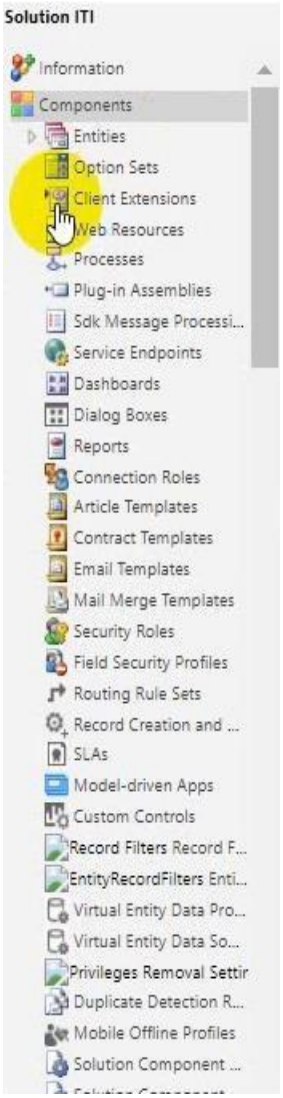


When we open solution:

Entities: all the entities and includes the forms, views, fields.

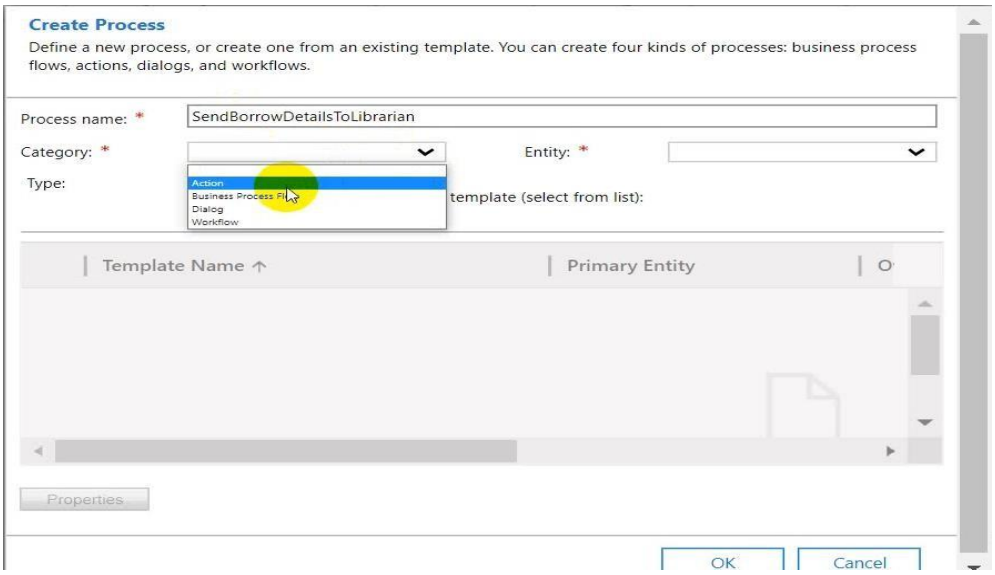
Option sets: global option sets to use any where

Client extension: Application ribbons, side map.

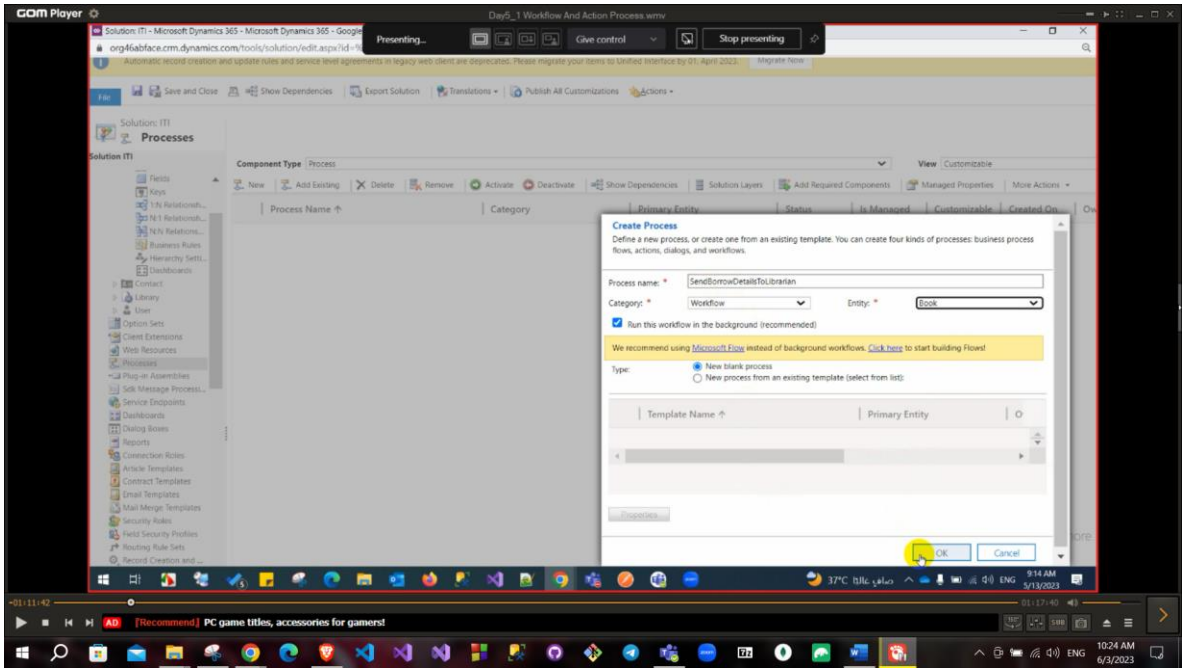
Web resources: can add in static file on it, like HTML, JS or any image or file.

Processes, plugins and SDK message processions

Processes: four types of process(Action, Workflow, Business Process, Dialog)

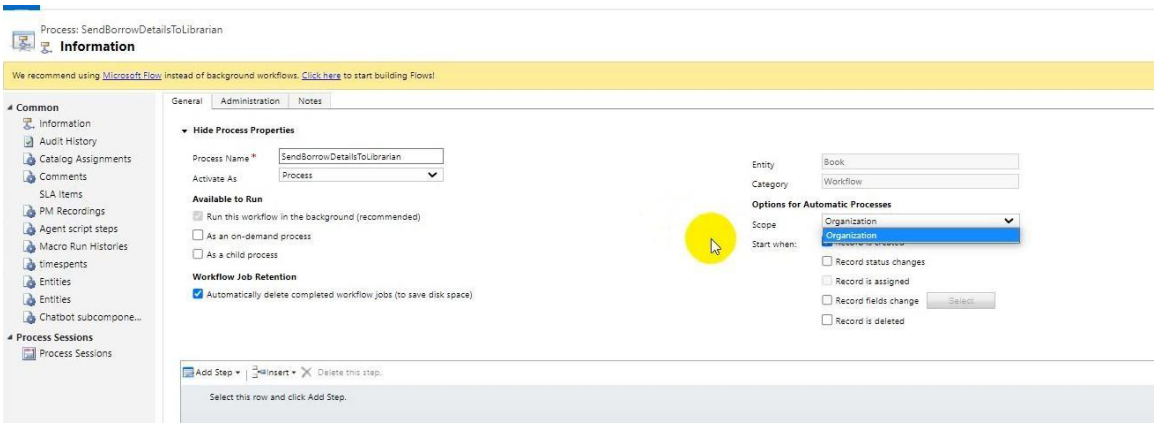


Workflow: Process to create workflow to execute pipeline (like tosend email to librarian if anyone borrow a book).



In process you can select one of these categories: **Workflow** like a pipeline

Then select entity which you want to execute this process on it based on condition.

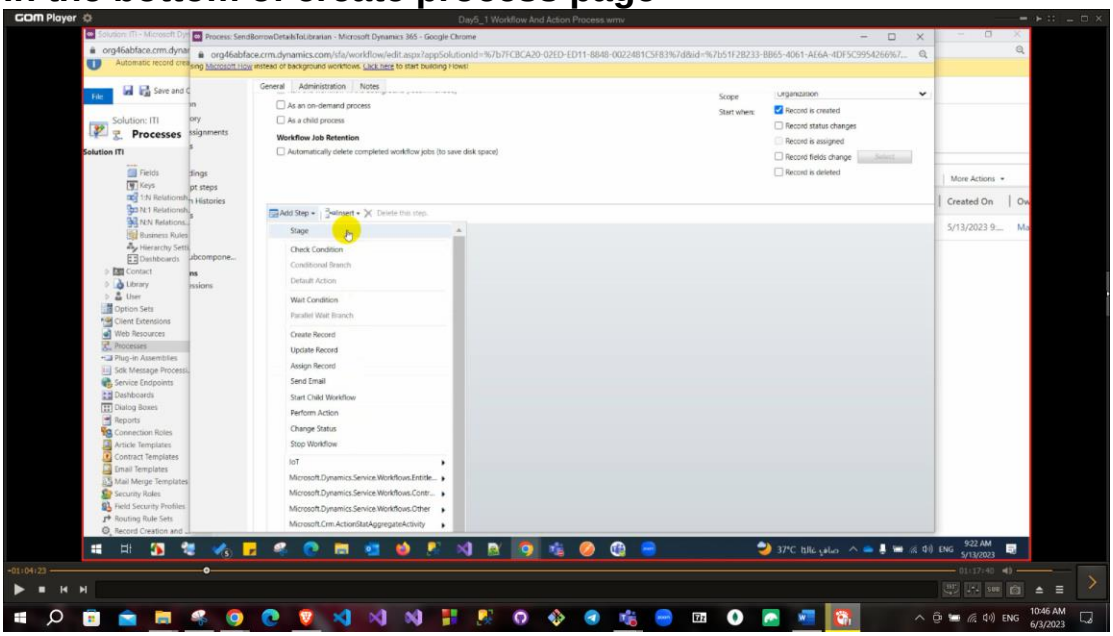


Scope: Organization or user (if you select user then this workflow will be executed for user who is owner of this entity else if you select organization then this workflow will be executed for all users).

Also, there is a check box for **start when** (when record created, status changed, record fields change or record is deleted).

Available to run check box as an on-demand process to enable runningthis workflow manually or as background process or As a child process to enable that you can call this workflow to run insideanother workflow

In the bottom of create process page

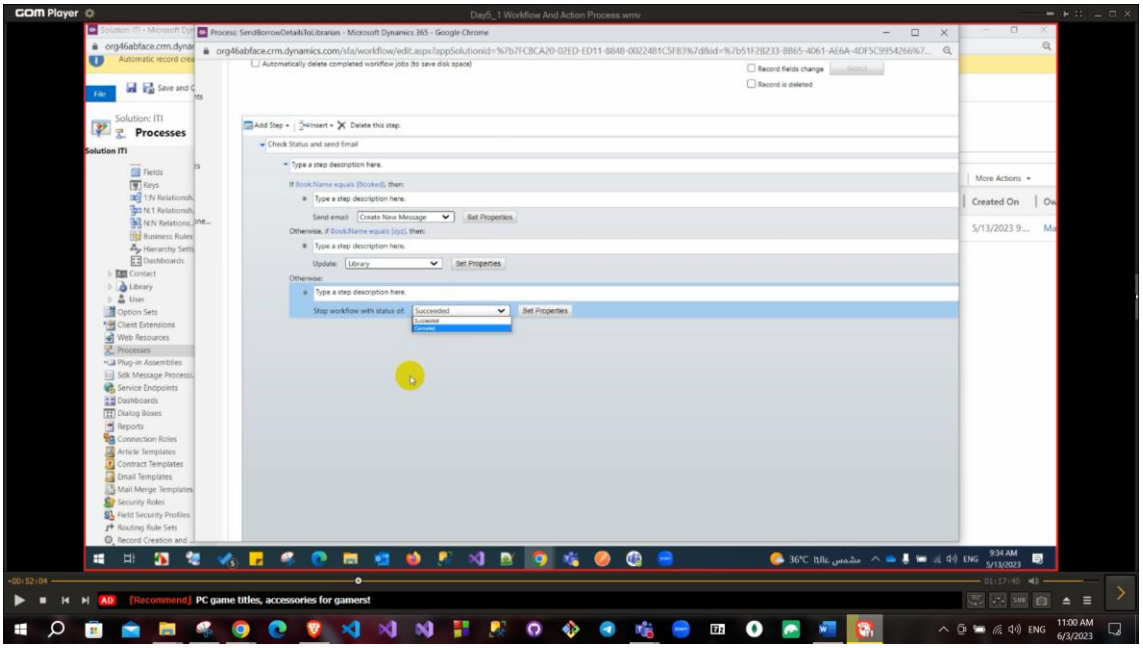


Stage: to add some steps into one stage.

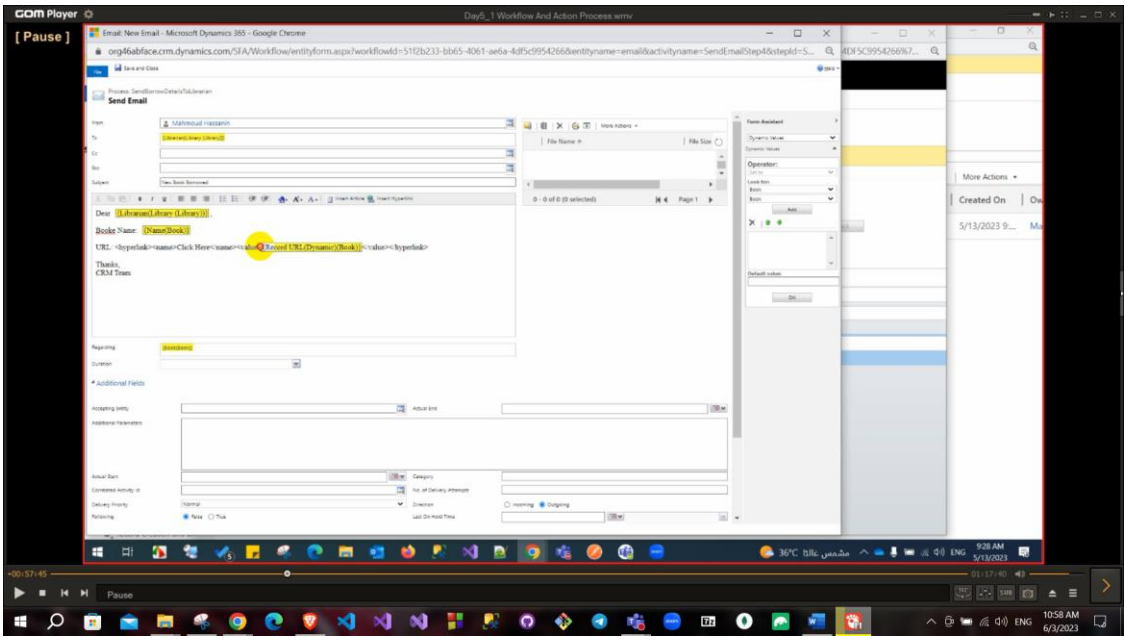
Check condition: to add condition or validation.

Wait condition: to add condition to continuerunning this workflow after this condition’s validation is true

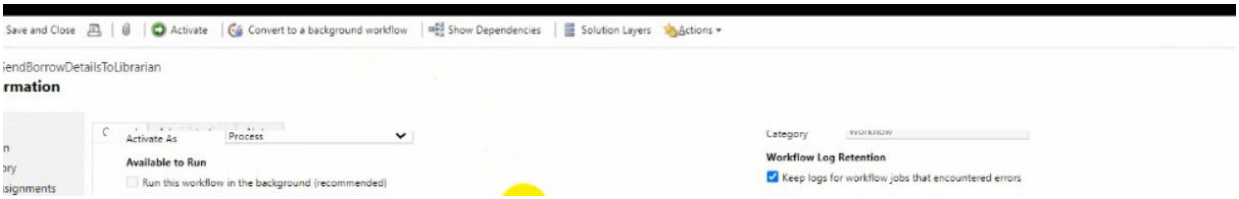
Remain choices are logical from their names.

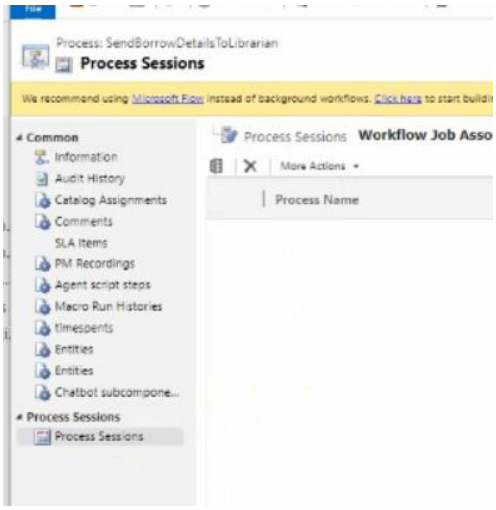


When we click on set properties the template of the email open



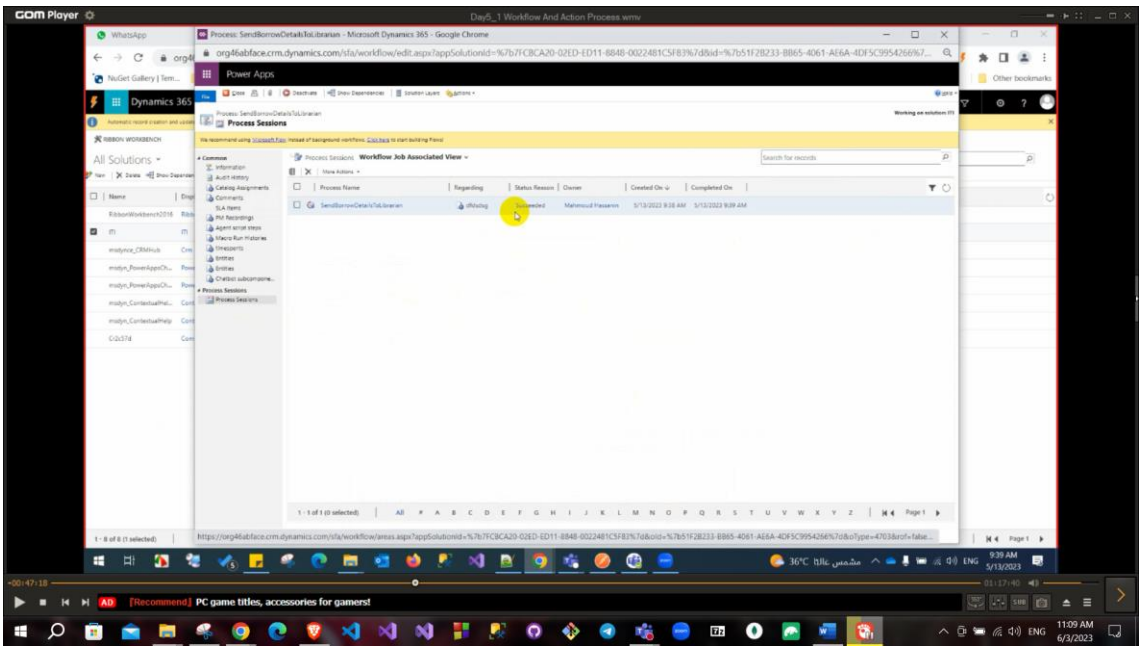
There is a choice beside Activate button called Convert to a real-time workflow to run this workflow synchronous on other hand running it in background is running asynchronously whichis recommended to use because if it failed or success no action needed.





To track execution of specific workflow: go to process in solution thenselect needed workflow from process sessions

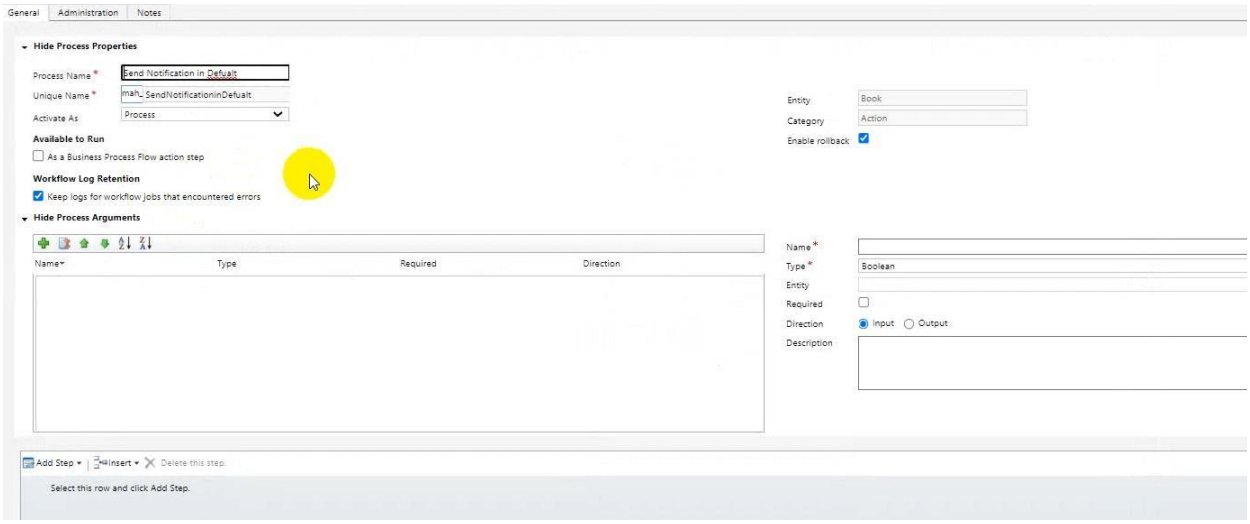
When you create new record you will find that in process sessions



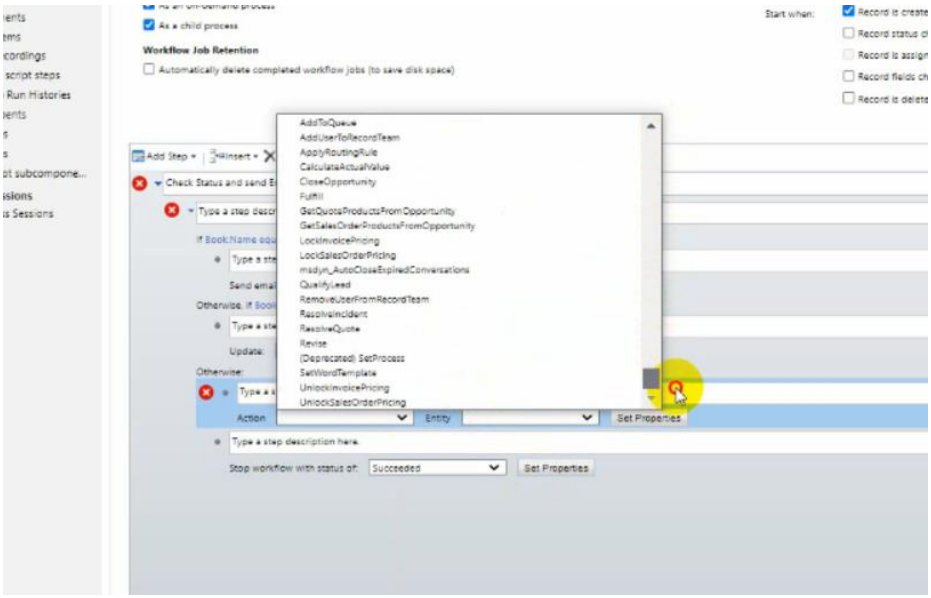
If this workflow failed due to some missing data (that included in it's condition) it's status will be **waiting** so, you can fill missing data and retry to execute workflow.

Action

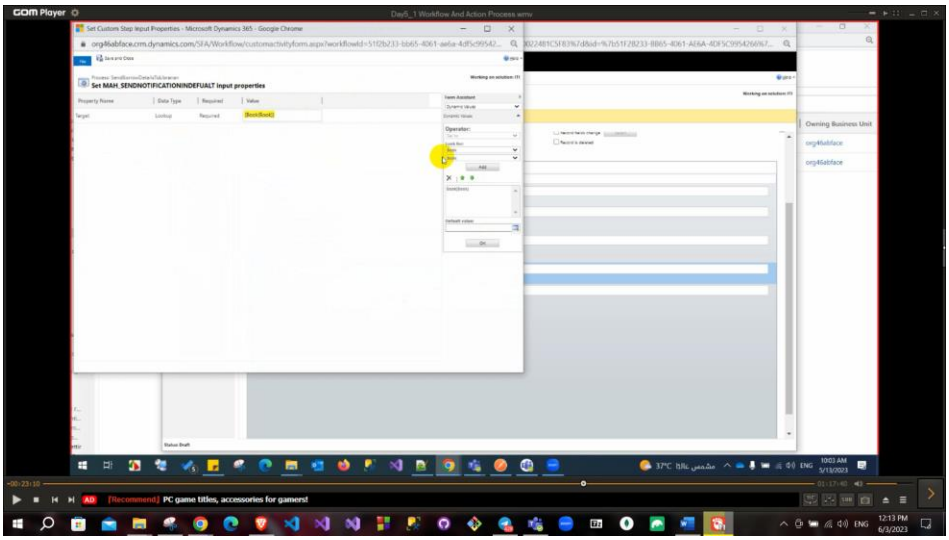
Action is like workflow but you can call this action from JS or from C# (using API) and this action has an input and output



You can use this action in workflow also you can set input or output for action (output may be Boolean to return true or false when run this action).



When you use action in workflow you will need to add target for this action.



You can do action of workflow using JS but workflow is faster also you can run workflow manually by open the entity that related to workflow then from ribbon select flow then select your workflow.

When you use Js vs workflow vs action

If you have the choice between JS and workflow: workflow will be better than JS because it is faster and running server side and user friendly than JS also you can track workflow execution from process session.

Workflow as a trigger because it run when record created or updated or it's status changed.

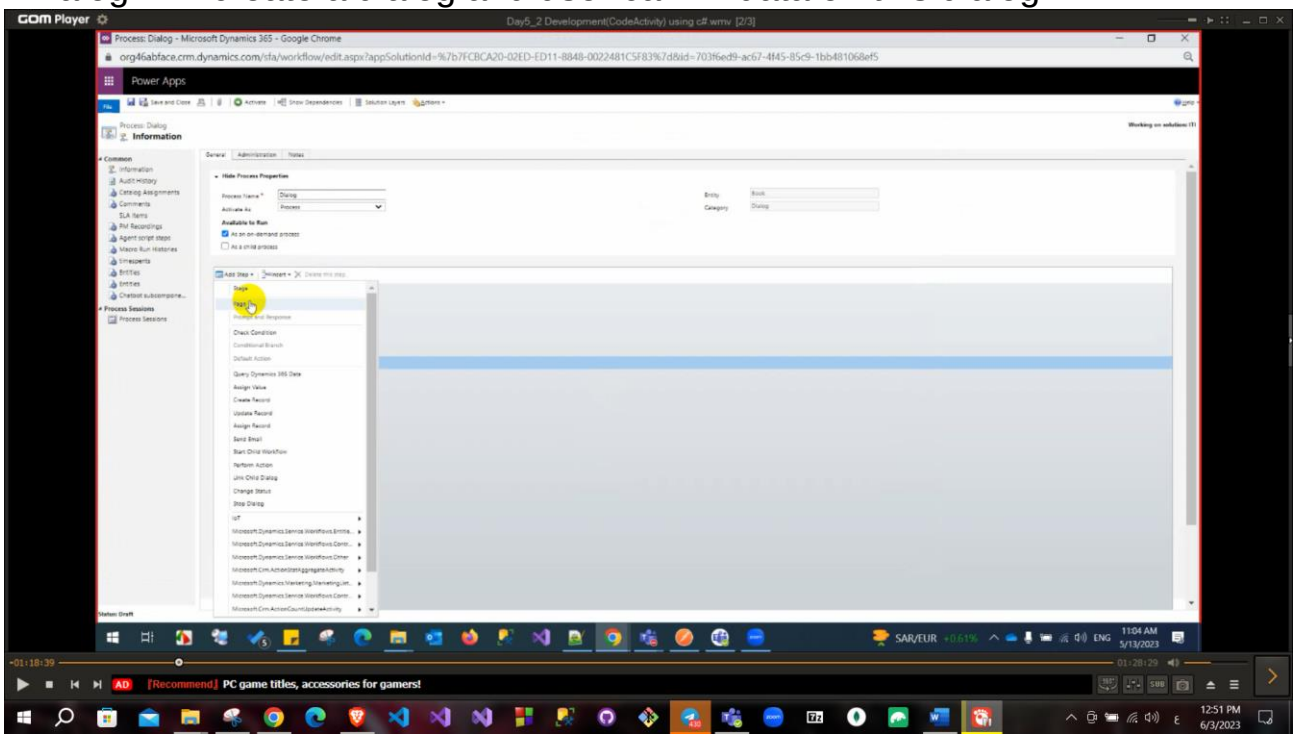
But action has an advantage that it has input or output so you can run it using JS and use it's return also you can use action inside workflow.

Action has not event to run so if you create all step of workflow and put it into action it will not run because action has no event to run when it occurs like workflow.

Very important question when you run action you need to pass argument called target to inform this action the record that it will run on it.

Dialog

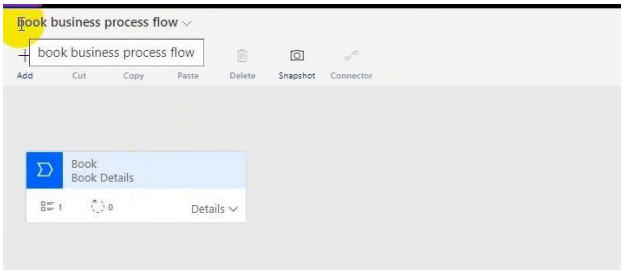
Dialog: will create a dialog and user can fill data on this dialog



You can add prompt or add HTML page.

Business process flow

Like sale process (appears in lead): you can add workflow as stages and run



Workflow process strats automatic

You can call action from JS.

```
function RunAction() {

    var Id = Xrm.Page.data.entity.getId().replace('{', '').replace('}', '');
    var serverURL = Xrm.Page.context.getClientUrl();
    var fromUser = {};
    fromUser.systemuserid = "9C9CA531-E021-E911-A9BC-000D3A1B913D";
    var toUser = {};
    toUser.systemuserid = "6E8B062B-E021-E911-A9BC-000D3A1B913D";

    var data = {
        "FromEmail": fromUser,
        "ToEmail": toUser
    };

    var query = "incidents(" + Id + ")/Microsoft.Dynamics.CRM.new_RouteBugtoDevelope
    var req = new XMLHttpRequest();
    req.open("POST", serverURL + "/api/data/v9.1/" + query, true);
    req.setRequestHeader("Accept", "application/json");
    req.setRequestHeader("Content-Type", "application/json; charset=utf-8");
    req.setRequestHeader("OData-MaxVersion", "4.0");
    req.setRequestHeader("OData-Version", "4.0");
    req.onreadystatechange = function () {
        if (this.readyState == 4 /* complete */) {
            req.onreadystatechange = null;
            if (req.status >= 200 && req.status <= 300) {
                var data = JSON.parse(this.response);
            } else {
                var error = JSON.parse(this.response).error;
            }
        }
    };
    req.send(window.JSON.stringify(data));
}
```

In the code, we are creating objects for the to and from users, as these are entityreferences in our process:

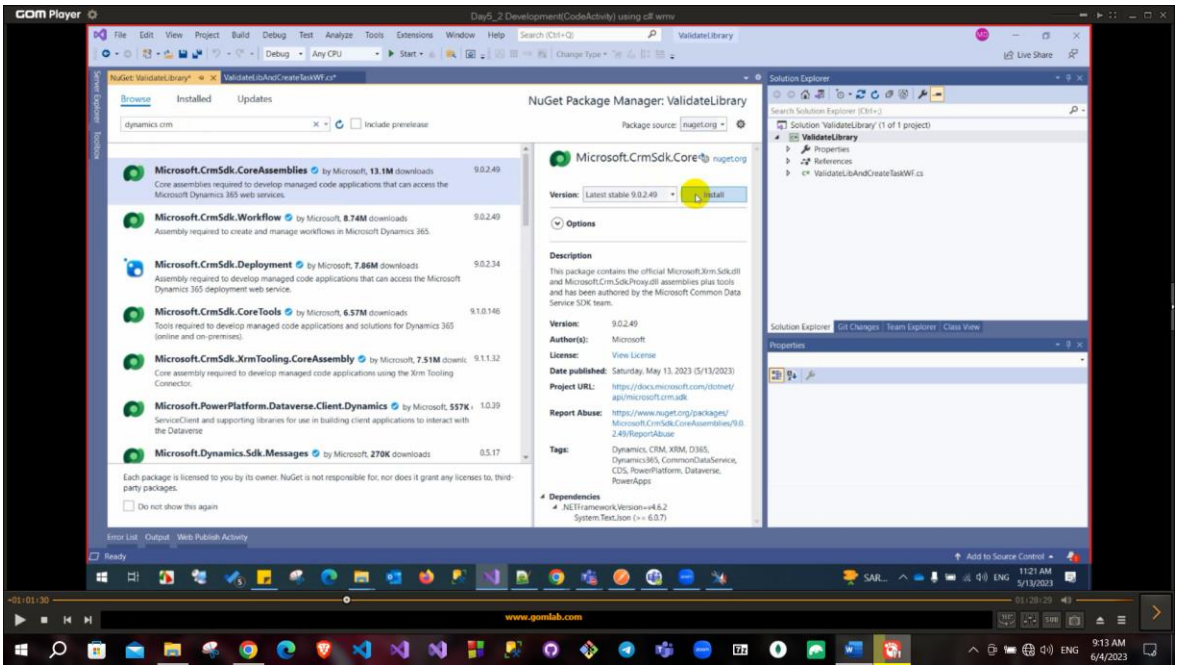
Development

Custom workflow

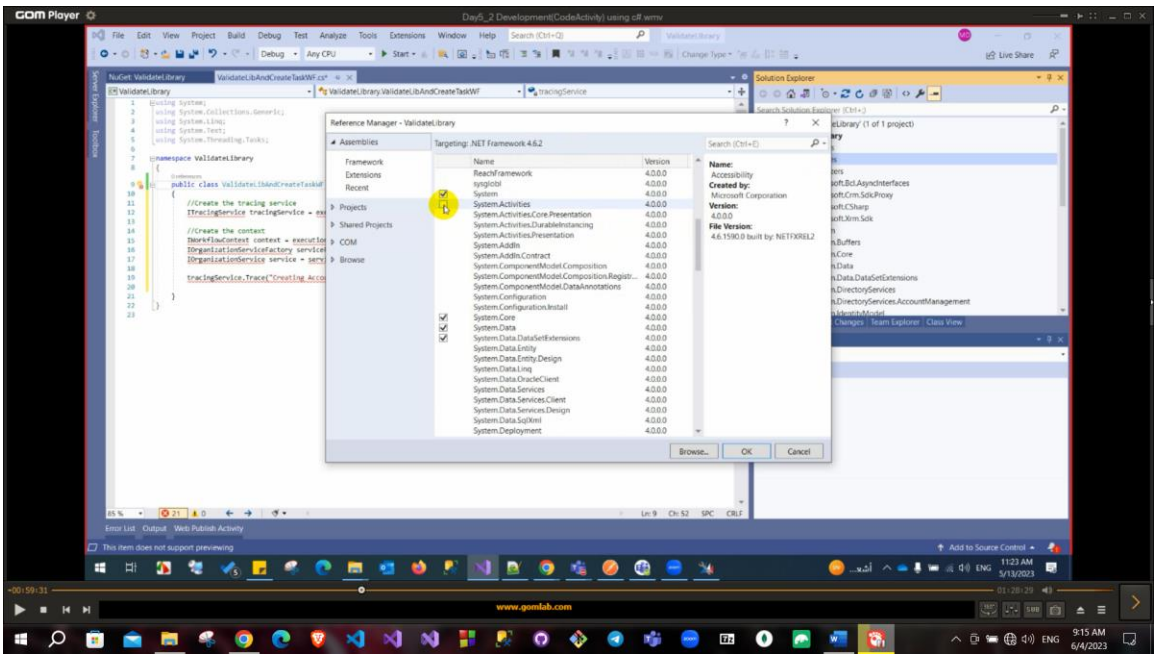
Most stable project is class library .NET framework (4.6.2)

Add class library project

Add Microsoft.CrmSdk.CoreAssemlies, Microsoft.CrmSdk.Workflow



Also, you need to add reference from System.Activities from Assemblies to read codeActivity class



Most important namespace are Microsoft.Crm.Sdk.Proxy and Microsoft.Xrm.Sdk

```
0 references
public class ValidateLibAndCreateTaskWF : CodeActivity
{
    0 references
    protected override void Execute(CodeActivityContext executionContext)
    {
        //Create the tracing service
        ITracingService tracingService = executionContext.GetExtension<ITracingService>();

        //Create the context
        IWorkflowContext context = executionContext.GetExtension<IWorkflowContext>();
        IOrganizationServiceFactory serviceFactory = executionContext.GetExtension<IOrganizationServiceFactory>();
        IOrganizationService service = serviceFactory.CreateOrganizationService(context.UserId);

        tracingService.Trace("Creating Validate Lib");


        tracingService.Trace("Validation done");
        tracingService.Trace("start creating task");

    }
}
```

First you must implement CodeActivity interface and implement Execute method. This called code activity also called custom workflow

ITracingServices to trace if there is an error.

IWorkflowContext to get your object context that you will use later like primaryControl in JS.

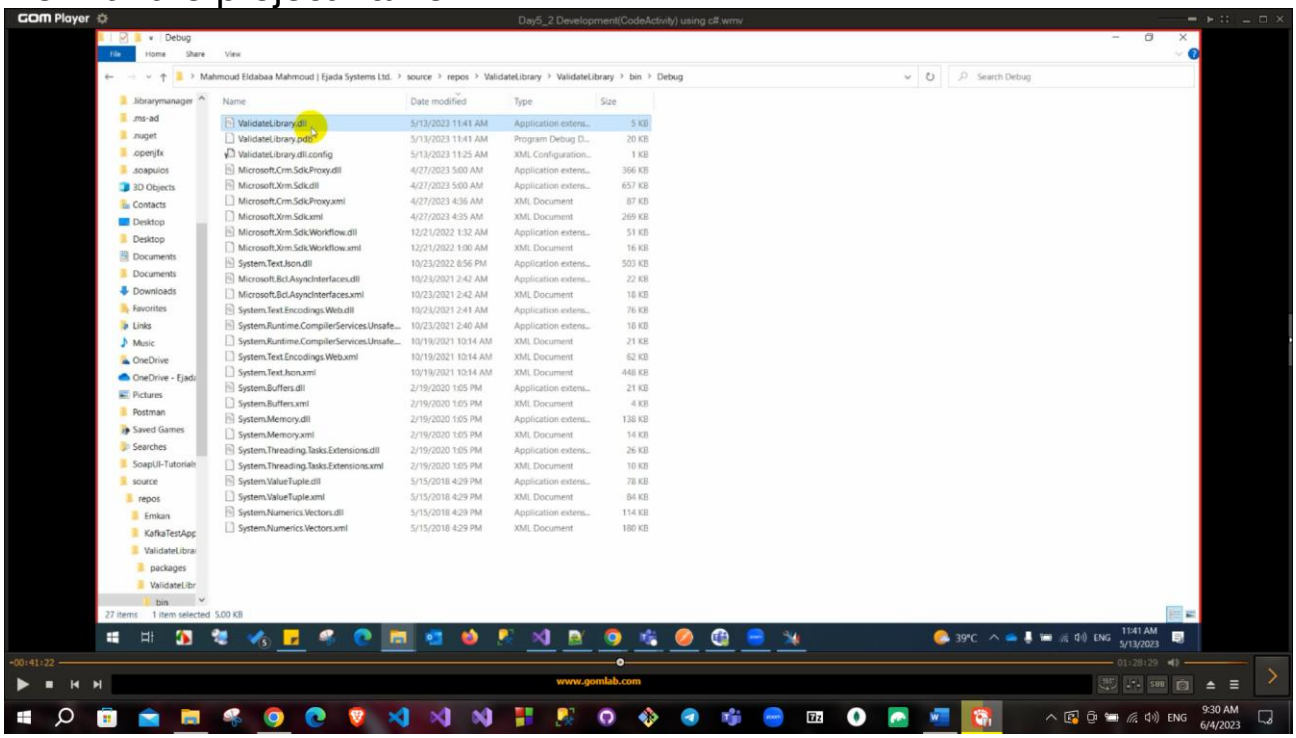
IOrganizationservices to get your services object that you will use in calling API, the services object has many methods like retrieve, retrieveMultiple, create, update, execute, ……

To add new record: first step you need to know name of entity in database it may be different from this display name (to know name of entity and the field of this entity) go to customization then customize and system then search for required entity then open it and see its name and fields

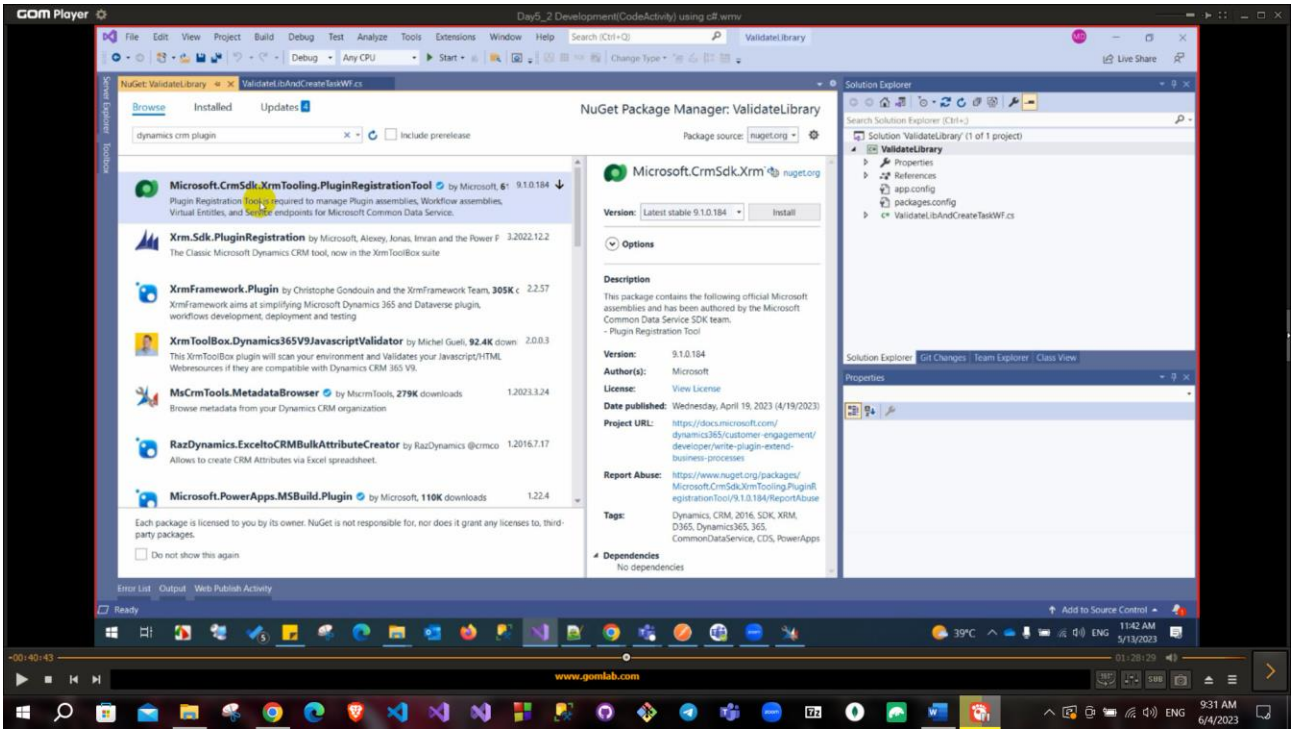
```
//tracingService.Trace("Creating Validate Lib");

//tracingService.Trace("Validation done");
tracingService.Trace("start creating task");
Entity taskEntity = new Entity("task");
taskEntity["subject"] = "Task 1";
taskEntity["description"] = "Task  Description";
taskEntity["regardingobjectid"] = new EntityReference(context.PrimaryEntityName,context.PrimaryEntityId);


service.Create(taskEntity);
```

This code creates task that have the regrading to the entity I am on and will appear on its notes.

Build the solution and go to bin folder then debug and you will find a dll file with the project name



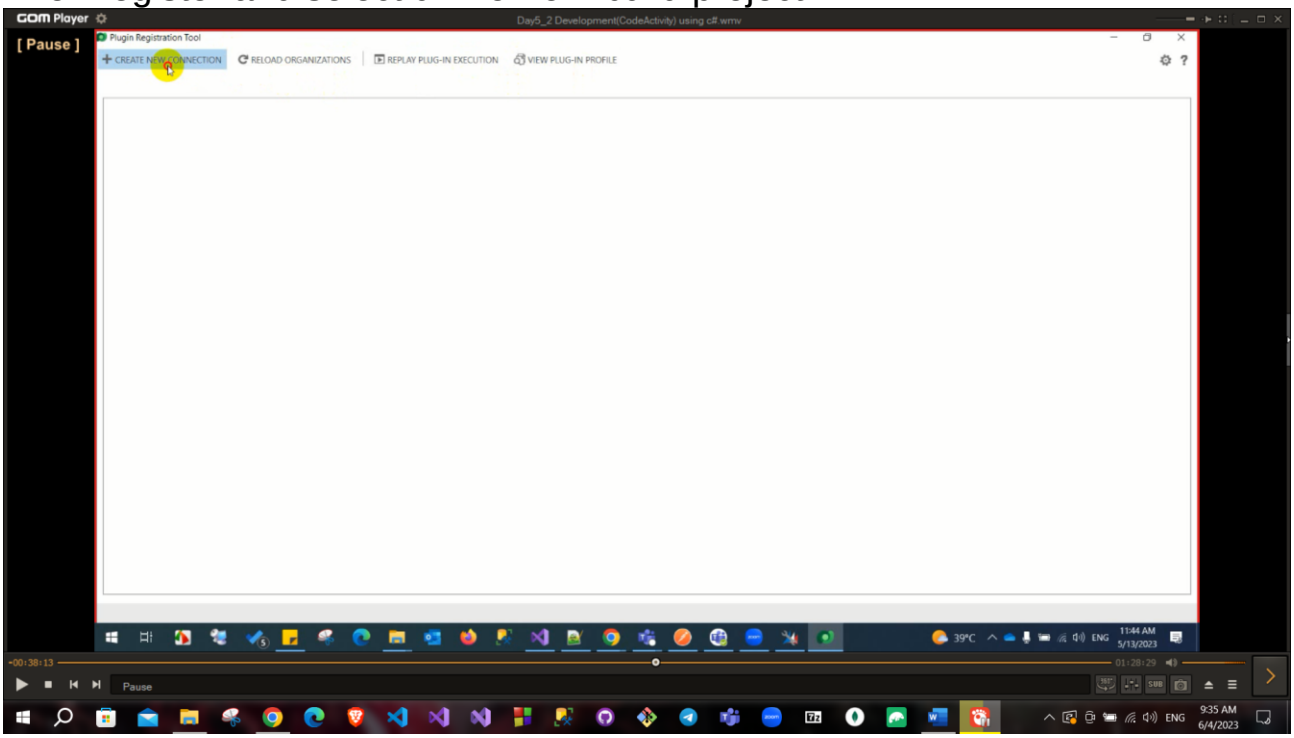
Plugin registration tool from NuGet manager to add dll into environment.



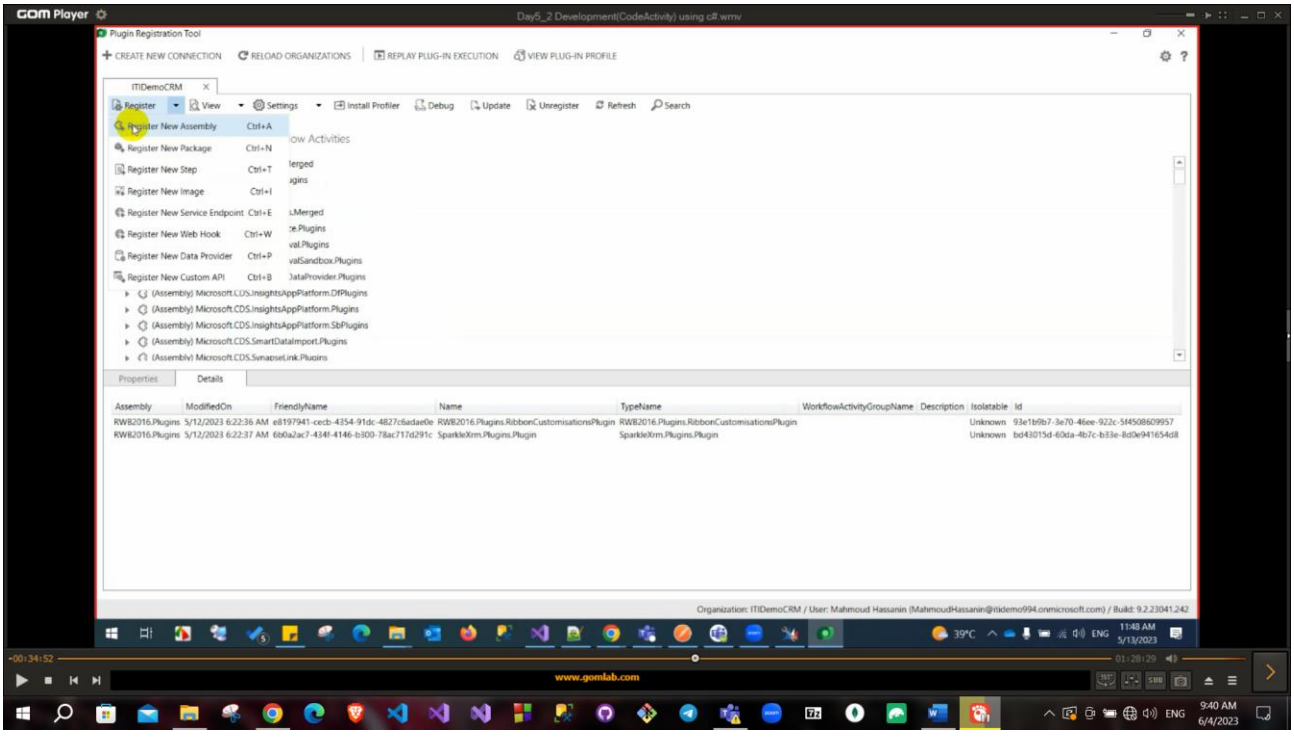
You need to add strong key before this step in project from properties then signing add key to secure publishing files.

Open this plugin from packages folder and fill login information.

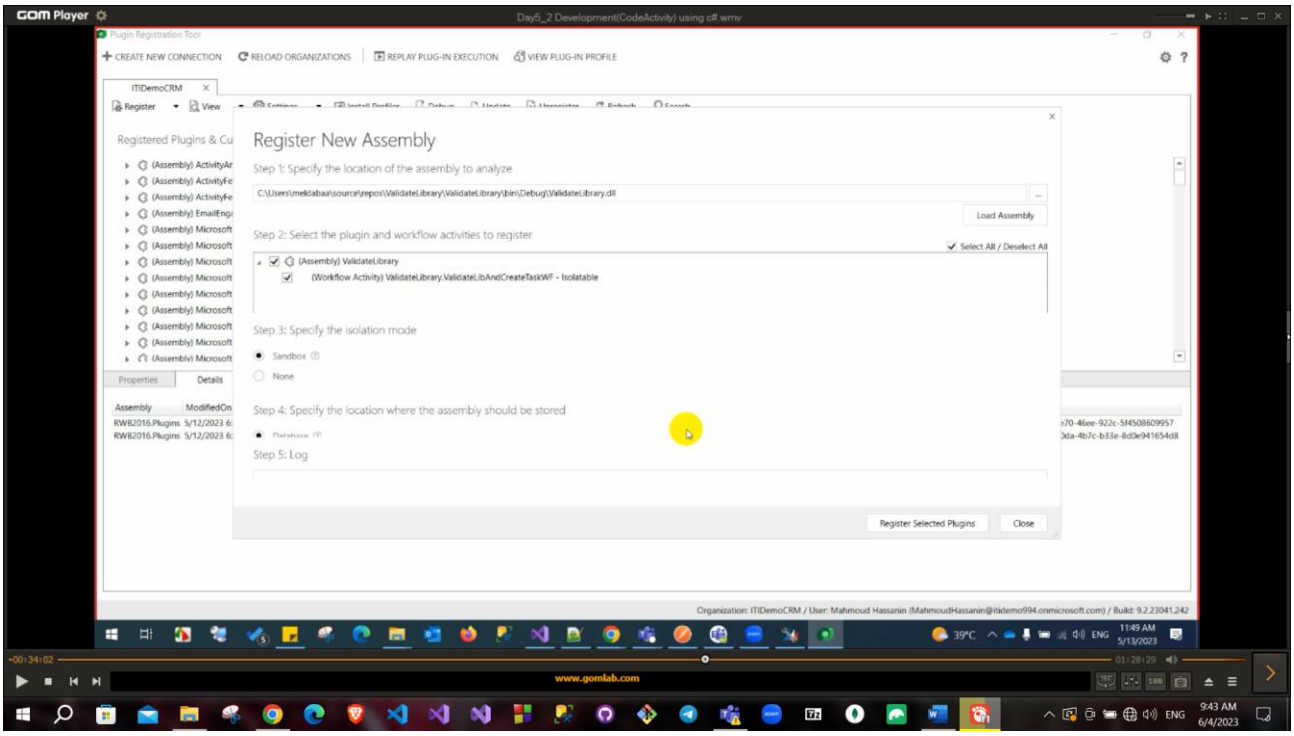
Then register and select dll file from build project.

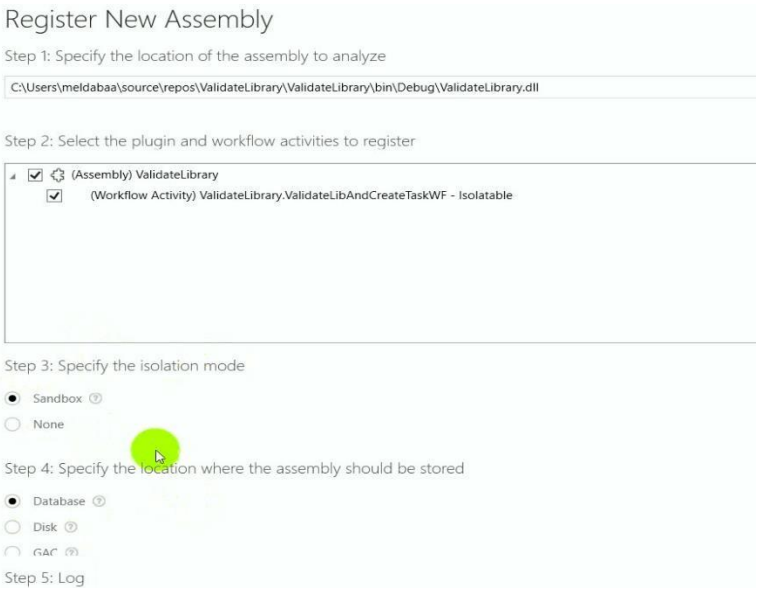


Then register the assembly



Choose the dll





Sandbox: more secure and put some restriction

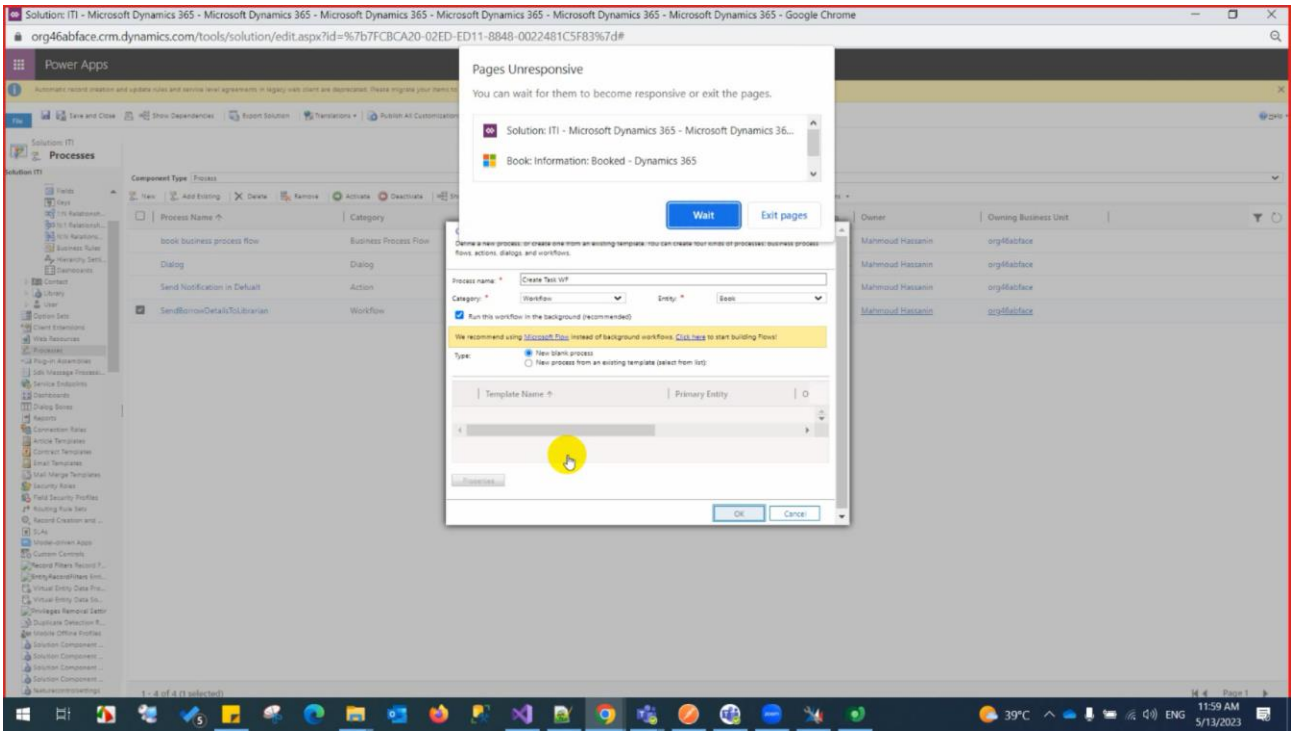
Database: this dll will be stored in database.

Disk: dll will be stored in path in server.

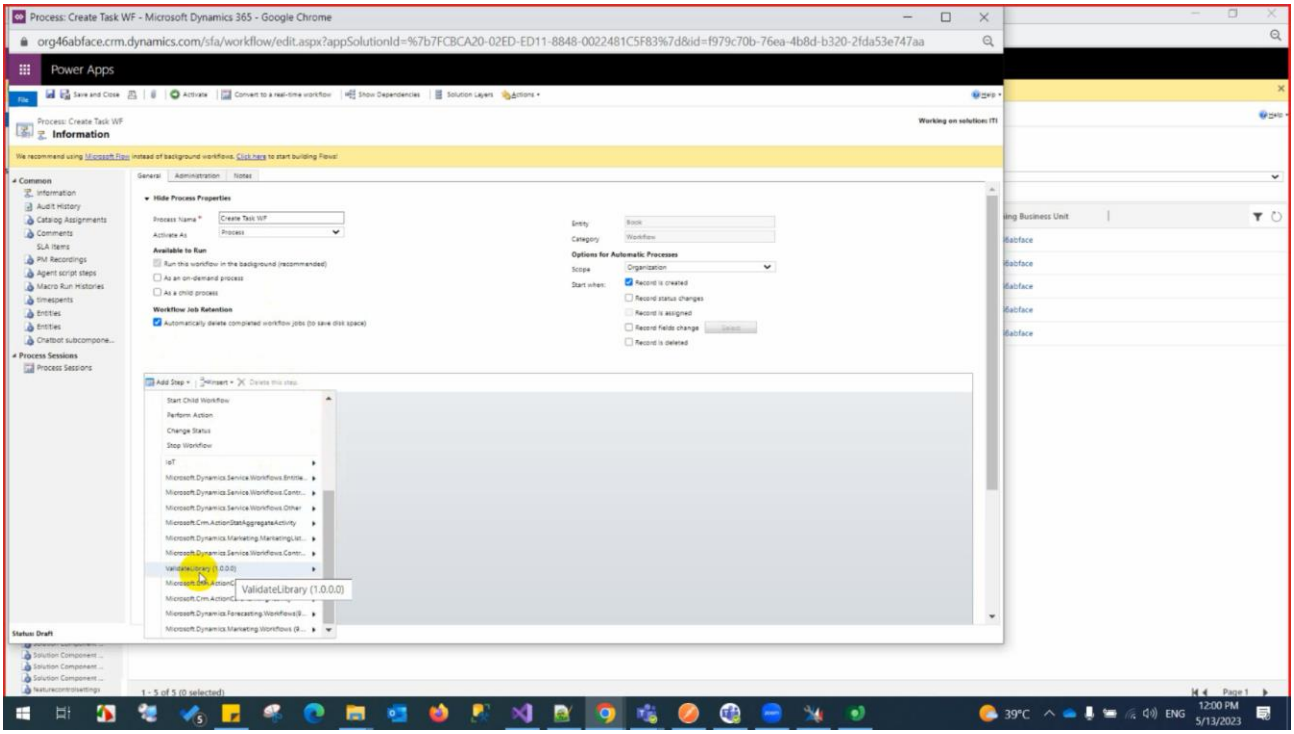
GAG: will be stored in GAG

<u>Most used in database.</u>

After adding assembly files now, now you need to add this custom workflow so create a workflow process normally from crm



Then from add step choose the custom workflow you created



Then activate

Now if any book created a task should be created on it

If you want to add parameters in workflow use [Input], [Output] attributes

To define parameter public InArgument<string> inputname {get;set;}

```
public class ValidateLibAndCreateTaskWF : CodeActivity
{
    //pass book name as input
    [RequiredArgument]
    [Input("Book Name")]
    1 reference
    public InArgument<string> BookName { get; set; }

    //return bool
    [Output("Is Success")]
    0 references
    public OutArgument<bool> IsSuccess { get; set; }
    0 references
    protected override void Execute(CodeActivityContext executionContext)
    {
        //Create the tracing service
        ITracingService tracingService = executionContext.GetExtension<ITracingService>()
```

To get parameter from user:

var name = inputname.Get(executionContext);

```
var bookName = BookName.Get(executionContext);
//tracingService.Trace("Creating Validate Lib");
```

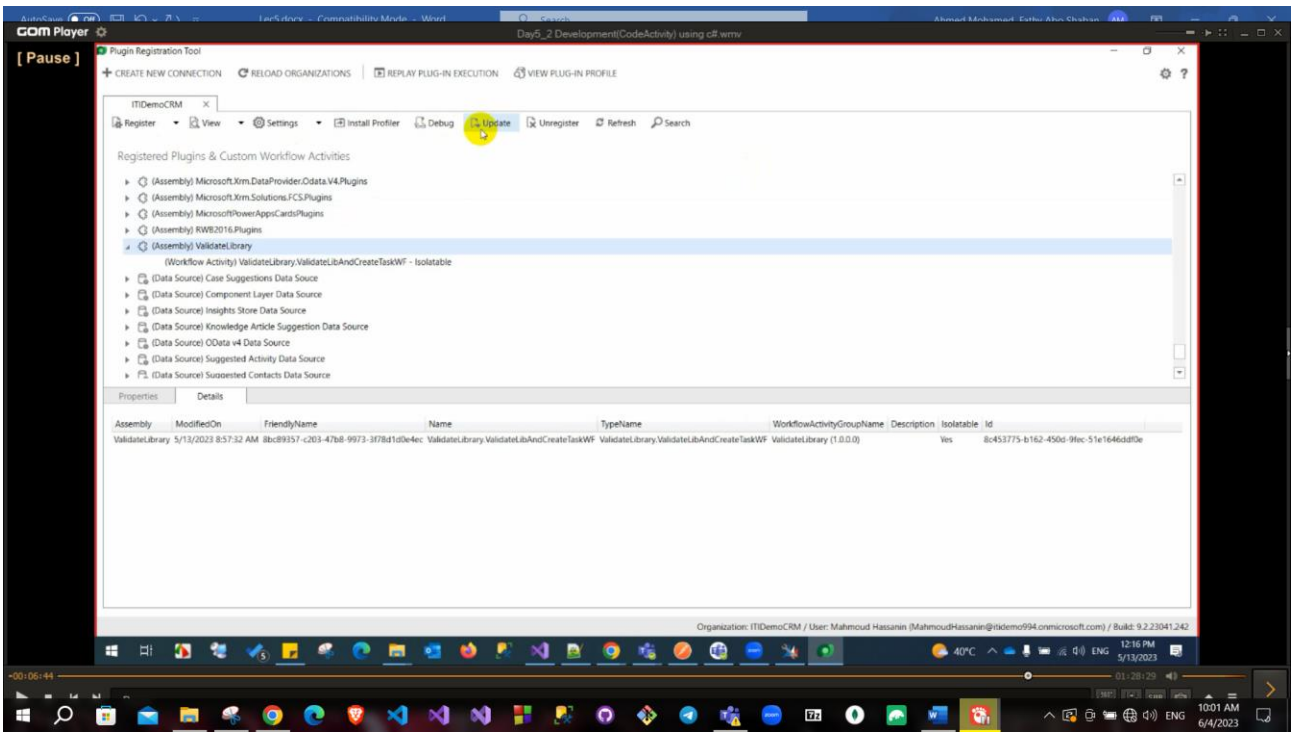
To ser parameter

outputname.Set(executionContext, value);

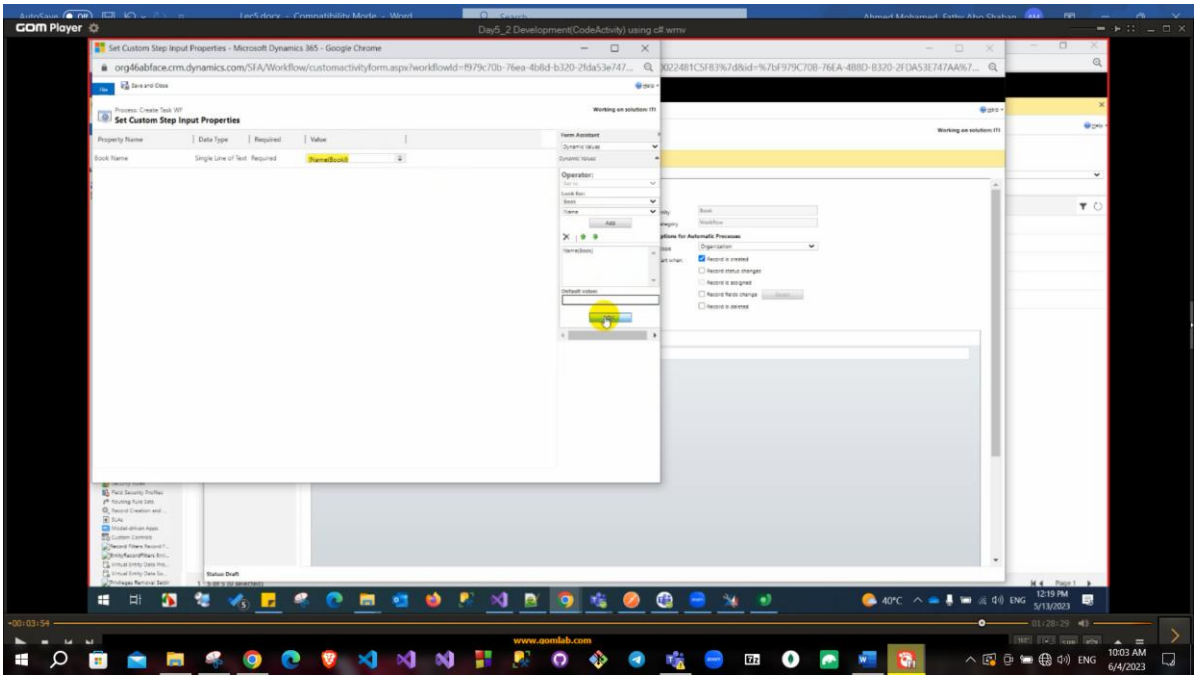
```
service.Create(taskEntity);

IsSuccess.Set(executionContext, true);
```

```
//pass book name as input
[RequiredArgument]
[Input("Book Name")]
1 reference
public InArgument<string> BookName { get; set; }

//return bool
[Output("Is Success")]
1 reference
public OutArgument<bool> IsSuccess { get; set; }
0 references
protected override void Execute(CodeActivityContext executionContext)
{
    //Create the tracing service
    ITracingService tracingService = executionContext.GetExtension<ITracingService>();

    //Create the context
    IWorkflowContext context = executionContext.GetExtension<IWorkflowContext>();
    IOrganizationServiceFactory serviceFactory = executionContext.GetExtension<IOrganizationServiceFactory>();
    IOrganizationService service = serviceFactory.CreateOrganizationService(context.UserId);



    var bookName = BookName.Get(executionContext);
    //tracingService.Trace("Creating Validate Lib");

    //tracingService.Trace("Validation done");
    tracingService.Trace("start creating task");
    Entity taskEntity = new Entity("task");
    taskEntity["subject"] = bookName;//"Task 1";
    taskEntity["description"] = "Task  Description";
    taskEntity["regardingobjectid"] = new EntityReference(context.PrimaryEntityName,context.PrimaryEntityId);


    service.Create(taskEntity);
```

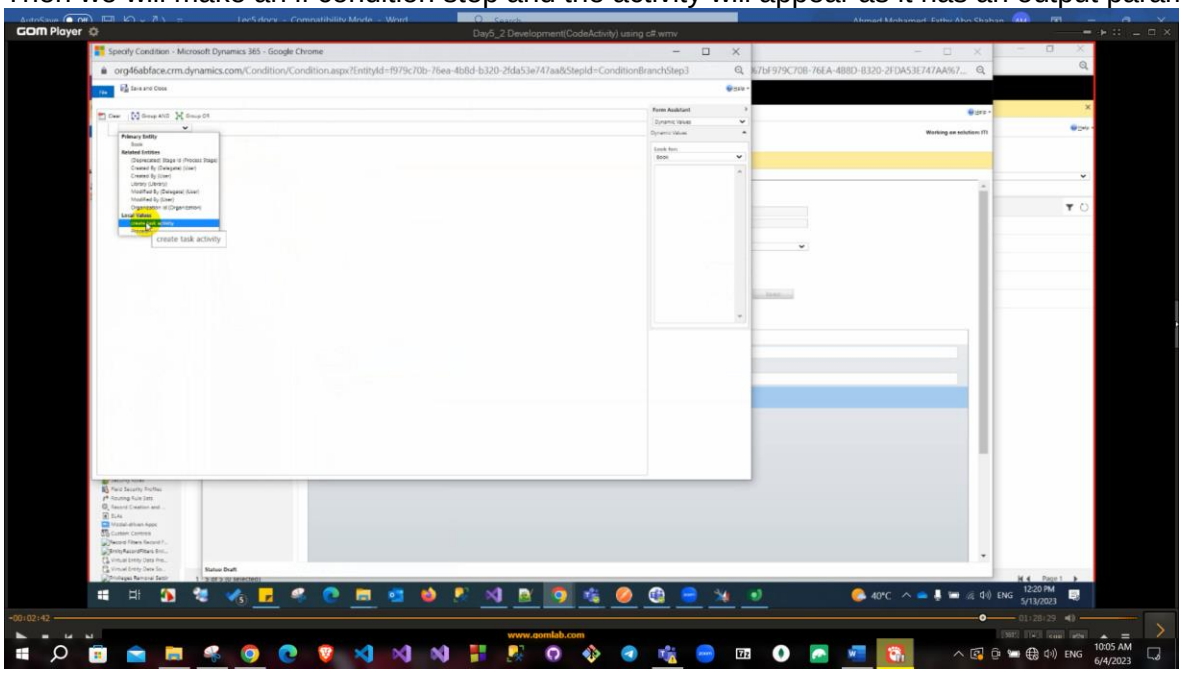
If you make changes in dll build and from plugin registration tool click on update on the assembly file



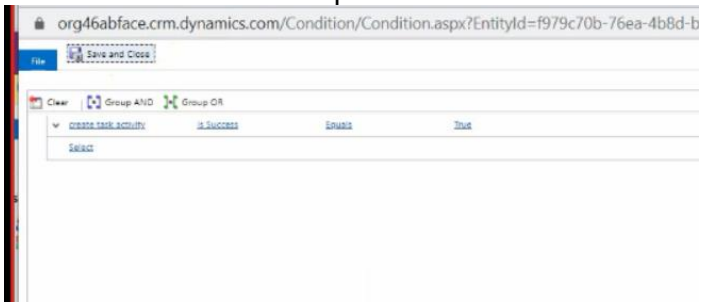
Then go again to the workflow and add parameters



Then we will make an if condition step and the activity will appear as it has an output parameters



Check if the isSuccess equals true



Plugin

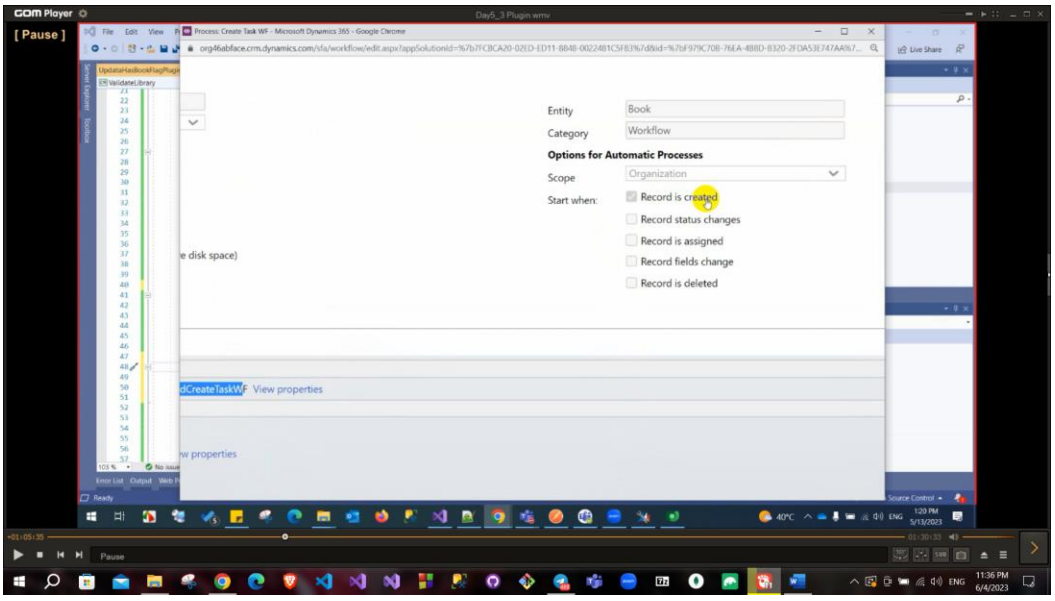
Plugin is similar to code activity but has more options in the start when (very importantquestion to search)

Plugin more generic than custom workflow (code activity)

Plugin has more events than CA like grant share or assign also no need to add plugin in workflow to run

But code activity is more simplified than plugin like GUI events (send email) is easier to make using workflow than making it using plugin.

(Record created, record status changes, record is assigned, record fields change and record is deleted) these are only workflow events



To create Plugin, you need to implement Iplugin interface

It is the same as code activity has context and service

The entity is the book and the lookup field will be gotten by the libraryId and then use the retrieve and pass the name and id to get the library

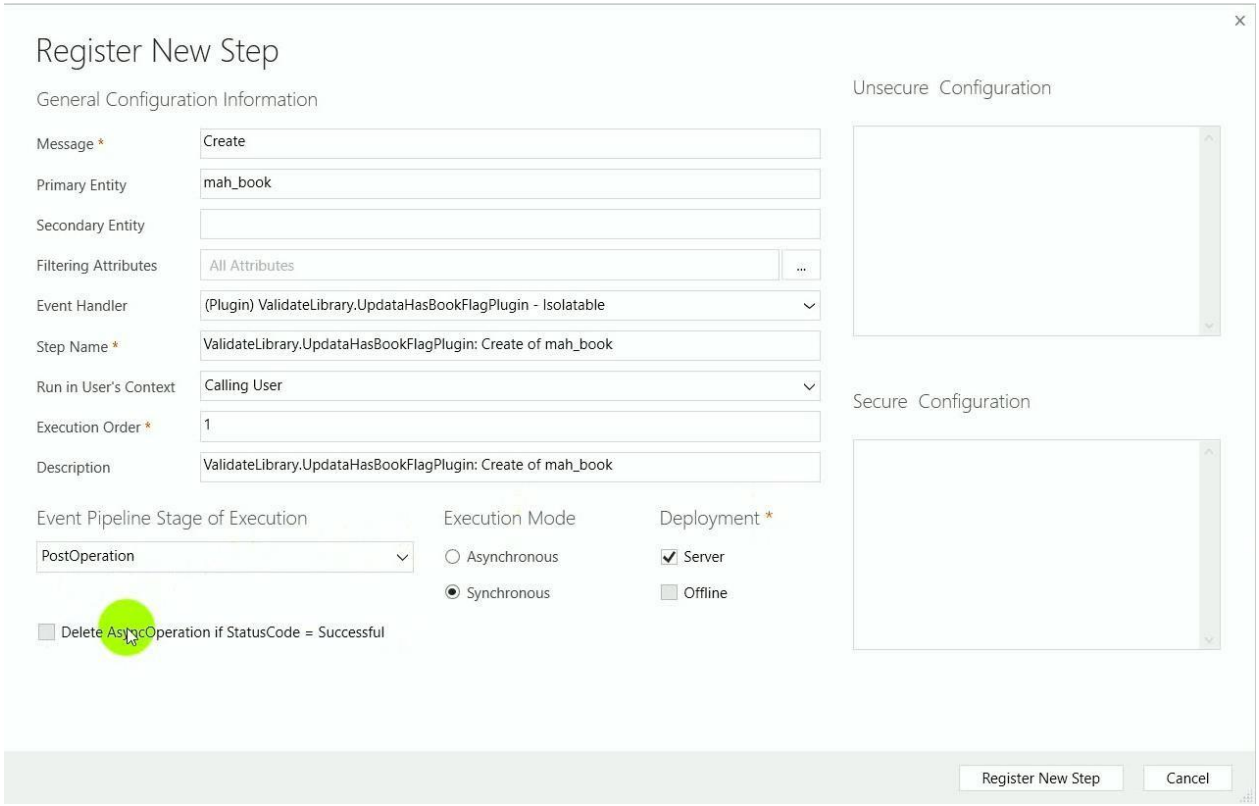
Then update the flag has_book to be true and then you can create a task on the book .

**You need to cast field to targeted data type

```
class UpdataHasBookFlagPlugin : IPlugin
{
    0 references
    public void Execute(IServiceProvider serviceProvider)
    {
        ITracingService tracingService =
(ITracingService)serviceProvider.GetService(typeof(ITracingService));

        // Obtain the execution context from the service provider.
        IPluginExecutionContext context = (IPluginExecutionContext)
            serviceProvider.GetService(typeof(IPluginExecutionContext));



        // The InputParameters collection contains all the data passed in the message request.
        if (context.InputParameters.Contains("Target") &&
            context.InputParameters["Target"] is Entity)
        {
            // Obtain the target entity from the input parameters.
            Entity entity = (Entity)context.InputParameters["Target"];

            //librariyEntityRefrance.LogicalName;
            //librariyEntityRefrance.Id;
            // Obtain the organization service reference which you will need for
            // web service calls.
            IOrganizationServiceFactory serviceFactory =
                (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));
            IOrganizationService service = serviceFactory.CreateOrganizationService(context.UserId);
```

```
try
{
    // Plug-in business logic goes here.
    //entity["mah_name"] //"new book"
    var librariyEntityRefrance = (EntityReference)entity["mah_libraryid"]; //()

    var library = service.Retrieve(librariyEntityRefrance.LogicalName,librariyEntityRefrance.Id,new Columr
    if ((bool)library["mah_hasbook"] == false)
    {
        library["mah_hasbook"] = true;
        service.Update(library);

        Entity taskEntity = new Entity("task");
        taskEntity["subject"] = entity["mah_name"];//"Task 1";
        taskEntity["description"] = "Task  Description " + entity["mah_name"].ToString();
        taskEntity["regardingobjectid"] = new EntityReference(entity.LogicalName, entity.Id);


        service.Create(taskEntity);
    }
}
```

After build project update assembly file then right click on plugin “Register new step” and fill information that appear in next image like to register in specific event and put it in pipeline.



Event Pipeline stage of execution has tree values:

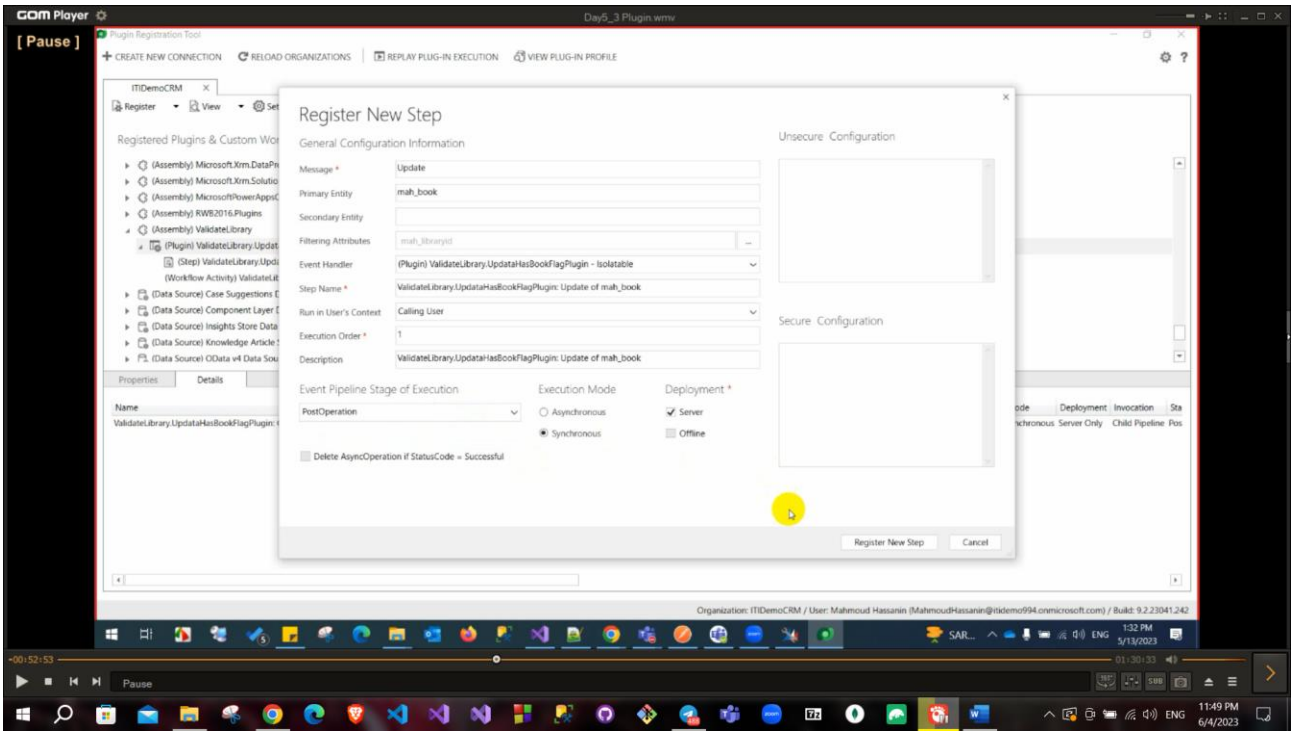
Pre-validation, pre-operation and post-operation and these values are so important

If my operation is create then pre validation is before starting to create

Pre-operation is just before the create and post is after creating

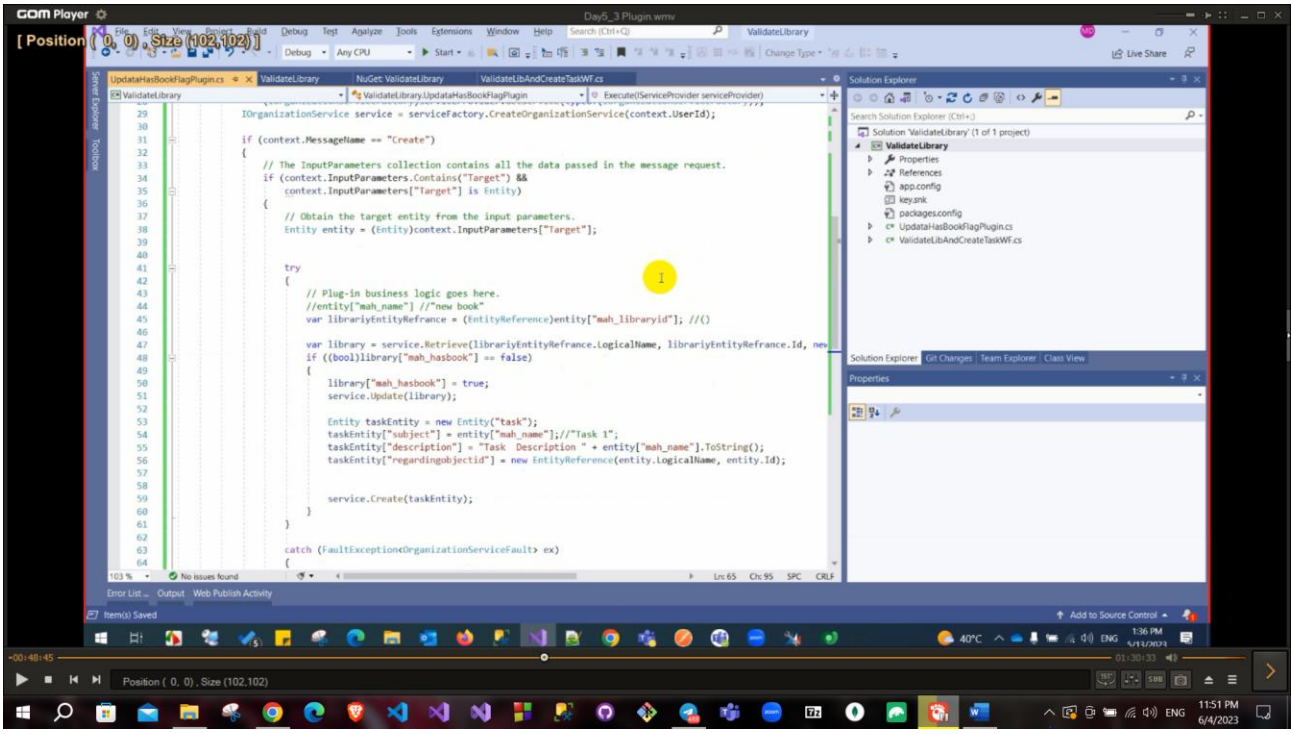
Execution order: if you had more than one plugin has the same event soyou need to order them.

Example: Register new step after update

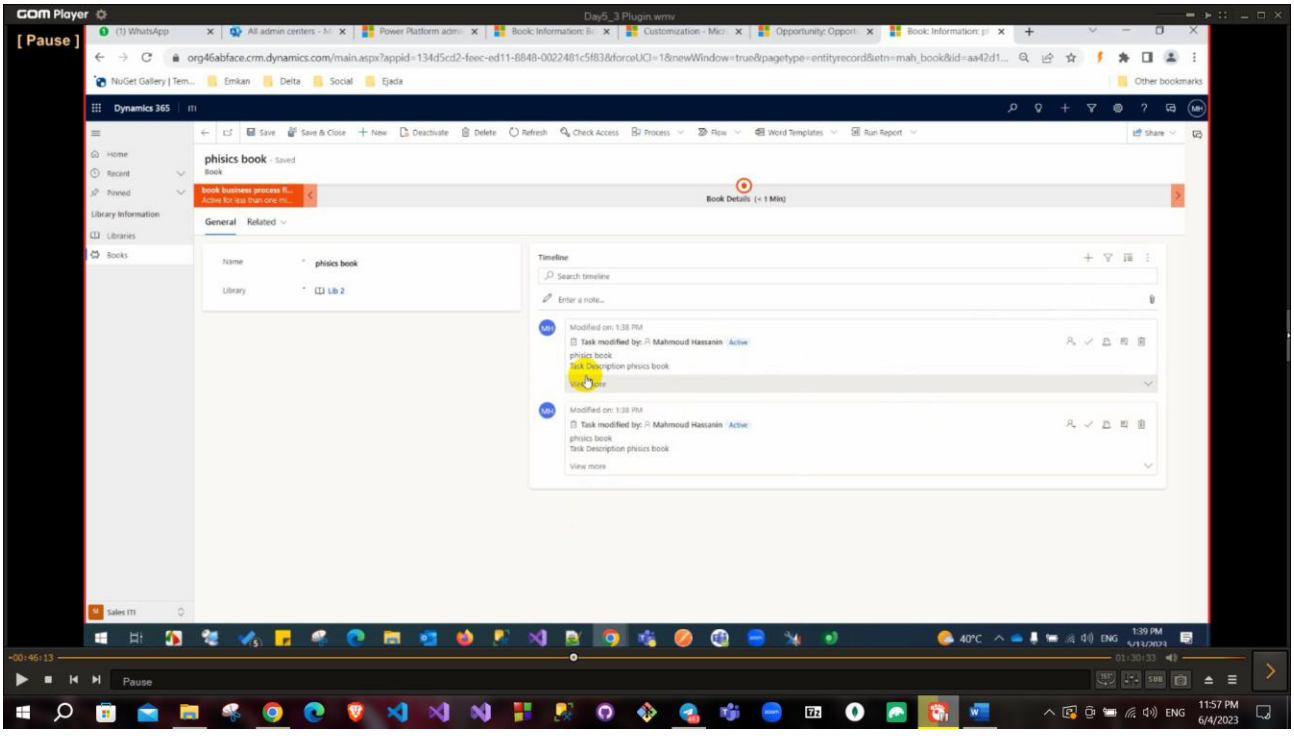


Then add the condition if(context.MessageName == “Create”){}

And add the condition for update also



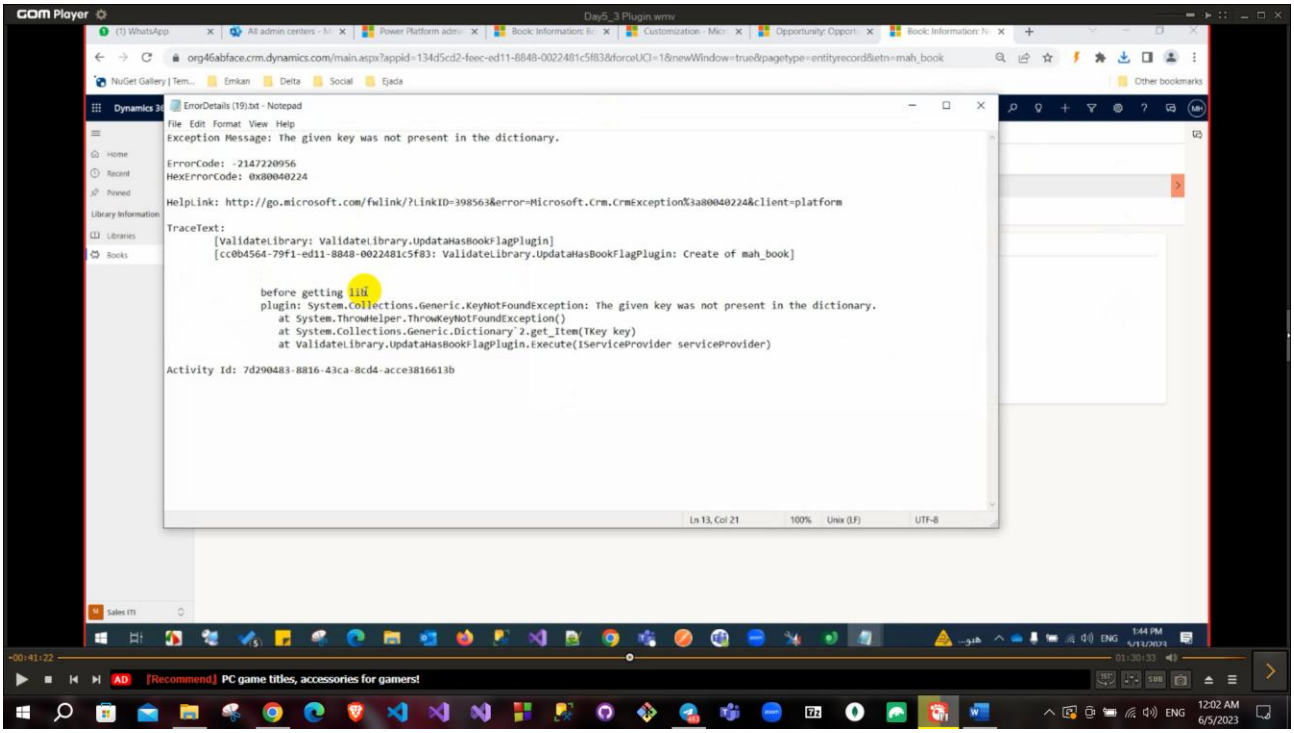
Now when we created a book there are two tasks that were added one by the workflow and the other by the plugin



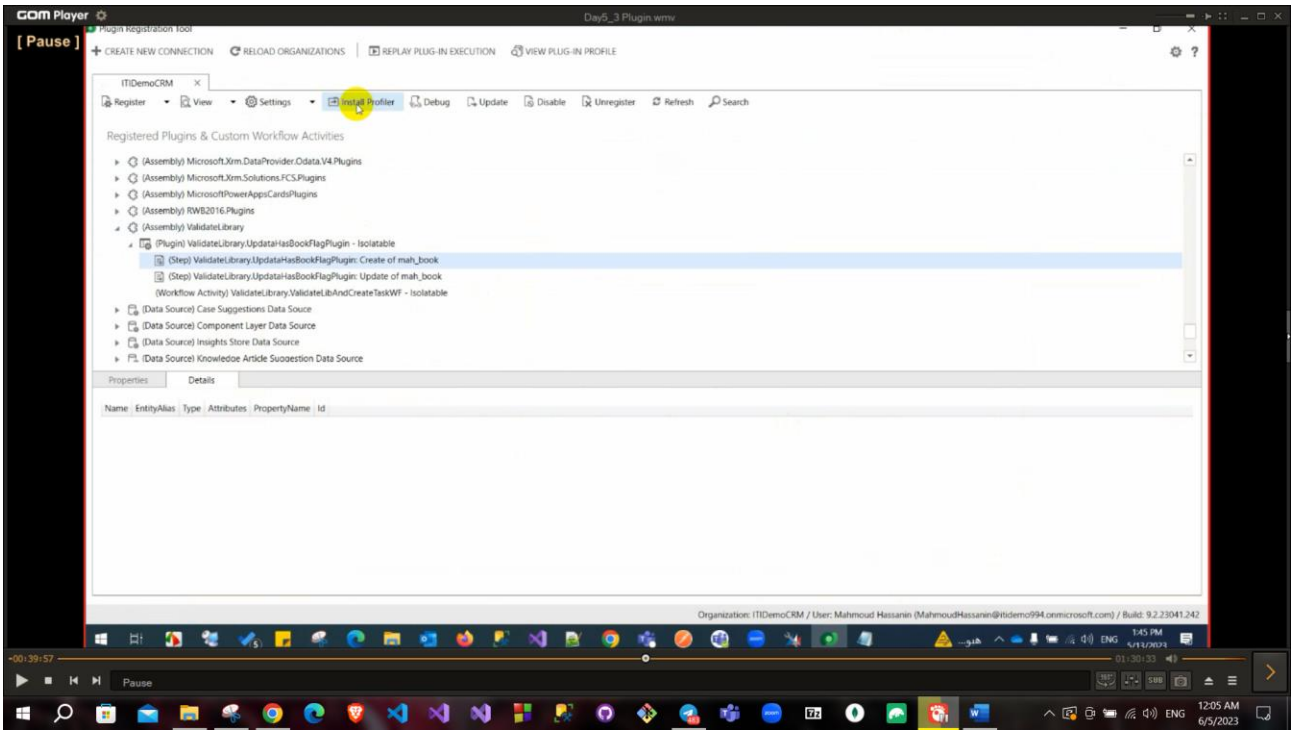
To Trace your plugin use the tracingservice

```
{
    // Obtain the target entity from the input parameters.
    Entity entity = (Entity)context.InputParameters["Target"];


    try
    {
        tracingService.Trace("before getting lib");
        // Plug-in business logic goes here.
        var librariyEntityRefrance = (EntityReference)entity["mah_libraryid"]; //()
        tracingService.Trace("after getting lib");

        var library = service.Retrieve(librariyEntityRefrance.LogicalName, librariyEntityR
        if ((bool)library["mah_hasbook"] == false)
        {
            library["mah_hasbook"] = true;
            service.Update(library);

            Entity taskEntity = new Entity("task");
            taskEntity["subject"] = entity["mah_name"];//"Task 1";
            taskEntity["description"] = "Task  Description " + entity["mah_name"].ToString
            taskEntity["regardingobjectid"] = new EntityReference(entity.LogicalName, enti
```

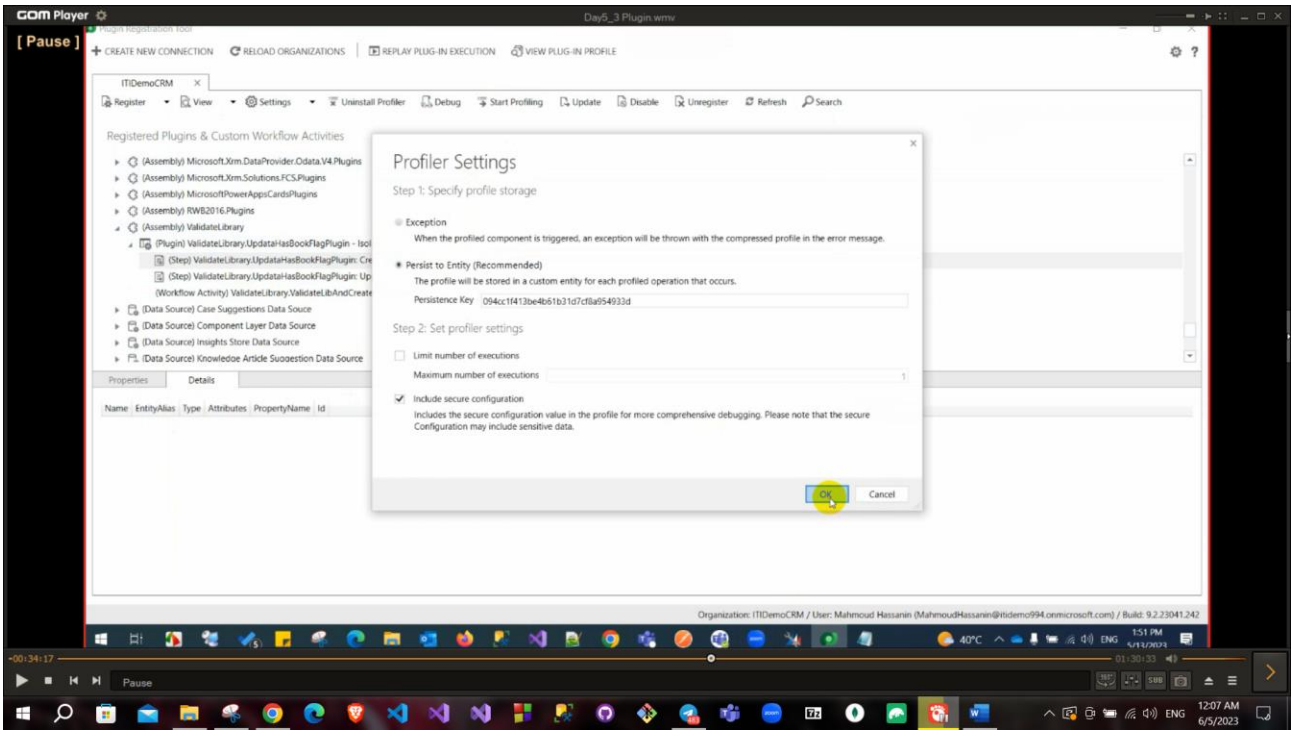
And then download the log file you should see what you printed



To debug you need to install profiler from plugin registration tool

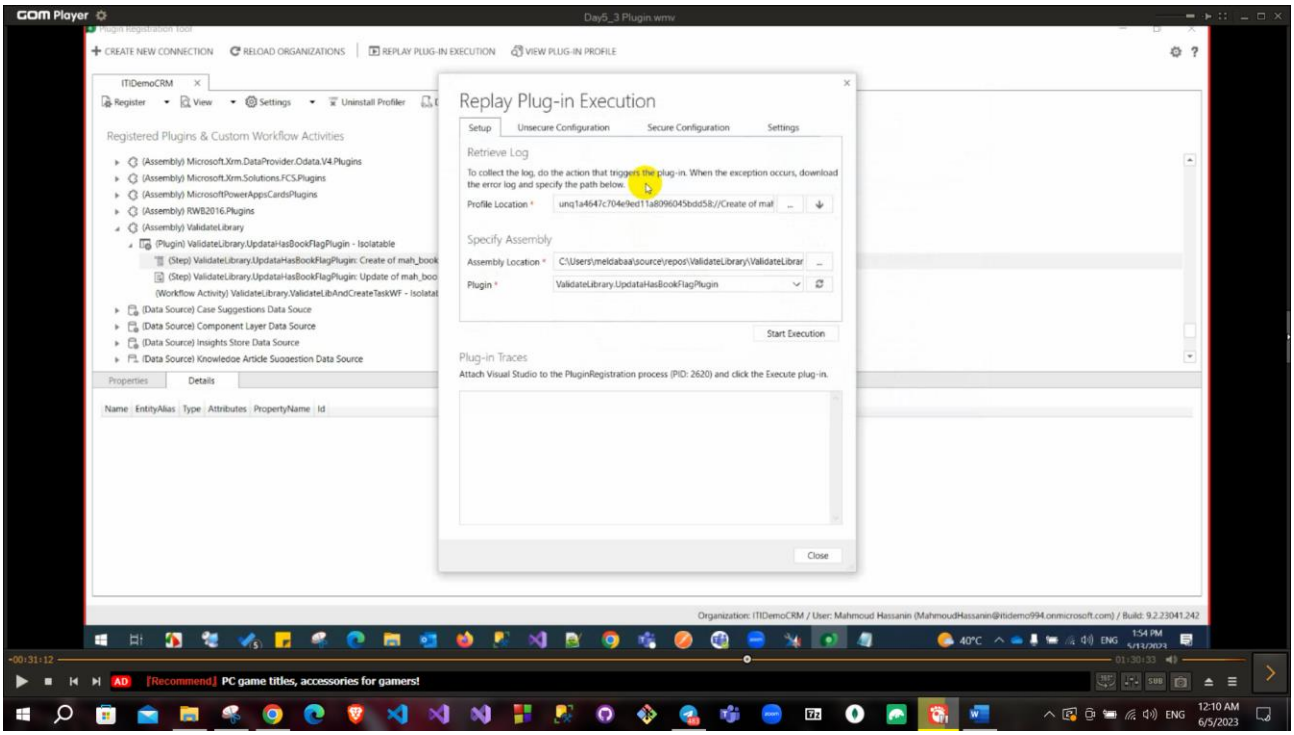


After installation select plugin and start profiling: there are 2 choices persist to entity or exception.

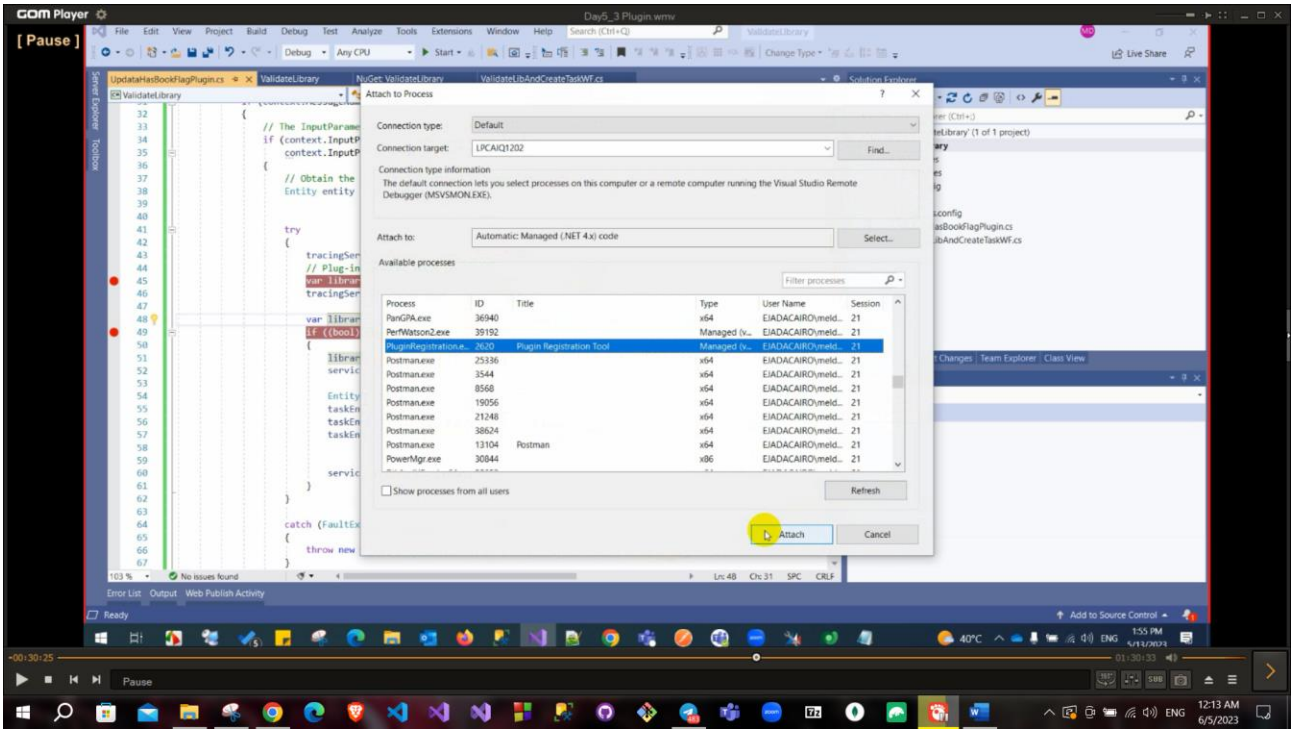


After you create the book and the plugin is triggered you can debug

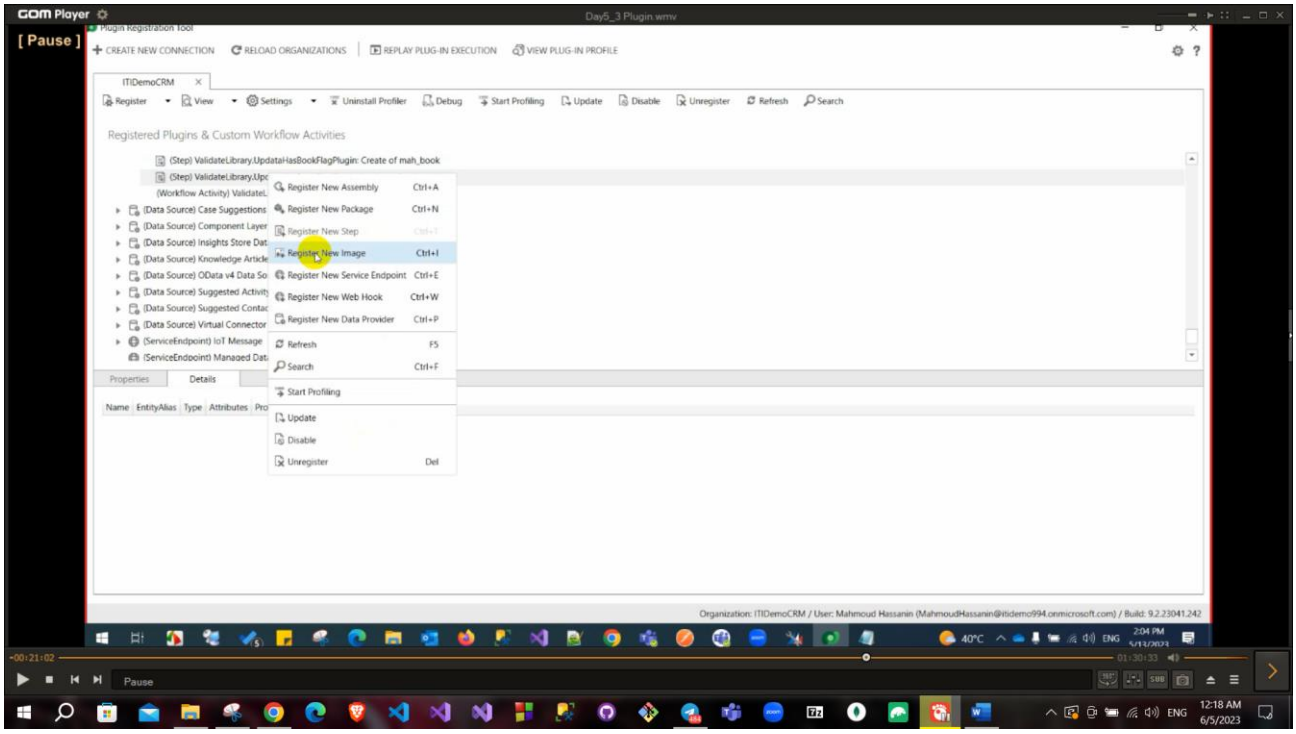
Click on debug and choose the assembly and profile and start execute



Also, you can debug from visual studio → from debug menu → attach to process → plugin registration tool



You can create an image to load old related attribute if you update this attribute (for example you update library when you are dealing with book entity so book entity now has 2 values for library lookup old value and new one so you create this image to track them although you can retrieve old value from service but best practice is to get it from image due to it has higher performance).

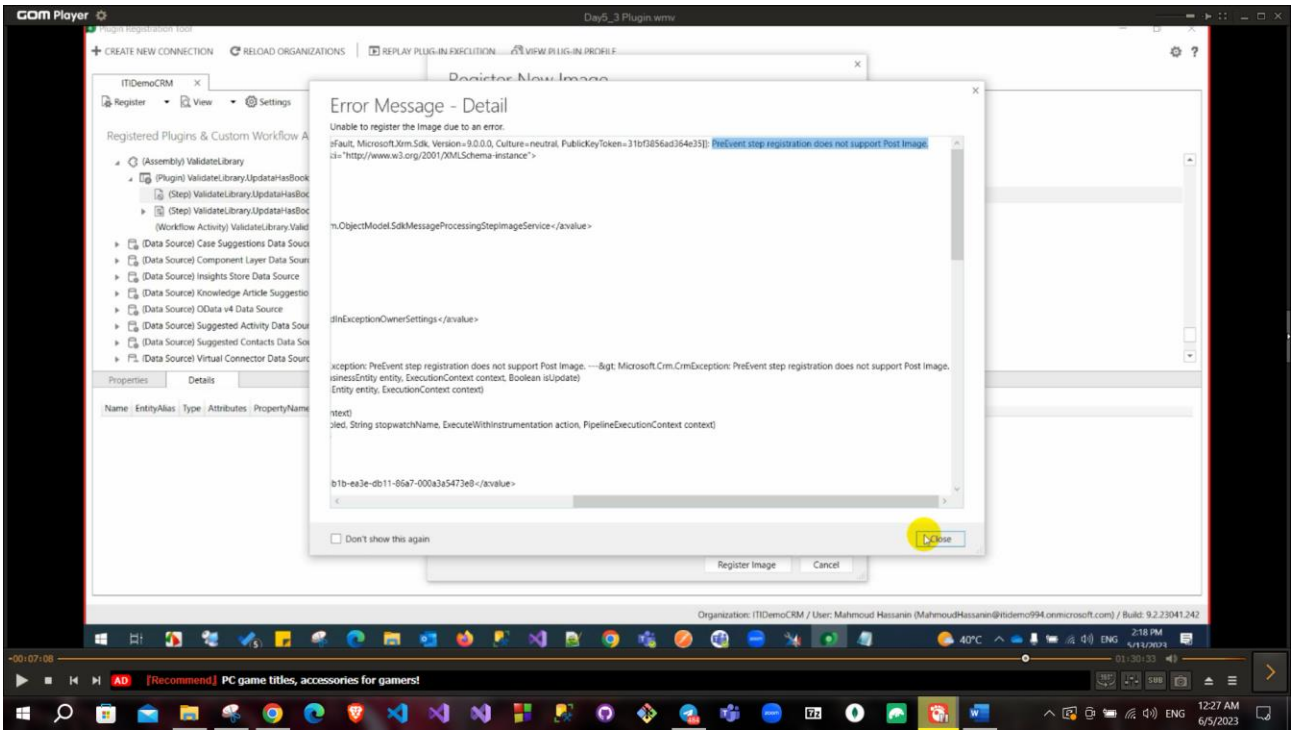


You can choose pre or post image or both

Then in update you can access the image values

```
else if (context.MessageName == "Update")
{
    var preBookImage = context.PreEntityImages["Img"];

    if (preBookImage.Contains("mah_libraryid"))
    {
        var oldValue = preBookImage["mah_libraryid"];
        var newValue = entity["mah_libraryid"];
        throw new InvalidPluginExecutionException("found lib in image (old value)");
    }
    else
    {
        throw new InvalidPluginExecutionException("not found lib as old value");

    }



}
```

If you try to add the plugin as a precreate and tried to add a post image this will throw an error



You have 2 version of images Pre and Post

Message	Stage	Pre-Image	Post-Image
Create	PRE	No	No
Create	POST	No	Yes
Update	PRE	Yes	No
Update	POST	Yes	Yes
Delete	PRE	Yes	No
Delete	POST	Yes	No

Unsecure configuration and secured configuration in plugin search Impersonation