities.Select(e => e.Id).SingleOrDefault();

                    string StepName = "create accountss";

                    var step = new Entity("sdkmessageprocessingstep");

                    step["name"] = StepName;

                    step["configuration"] = "";

                    step["mode"] = new OptionSetValue(0);

                    step["rank"] = 1;

                    step["stage"] = new OptionSetValue(40);

                    step["supporteddeployment"] = new OptionSetValue(0); ;

                    step["invocationsource"] = new OptionSetValue(0);//0-parent 1-child

                                                                     // step["plugintypeid"] = new EntityReference("plugintype", pluginTypeId);

                                                                     // step["assemblyid"] = "1fb173ec-445b-42af-86bf-7c01d4d34840";

                                                                     //step["isolationMode"] = "sandbox";

                                                                     //step["entityName"] = "account";

                                                                     //step["asyncAutoDelete"] = false;

                    step["sdkmessageid"] = new EntityReference("sdkmessage", s);

                    step["plugintypeid"] = new EntityReference("plugintype", ec[0].Id);

                    organizationService.Create(step);

                    Console.Write("connected successfully");

                }

                else

                {

                    Console.WriteLine("Failed to Established Connection!!!");

                }

            }

            catch (Exception ex)

            {

                Console.WriteLine("Exception caught - " + ex.Message);

            }

            Console.ReadKey();

        }

    }

    }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 30

**retry the page once failed**

using System;

using System.Net;

using System.IO;

using System.Net.Http;

using System.Threading;

namespace ConsoleApplication2

{

    class Program

    {

        static void Main(string[] args)

        {

            const int maxretries = 3;

            for (int i = 0; i < maxretries; i++)

            {

                string completeUrl = "<http://localhost:51062/WebService1.asmx>";

                WebRequest request = WebRequest.Create(completeUrl);

                HttpWebResponse response = (HttpWebResponse)request.GetResponse();

                Stream dataStream = response.GetResponseStream();

                StreamReader reader = new StreamReader(dataStream);

                string responseFromServer = reader.ReadToEnd();

                if (response.StatusDescription.Equals("OK"))

                {

                    Console.Write("enter the value:");

                    int value1 = Convert.ToInt32(Console.ReadLine());

                    Console.Write("enter the value:");

                    int value2 = Convert.ToInt32(Console.ReadLine());

                    string result = Convert.ToString(value1 + value2);

                    Console.WriteLine(j);

                    break;

                    //Console.ReadKey();

                }

            }

            reader.Close();

            dataStream.Close();

            Console.ReadKey();

        }

        }

    }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 31

**custom workflow**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace custom\_workflow

{

    class Class1

    {

        public static void Main(string[] args)

        {

            for(int i=0;i<25;i++)

            {

                for(int j=0;j<1;j++)

                {

                    Console.Write(j);

                }

                if(i==5||i==10||i==15||i==20)

                {

                    Console.Write("\n");

                }

            }

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 32

**displaying alert using plugins**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

namespace alertusingc

{

    public class Class1 : IPlugin

    {

        private Guid guid;

        private EntityCollection con;

        public void Execute(IServiceProvider serviceProvider)

        {

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationServiceFactory factory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService service = factory.CreateOrganizationService(context.UserId);

            if (context.InputParameters.Contains("Target") && context.InputParameters["Target"] is Entity)

            {

                Entity Entitycontact = (Entity)context.InputParameters["Target"];

                //guid=EntityReferenceTarget.Id;

                Entity contact = service.Retrieve(Entitycontact.LogicalName, Entitycontact.Id, new ColumnSet(true));

                if (contact.Attributes.Contains("emailaddress1") == true)

                {

                    QueryExpression query = new QueryExpression("contact");

                    query.ColumnSet = new ColumnSet(true);

                    query.Criteria.AddCondition("emailaddress1", ConditionOperator.Equal, contact.Attributes["emailaddress1"]);

                    query.Criteria.AddCondition("contactid", ConditionOperator.NotEqual, Entitycontact.Id);

                    con = service.RetrieveMultiple(query);

                }

                if (con.Entities.Count > 0)

                {

                    throw new InvalidPluginExecutionException(@"already exist" + con.Entities.Count + "");

                }

                else

                {

                    throw new InvalidPluginExecutionException(@"updated" + con.Entities.Count + "");

                }

            }

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 33

**email validation using plugins**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Remoting.Contexts;

using System.Text;

using System.Threading.Tasks;

using System.Web.UI.WebControls;

namespace postpre

{

    public class emailvalidation : IPlugin

    {

        public void Execute(IServiceProvider serviceProvider)

        {

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationServiceFactory serviceFactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService service = serviceFactory.CreateOrganizationService(context.UserId);

            ITracingService trace = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            Entity entity = (Entity)context.InputParameters["Target"];

            Entity records = service.Retrieve("contact", entity.Id, new ColumnSet(true));

            string emailid;

            try

            {

                if (records.Attributes.Contains("emailaddress1"))

                {

                    emailid = Convert.ToString(records.Attributes["emailaddress1"]);

                    if (records.Attributes.Contains("emailaddress1"))

                    {

                        QueryExpression query = new QueryExpression()

                        {

                            EntityName = "contact",

                            Criteria = new FilterExpression(LogicalOperator.And),

                            ColumnSet = new ColumnSet(true)

                        };

                        query.Criteria.AddCondition("contactid", ConditionOperator.NotEqual, entity.Id);

                        query.Criteria.AddCondition("emailaddress1", ConditionOperator.Equal, emailid);

                        EntityCollection res = service.RetrieveMultiple(query);

                        trace.Trace(res.Entities.Count.ToString());

                        // try

                        //{

                        if (res.Entities.Count > 0)

                        {

                            trace.Trace("already exist");

                        }

                        else

                        {

                            trace.Trace(emailid);

                        }

                    }

                }

            }

            catch (Exception)

            {

            }

                }

            }

        }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 34

**send mail in console application using plugin**

using Microsoft.Crm.Sdk.Messages;

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Client;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.ServiceModel.Description;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication4

{

    class Program

    {

        static void Main(string[] args)

        {

            IOrganizationService organizationService = null;

            try

            {

                ClientCredentials clientCredentials = new ClientCredentials();

                clientCredentials.UserName.UserName = "[john@infocarter.onmicrosoft.com](mailto:john@infocarter.onmicrosoft.com)";

                clientCredentials.UserName.Password = "pass@123";

                // For Dynamics 365 Customer Engagement V9.X, set Security Protocol as TLS12

                ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;

                // Get the URL from CRM, Navigate to Settings -> Customizations -> Developer Resources

                // Copy and Paste Organization Service Endpoint Address URL

                organizationService = (IOrganizationService)new OrganizationServiceProxy(new Uri("https://selva.api.crm8.dynamics.com/XRMServices/2011/Organization.svc"), null, clientCredentials, null);

                if (organizationService != null)

                {

                    Console.Write("connected successfully");

                }

                else

                {

                    Console.WriteLine("Failed to Established Connection!!!");

                }

            }

            catch (Exception ex)

            {

                Console.WriteLine("Exception caught - " + ex.Message);

            }

            Console.ReadKey();

            Entity fromActivityParty = new Entity("activityparty");

            Entity toActivityParty = new Entity("activityparty");

            Guid contactId = new Guid("EEA08F7C-E05E-EA11-A811-000D3AF057DD");

            fromActivityParty["partyid"] = new EntityReference("systemuser", new Guid("3DC91032-F77E-4667-998F-A5D53580C384"));

            toActivityParty["partyid"] = new EntityReference("contact", contactId);

            Entity email = new Entity("email");

            email["from"] = new Entity[] { fromActivityParty };

            email["to"] = new Entity[] { toActivityParty };

            email["regardingobjectid"] = new EntityReference("contact", contactId);

            email["subject"] = "This is the subject";

            email["description"] = "This is the description.";

            email["directioncode"] = true;

            Guid emailId = organizationService.Create(email);

            // Use the SendEmail message to send an e-mail message.

            SendEmailRequest sendEmailRequest = new SendEmailRequest

            {

                EmailId = emailId,

                TrackingToken = "",

                IssueSend = true

            };

            SendEmailResponse sendEmailresp = (SendEmailResponse)organizationService.Execute(sendEmailRequest);

            Console.WriteLine("Email sent");

            Console.ReadLine();

       }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 35

**send email if some tries to create same mail**

using Microsoft.Crm.Sdk.Messages;

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Sendmail

{

    public class Class1 : IPlugin

    {

        public void Execute(IServiceProvider serviceProvider)

        {

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationServiceFactory serviceFactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService service = serviceFactory.CreateOrganizationService(context.UserId);

            ITracingService trace = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            Entity entity = (Entity)context.InputParameters["Target"];

            Entity records = service.Retrieve("contact", entity.Id, new ColumnSet(true));

            string emailid;

            try

            {

                if (records.Attributes.Contains("emailaddress1"))

                {

                    emailid = Convert.ToString(records.Attributes["emailaddress1"]);

                    if (records.Attributes.Contains("emailaddress1"))

                    {

                        QueryExpression query = new QueryExpression()

                        {

                            EntityName = "contact",

                            Criteria = new FilterExpression(LogicalOperator.And),

                            ColumnSet = new ColumnSet(true)

                        };

                        query.Criteria.AddCondition("contactid", ConditionOperator.NotEqual, entity.Id);

                        query.Criteria.AddCondition("emailaddress1", ConditionOperator.Equal, emailid);

                        EntityCollection res = service.RetrieveMultiple(query);

                        trace.Trace(res.Entities.Count.ToString());

                        // try

                        //{

                        if (res.Entities.Count > 0)

                        {

                            if (context.InputParameters.Contains("Target"))

                            {

                                if (context.InputParameters["Target"] is Entity)

                                {

                                    //string emailBody = "send mail";

                                    //string emailSubject = "duplicate mail";

                                    Guid OwnerID;

                                    OwnerID = ((EntityReference)(records.Attributes["ownerid"])).Id;

                                    try

                                    {

                                        Entity fromActivityParty = new Entity("activityparty");

                                        Entity toActivityParty = new Entity("activityparty");

                                            Guid contactId = res.Entities[0].Id;//context.PrimaryEntityId;(current contact id)

                                            fromActivityParty["partyid"] = new EntityReference("systemuser", OwnerID);

                                            toActivityParty["partyid"] = new EntityReference("contact", contactId);

                                            Entity email = new Entity("email");

                                            email["from"] = new Entity[] { fromActivityParty };

                                            email["to"] = new Entity[] { toActivityParty };

                                            email["regardingobjectid"] = new EntityReference("contact", contactId);

                                            email["subject"] = "Duplicate mail";

                                            email["description"] = "someone trying to create duplicate mail";

                                            email["directioncode"] = true;

                                            Guid emailId = service.Create(email);

                                            // Use the SendEmail message to send an e-mail message.

                                            SendEmailRequest sendEmailRequest = new SendEmailRequest

                                            {

                                                EmailId = emailId,

                                                TrackingToken = "",

                                                IssueSend = true

                                            };

                                        SendEmailResponse sendEmailresp = (SendEmailResponse)service.Execute(sendEmailRequest);

                                        trace.Trace("mail created");

                                       }

                                    catch (Exception)

                                    {

                                    }

                                }

                                else

                                {

                                    trace.Trace(emailid);

                                }

                           }

                        }

                    }

                }

            }

            catch (Exception)

            {

            }

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 36

**update currency field in account with adding the contact currency field**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Addmoney

{

    public class Class1 : IPlugin

    {

        private EntityReference contactentity;

        decimal cost;

        decimal total=0;

        //private EntityReference contactid;

        public void Execute(IServiceProvider serviceProvider)

        {

            ITracingService tracingService = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationServiceFactory servicefactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService client = servicefactory.CreateOrganizationService(context.UserId);

            Entity entity = (Entity)context.InputParameters["Target"];

            // ColumnSet cols = new ColumnSet(new string[] { "fullname", "parentcustomerid" });

            //Guid id = context.PrimaryEntityId;

            Entity retrContact = (Entity)client.Retrieve("contact", entity.Id, new ColumnSet(true));

            var parentAccount = (EntityReference)retrContact.Attributes["parentcustomerid"];

            var companyStr = parentAccount.Name;

            tracingService.Trace(companyStr);

            // string contactid;

            //try

            //{

                if (retrContact.Attributes.Contains("parentcustomerid"))

                {

                    contactentity = retrContact.GetAttributeValue<EntityReference>("parentcustomerid");

                    //contactid = retrContact.Attributes["parentcustomerid"];

                    if (retrContact.Attributes.Contains("parentcustomerid"))

                    {

                        QueryExpression query = new QueryExpression()

                        {

                            EntityName = "contact",

                            //Criteria = new FilterExpression(LogicalOperator.And),

                            ColumnSet = new ColumnSet(true)

                        };

                        // query.Criteria.AddCondition("contactid", ConditionOperator.NotEqual, entity.Id);

                        query.Criteria.AddCondition("parentcustomerid", ConditionOperator.Equal, contactentity.Id);

                        EntityCollection res = client.RetrieveMultiple(query);

                        tracingService.Trace(res.Entities.Count.ToString());

                        if (res != null && res.Entities.Count > 0)

                        {

                            foreach (Entity en in res.Entities)//loop through every record

                            {

                                cost = (((Money)en.Attributes["new\_currency"]).Value);

                                total += cost;

                            }

                            tracingService.Trace(total.ToString());

                        // Entity acc = new Entity("account");

                        // Guid Id = context.PrimaryEntityId;

                        //// if (acc.Attributes.Contains("new\_currency"))

                        //// {

                        //     //Guid accid = acc.Id;

                        //     acc["new\_currency"] = total;

                        //     client.Update(acc);

                        //     tracingService.Trace(acc.ToString());

                        //Entity updateaccount = new Entity("account");

                        //string id = entity.Id.ToString();

                        //updateaccount["new\_currency"] = total;

                        //client.Update(updateaccount);

                        Entity acc = new Entity("account");

                        acc.Attributes["accountid"] = contactentity.Id;

                        acc.Attributes["new\_currency"] = new Money(total);

                        client.Update(acc);

                        tracingService.Trace(acc.ToString());

                        // }

                    }

                        }

               // }

            }

            //catch (Exception)

            //{

            //}

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 37

**Autosave using js**

window.parent.Xrm.Page.data.entity.save();

Xrm.Page.data.entity.save();

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 39

**update currency field in account with adding the contact currency field with null values (updating in account)**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Updateacc

{

    public class Class1 : IPlugin

    {

        private EntityReference contactentity;

        private decimal cost=0;

        private decimal total=0;

        public void Execute(IServiceProvider serviceProvider)

        {

            ITracingService tracingService = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationServiceFactory servicefactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService client = servicefactory.CreateOrganizationService(context.UserId);

            Entity entity = (Entity)context.InputParameters["Target"];

            //Entity retrContact = (Entity)client.Retrieve("account", entity.Id, new ColumnSet(true));

            //tracingService.Trace(retrContact.ToString());

            //tracingService.Trace(entity.Id.ToString());

            QueryExpression query = new QueryExpression()

            {

                EntityName = "account",

                //Criteria = new FilterExpression(LogicalOperator.And),

                ColumnSet = new ColumnSet(true)

            };

            query.Criteria.AddCondition("accountid", ConditionOperator.Equal, entity.Id);

            EntityCollection Accounts = client.RetrieveMultiple(query);

            QueryExpression query1 = new QueryExpression()

            {

                EntityName = "contact",

                //Criteria = new FilterExpression(LogicalOperator.And),

                ColumnSet = new ColumnSet(true)

            };

            //contactentity = entity.GetAttributeValue<EntityReference>("parentcustomerid");

            query1.Criteria.AddCondition("parentcustomerid", ConditionOperator.Equal, entity.Id);

            //EntityCollection contacts = client.RetrieveMultiple(query1);

            //query.ColumnSet = new ColumnSet("primarycontactid");

            EntityCollection retrieved = client.RetrieveMultiple(query1);

            tracingService.Trace(retrieved.Entities.Count.ToString());

            if(retrieved!=null && retrieved.Entities.Count>0)

            {

                foreach (Entity en in retrieved.Entities)//loop through every record

                {

                    if (en.Attributes.Contains("new\_currency"))

                    {

                        cost = (((Money)en.Attributes["new\_currency"]).Value);

                        total += cost;

                    }

                }

            }

            tracingService.Trace(total.ToString());

            Entity acc = new Entity("account");

            acc.Attributes["accountid"] = Accounts.Entities[0].Id;

            //acc.Attributes["name"] = "keerthana";

            acc.Attributes["new\_currency"] = new Money(total);

            client.Update(acc);

        }

}

    }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**TASK 40**

**Retrieving lookup and updating it to another field**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lookupupdate

{

    public class Class1 : IPlugin

    {

        private string parentacc;

        public void Execute(IServiceProvider serviceProvider)

        {

            ITracingService tracingService = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationServiceFactory servicefactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

            IOrganizationService client = servicefactory.CreateOrganizationService(context.UserId);

            Entity entity = (Entity)context.InputParameters["Target"];

            Entity records = client.Retrieve("account", entity.Id, new ColumnSet(true));

            if (records.Attributes.Contains("parentaccountid"))

            {

                parentacc = ((EntityReference)(records.Attributes["parentaccountid"])).Name;

                Guid accid = ((EntityReference)(records.Attributes["parentaccountid"])).Id;

                // Console.WriteLine(accid);

                tracingService.Trace(parentacc);

                //records.Attributes["new\_desc"] = parentacc;

            }

            Entity acoun1 = new Entity("account");

            acoun1["accountid"] = records.Id;

            acoun1.Attributes["new\_desc"] = parentacc;

            client.Update(acoun1);

            tracingService.Trace(records.ToString());

        }

    }

    }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**TASK 41**

**Normal lookup retrieve**

lookupFieldObject = Xrm.Page.data.entity.attributes.get('new\_account');

    if (lookupFieldObject.getValue() != null) {

   name = lookupFieldObject.getValue()[0].name;

   id=lookupfieldObject.getValue()[0].id;

  entitytype=lookupFieldObject.getValue()[0].entitytype;

}

lookupFieldObject = Xrm.Page.data.entity.attributes.get('new\_cust\_details');

    if (lookupFieldObject.getValue() != null) {

       name = lookupFieldObject.getValue()[0].name;

    }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**TASK 42**

**create contact**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Xrm.Sdk;

namespace Plugin\_sample1

{

    public class Class1 : IPlugin

    {

        public void Execute(IServiceProvider serviceProvider)

        {

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            if(context.InputParameters.Contains("Target"))

            {

                if(context.InputParameters["target"]is Entity)

                {

                    try

                    {

                        Entity createcontact = new Entity("contact");

                        createcontact["fname"] = "keer";

                        createcontact["lname"] = "s";

                        if(context.OutputParameters.Contains("id"))

                        {

                            createcontact["parentcustomerid"] = new EntityReference { Id = new Guid(context.OutputParameters["id"].ToString())};

                        }

                        IOrganizationServiceFactory servicefactory = (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));

                        IOrganizationService client = servicefactory.CreateOrganizationService(context.UserId);

                        client.Create(createcontact);

                    }

                    catch(Exception e)

                    { }

                }

            }

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**TASK 43**

**retrieve elements**

Xrm.Page.getAttribute('name').getValue();

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 44

**Custom workflow using plugin**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Workflow;

using System;

using System.Activities;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace customworkflowprocess

{

    public class Class1 : CodeActivity

    {

        protected override void Execute(CodeActivityContext context)

        {

            try

            {

                IWorkflowContext wfContext = context.GetExtension<IWorkflowContext>();

                IOrganizationServiceFactory serviceFactory = context.GetExtension<IOrganizationServiceFactory>();

                IOrganizationService service = serviceFactory.CreateOrganizationService(wfContext.InitiatingUserId);

                var guid = wfContext.PrimaryEntityId.ToString();

                var acc = new Entity("account",new Guid(guid));

                acc["new\_desc"] = "updated";

                service.Update(acc);

            }

            catch (Exception)

            {

            }

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 45

**customworkflow using parameters counting words**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using Microsoft.Xrm.Sdk.Workflow;

using System;

using System.Activities;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CustomworkflowParameters

{

    public class Class1 : CodeActivity

    {

        [RequiredArgument]

        [Input("Input Text")]

        public InArgument<string> InputText { get; set; }

        [Output("Word Count")]

        public OutArgument<int> CountOfWords { get; set; }

        protected override void Execute(CodeActivityContext context)

        {

            this.CountOfWords.Set(context,this.InputText.Get<string>(context).Split(new char[] { ' ', '\r', '\n' },StringSplitOptions.RemoveEmptyEntries).Length);

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 46

**customworkflow using parameters setting account name to the another field**

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Query;

using Microsoft.Xrm.Sdk.Workflow;

using System;

using System.Activities;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace customworkflowplugin

{

    public class Class1 : CodeActivity

    {

        [Input("Account name")]

        public InArgument<string> Accountname { get; set; }

        [Output("id")]

        public OutArgument<string> id { get; set; }

        protected override void Execute(CodeActivityContext context)

        {

            IWorkflowContext wfContext = context.GetExtension<IWorkflowContext>();

            IOrganizationServiceFactory serviceFactory = context.GetExtension<IOrganizationServiceFactory>();

            IOrganizationService service = serviceFactory.CreateOrganizationService(wfContext.InitiatingUserId);

            var acc = Accountname.Get<string>(context);

            QueryExpression query = new QueryExpression("account");

            query.ColumnSet = new ColumnSet(true);

            query.Criteria.AddCondition("name", ConditionOperator.Equal, acc);

            EntityCollection retr = service.RetrieveMultiple(query);

            var output = retr.Entities[0].Attributes["name"].ToString();

            this.id.Set(context, output);

           // this.id.Set(context, (account.Attributes["new\_desc"]));

        }

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 47

**retrieve attribute values in js**

Xrm.Page.getAttribute(“CRMFieldSchemaName”).getValue();

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 48

**Link for debugging workflow**

<https://ssharmacrm.wordpress.com/2015/06/22/how-to-debug-a-custom-workflow-activity-in-dynamics-crm/>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 49

Link for quick find search

<https://www.powerobjects.com/blog/2015/10/22/how-to-add-a-field-to-quick-find-in-crm-2015/>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 50

**Locking particular field if another field has some values using js**

var x=Xrm.Page.getAttribute("orb\_projectnumber").getValue();

if(x!=null)

{

Xrm.Page.getControl("transactioncurrencyid").setDisabled(true);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TASK 51

Preimage

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Xrm.Sdk;

using Microsoft.Xrm.Sdk.Client;

namespace TestPlugin

{

    public class ImageDemo : IPlugin

    {

        public void Execute(IServiceProvider serviceProvider)

        {

            #region Setup

            IPluginExecutionContext context = (IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

            IOrganizationService service = ((IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory))).

            CreateOrganizationService(new Guid?(context.UserId));

            ITracingService tracingService = (ITracingService)serviceProvider.GetService(typeof(ITracingService));

            #endregion

            if ((context.InputParameters.Contains("Target")) && (context.InputParameters["Target"] is Entity) && context.MessageName.ToUpper() == "UPDATE")

            {

                Entity entity = (Entity)context.InputParameters["Target"];

                if (entity.LogicalName.ToLower() != "incident") return;

                try

                {

                    Entity preImage = (Entity)context.PreEntityImages["Image"];

                    if (preImage.Contains("prioritycode"))

                    {

                        if (preImage.GetAttributeValue<OptionSetValue>("prioritycode").Value == 3)//if Pre image have priority = Low

                        { //if Current entity have priority = High

                            if (entity.Contains("prioritycode") && entity.GetAttributeValue<OptionSetValue>("prioritycode").Value == 1)

                            {

                                throw new Exception("Changing priority form Low to High is now allowed");

                            }

                        }

                    }

                }

                catch (Exception ex)

                {

                    throw new Exception("" + ex.Message);

                }

            }

        }

    }

}

**Task 52**

**Changing url field**

Function GetSet(executionContext)

{

Var formContext=executionContext.getFormContext();

Var website=formcontext.getAttribute(“website”).getValue();

Alert(website);

formContext.getAttribute(“website”).setValue(“http://newvalue.com”);

website=formContext.getAttribute(“websiteurl”).getvalue();

alert(website);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_