



Fest 2018 Workshop

Building Applications for Relativity and RelativityOne

November 7, 2018 - Version 1.0

Contents

1	Course Overview	3
1.1	What You'll Learn	3
1.2	Required Development Software	3
2	Getting Started.....	4
2.1	Workshop Projects	4
2.2	DevVM Setup	5
3	Visual Studio Templates.....	9
3.1	Application Set Up	9
3.2	New Event Handler	13
3.3	Publish	18
3.4	New Agent.....	23
3.5	New CustomPage	24
4	Sample Application Overview.....	28
4.1	Install Sample Application	28
4.2	Run Sample Application	30
5	Unit Tests.....	33
5.1	Writing Unit Tests.....	33
5.2	Running Unit Tests	38
6	Integration Tests	40
6.1	Writing Integration Tests	40
6.2	Running Integration Tests.....	43
6.2	Running Integration Tests as CI	45
7	Gravity API.....	50

1 Course Overview

As a developer, you want a solid process for building and testing your apps, so you can focus on delivering value quickly. The Application Deployment System (ADS) is the standard way to build applications on the Relativity platform. Relativity Test Helpers is our new, open source testing framework.

In this advanced session, we'll review both of these standard approaches, covering common patterns for setting up effective ADS development solutions and integration tests. Topics will include: setting up your development environment using DevVM's; Visual Studio developer tools including templates, NuGet packages, and the "publish to Relativity" extension; and GitHub open source solutions such as Gravity, Test Helpers, and more. We'll also highlight ways to ensure your applications make a smooth transition from the on-premise world to RelativityOne. You will create tests and learn how to run those test from scripts as well as gain practical experience writing tests and potentially automating tests for your solutions, ensuring your customization will be ready run with the latest release of Relativity and RelativityOne.

1.1 What You'll Learn

- Relativity Templates
- Publish to Relativity
- Unit Tests
- Integration Tests
- Run Integration Tests as CI
- Gravity API

1.2 Required Development Software

This workshop and associated development resources require a development environment that includes the following items:

- Relativity 9.6.134.78 DevVM
- Visual Studio 2017
- Relativity Visual Studio Templates
- .NET Framework 4.6.2
- SQL Server 2016

We have provided the required software for you for this workshop.

2 Getting Started

2.1 Workshop Projects

1. Make sure you have a folder shortcut on Desktop with name **ads-workshop-fest-2018**.



2. Double click on the **ads-workshop-fest-2018** folder to open it. Make sure you see the files as shown in the below screenshot.

Name	Date modified	Type
.git	9/13/2018 7:27 PM	File folder
DLLs	9/13/2018 7:15 PM	File folder
PowerPoint	9/13/2018 7:27 PM	File folder
Projects	9/13/2018 7:16 PM	File folder
Workbook	9/13/2018 7:27 PM	File folder
.gitattributes	9/13/2018 7:15 PM	Text Document
.gitignore	9/13/2018 7:15 PM	Text Document
LICENSE	9/13/2018 7:15 PM	File
README.md	9/13/2018 7:16 PM	MD File

3. Go to the **Projects/Starting** folder to make sure you see the folders as shown in the below screenshot. These folders contain Visual Studio projects for different sections in the workbook.

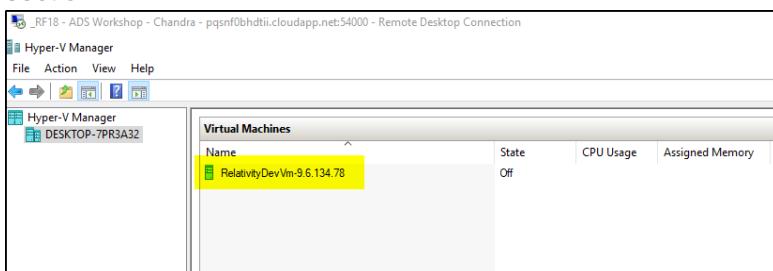
Name	Date modified	Type
1_Templates	9/13/2018 7:16 PM	File folder
2_Application	9/13/2018 7:16 PM	File folder
3_UnitTests	9/13/2018 7:16 PM	File folder
4_IntegrationTests	9/13/2018 7:16 PM	File folder
5_Gravity	9/13/2018 7:16 PM	File folder

2.2 DevVM Setup

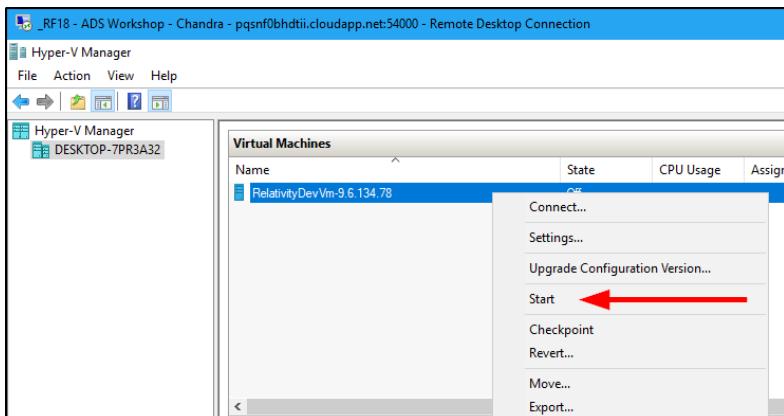
1. Open the **Hyper-V Manager** application by clicking the blue icon in the taskbar.



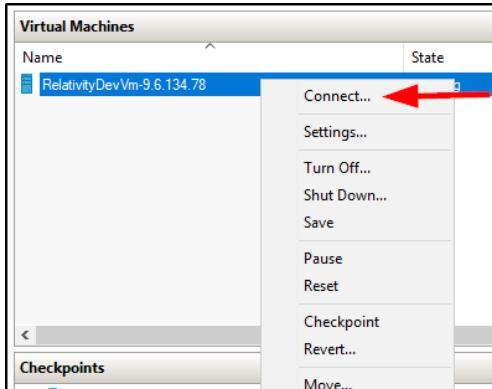
2. You will see a **Relativity 9.6.134.78 DevVM** listed in the Hyper-V Manager **Virtual Machines** section.



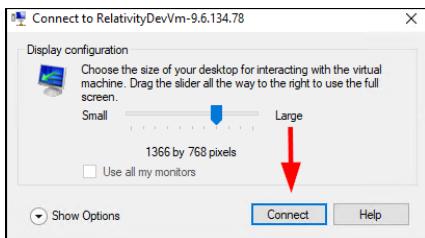
3. Right click on the **DevVM** and select the **Start** option.



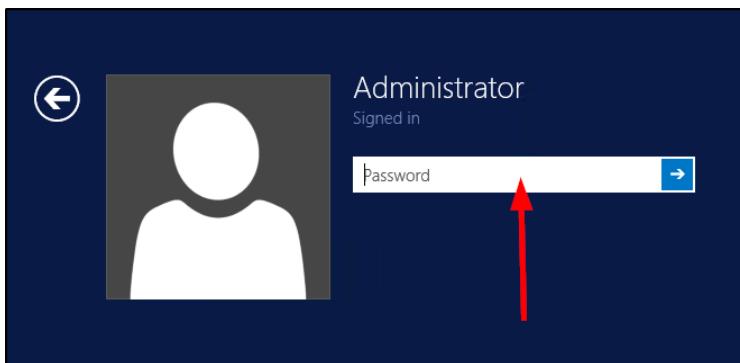
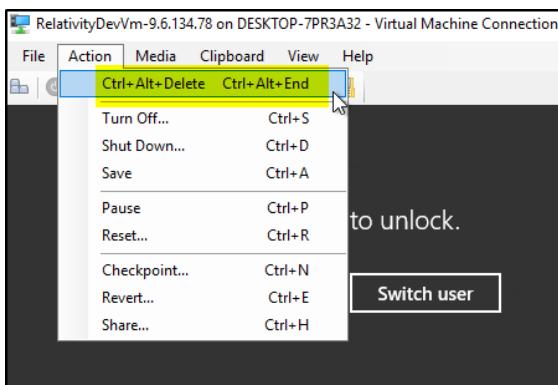
4. Right click on the **DevVM** and select the **Connect** option.



5. If you get a Display configuration pop-up, click the **Connect** button.



6. Select the **Action** Menu item and then select **Ctrl+Alt+Delete** option.



7. Use the below DevVM credentials highlighted in **yellow** to login to the VM.

DevVM:

Windows Admin login: Administrator

Windows Admin password: Test1234!

Relativity:

Admin login: relativity.admin@relativity.com

Admin password: Test1234!

SQL:

login: eddsdbo

password: Test1234!

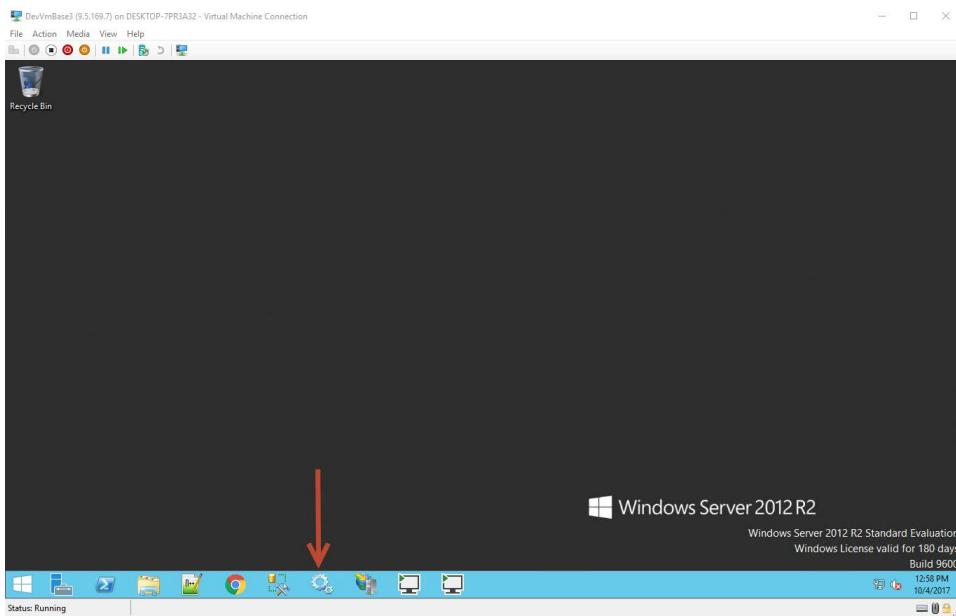
Admin login: sa

Admin password: Test1234!

8. Once you log in to the DevVM, you should see the Desktop with few applications pinned to the taskbar, which you will be using throughout the workshop.



9. Click on the **Services** program icon in the taskbar as shown in the below screenshot.



10. Make sure all the **services** listed below are **running**. If they are not already running, right click on the service and select the **Start** option,

- Service Bus Gateway
- Service Bus Message Broker
- Service Bus Resource Provider
- Service Bus VSS
- kCura Service Host Manager
- kCura EDDS Web Processing Manager
- kCura EDDS Agent Manager

Name	Description	Status	Startup Type	Log On As
kCura EDDS Agent Manager	Performs ba...	Running	Automatic (D...	\Relativity...
kCura EDDS Web Processing Manager	Deploys cus...	Starting	Automatic (D...	\Relativity...
kCura Service Host Manager	Hosts Relati...	Automatic (D...	Automatic (D...	\Relativity...
KDC Proxy Server ser...	KDC Proxy S...	Manual	Network S...	
KtmRm for Distribut...	Coordinates...	Manual (Trig...	Network S...	
Link-Layer Topology...	Creates a N...	Manual	Local Service	
Local Session Manag...	Core Windo...	Running	Automatic	Local Syste...
Lotus Notes Diagnos...	Performs di...	Automatic	Local Syste...	
Lotus Notes Smart U...	A service th...	Running	Automatic	Local Syste...
Microsoft Account S...	Enables use...	Running	Manual (Trig...	Local Syste...
Microsoft iSCSI Initi...	Manages In...	Manual	Local Syste...	
Microsoft SharePoint...	Manages so...	Manual	Local Syste...	
Microsoft Software S...	Host service...	Manual	Network S...	
Microsoft Storage Sp...	Enables rela...	Manual	Local Syste...	
Multimedia Class Scheduler				

11. Open PowerShell on your DevVM and get the IP address of the VM.

```
Administrator: Windows PowerShell
PS C:\Users\Administrator> ipconfig
Windows IP Configuration

Ethernet adapter Ethernet:

  Connection-specific DNS Suffix . : mshome.net
  IPv4 Address . . . . . : 172.27.145.78 <-- Red arrow here
  Subnet Mask . . . . . : 255.255.255.240
  Default Gateway . . . . . : 172.27.145.65

Tunnel adapter isatap.mshome.net:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . : mshome.net
PS C:\Users\Administrator>
```

12. Open **Chrome** on your Workshop machine (Not the DevVM) and go to <http://172.27.145.78/Relativity>. Use the below Relativity admin credentials to login. Verify that you can login to Relativity.

a. If the workspaces tab doesn't load, give it a few mins and then try refreshing the tab.

Relativity:

Admin login: relativity.admin@relativity.com

Admin password: Test1234!

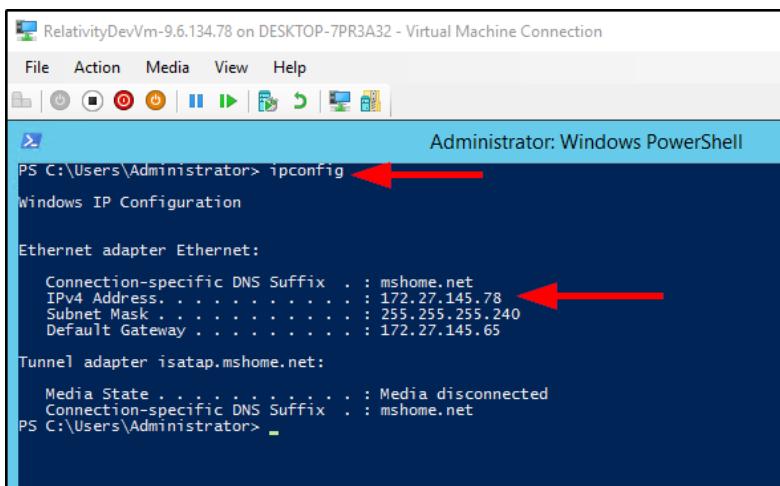
#	Case Artifact ID	Name	Client Name	Matter Name	Last Modified On
1	1017168	Edit ADS Workshop Fest 2018	Relativity Template	Relativity Template	09/13/2018 8:38 PM CDT

3 Visual Studio Templates

Relativity is an extensible platform that allows an administrator/developer to create customizations. Development on the Relativity Platform is made easier with the use of some tools such as Visual Studio Templates and the Publish to Relativity GUI. This section demonstrates utilizing the visual studio templates and Publish to Relativity tool to quickly iterate on application development.

3.1 Application Set Up

1. Open PowerShell on your DevVM and get the IP address of the VM.



```
RelativityDevVm-9.6.134.78 on DESKTOP-7PR3A32 - Virtual Machine Connection
File Action Media View Help
Administrator: Windows PowerShell
PS C:\Users\Administrator> ipconfig
Windows IP Configuration

Ethernet adapter Ethernet:
  Connection-specific DNS Suffix . : mshome.net
  IPv4 Address . . . . . : 172.27.145.78
  Subnet Mask . . . . . : 255.255.255.240
  Default Gateway . . . . : 172.27.145.65
  Tunnel adapter isatap.mshome.net:
    Media State . . . . : Media disconnected
    Connection-specific DNS Suffix . : mshome.net
PS C:\Users\Administrator>
```

2. Open **Chrome** on your Workshop machine (Not the DevVM) and go to <http://172.27.145.78/Relativity>. Use the below Relativity admin credentials to login.
 - a. If the workspaces tab doesn't load, give it a few mins and then try refreshing the tab.

Relativity:

Admin login: relativity.admin@relativity.com

Admin password: Test1234!

3. Enter **ADS Workshop Fest 2018** and navigate to the **Relativity Applications Tab**.

The screenshot shows the Relativity application interface. At the top, there is a navigation bar with links for 'Documents', 'Review Batches', 'Reporting', 'Case Admin', 'Job Admin', 'Workspace Admin' (which is highlighted), 'Indexing & Analytics', 'Persistent Lists', 'Production', 'Relativity Applications' (which is also highlighted with a red box), and 'Custom Pages'. Below the navigation bar is a search bar with a placeholder 'All Relativity Applications' and a dropdown menu. The main content area displays a table with 10 rows of data, each representing a Relativity Application. The columns are labeled '#', 'Artifact ID', 'Name', and 'System Last Modified On'. The data is as follows:

#	Artifact ID	Name	System Last Modified On
1	1039380	Edit Workspace Portal (Duplicate)	
2	1039360	Edit Relativity Color Map	
3	1039356	Edit Single File Upload	
4	1039337	Edit Field Catalog	
5	1039332	Edit Data Grid Core	
6	1039327	Edit Choice Editor	
7	1039012	Edit Relativity List Page	
8	1038682	Edit Production	
9	1038500	Edit Document Viewer	
10	1038379	Edit Lists	

4. Create a new **Relativity Application** with the name of your choice.

ADS Workshop Fest 2018 - Relativity

Developer Mode Is Active

ADS Workshop Fest 2018

Application Type

Application Type: Create new Application
 Select from Application Library
 Import from File

Application Information

New Application Name:

Version
Enter in the format X.X.X
Revision number assigned automatically:

User-friendly URL:

Save Save and New Save and Back Cancel

5. Create a new **Object Type** to the application from the view page. This auto-associates the objects initial views, fields, and layouts.

Default Relativity Application Layout

Application Information

Name: MyNewApp

Version:

User-friendly URL:

History

System Created By: Admin, Relativity

System Last Modified By: Admin

System Created On: 9/13/2018 10:51 PM

System Last Modified On: 9/13/2018 10:51 PM

Object Type (Relativity Applications)

New Link Unlink

	Name	System	Dynamic	Created On	Created By
<input type="checkbox"/>	(All)	(All)	(All)	(All)	(All)

0 Selected Item(s)

Documents Review Batches Reporting Case Admin Job Admin Workspace Admin Indexing & Analytics Persistent Lists

Workspace Details Search Indexes Object Type Fields Choices Choices List Layouts Views Tabs Relativity Applications Custom

Save Save and New

Object Type Information

Name: MyNewObject

Parent Object Type: Workspace

Dynamic: Yes

Enable Snapshot Auditing On

Delete: Yes

Pivot: Enabled

Sampling: Enabled

Lists: Disabled

Copy Instances On Workspace Creation:

No

Copy Instances On Parent Copy:

No

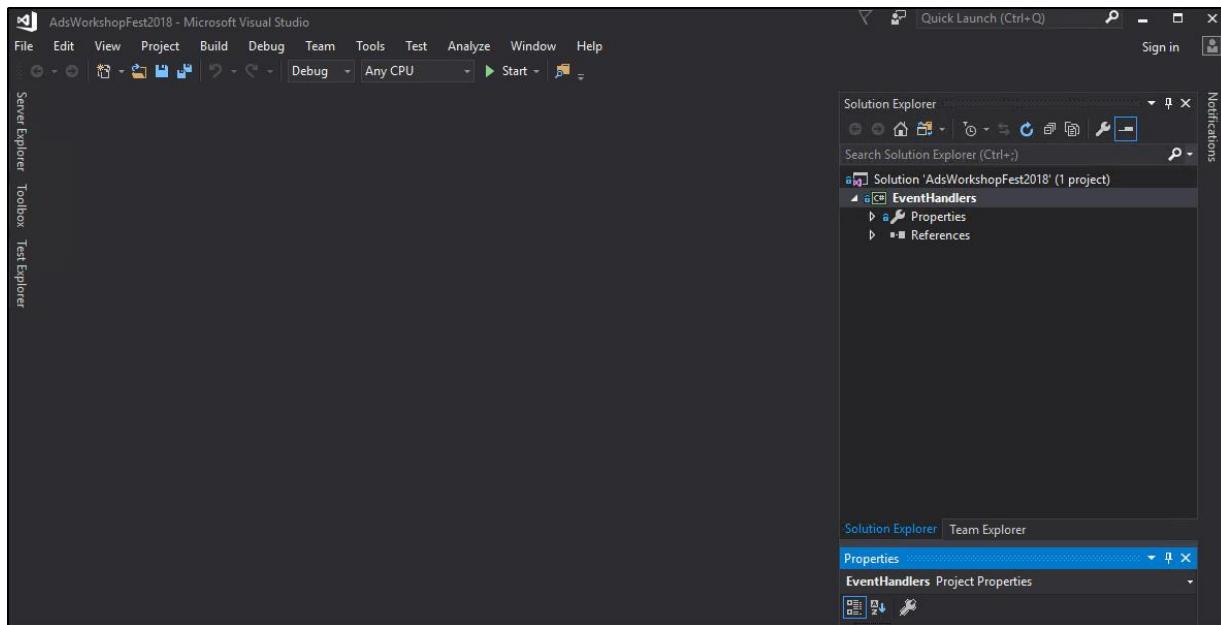
Relativity Applications: MyNewApp

Keywords:

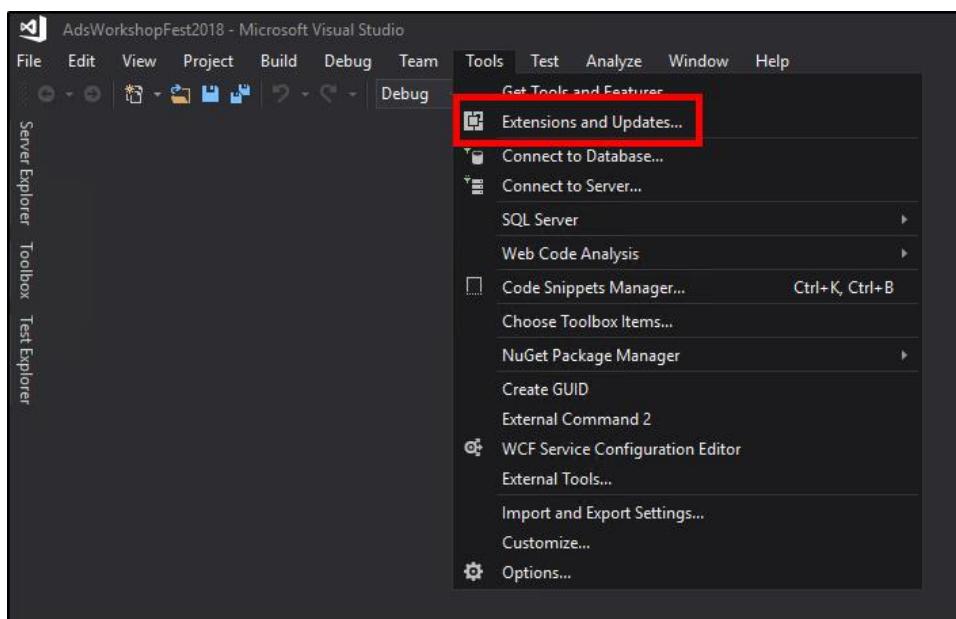
Notes:

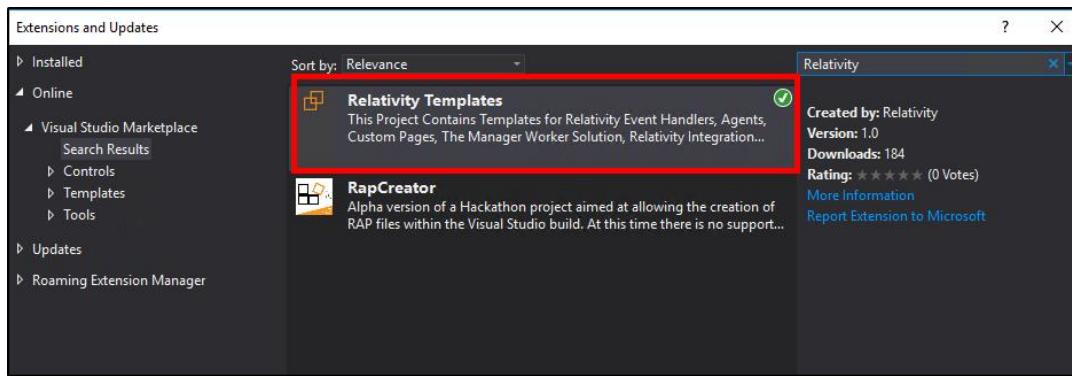
3.2 New Event Handler

1. Open the Visual Studio solution **AdsWorkshopFest2018.sln** in the **E:\Github\ads-workshop-fest-2018\Projects\Starting\1_Templates\Project** folder.

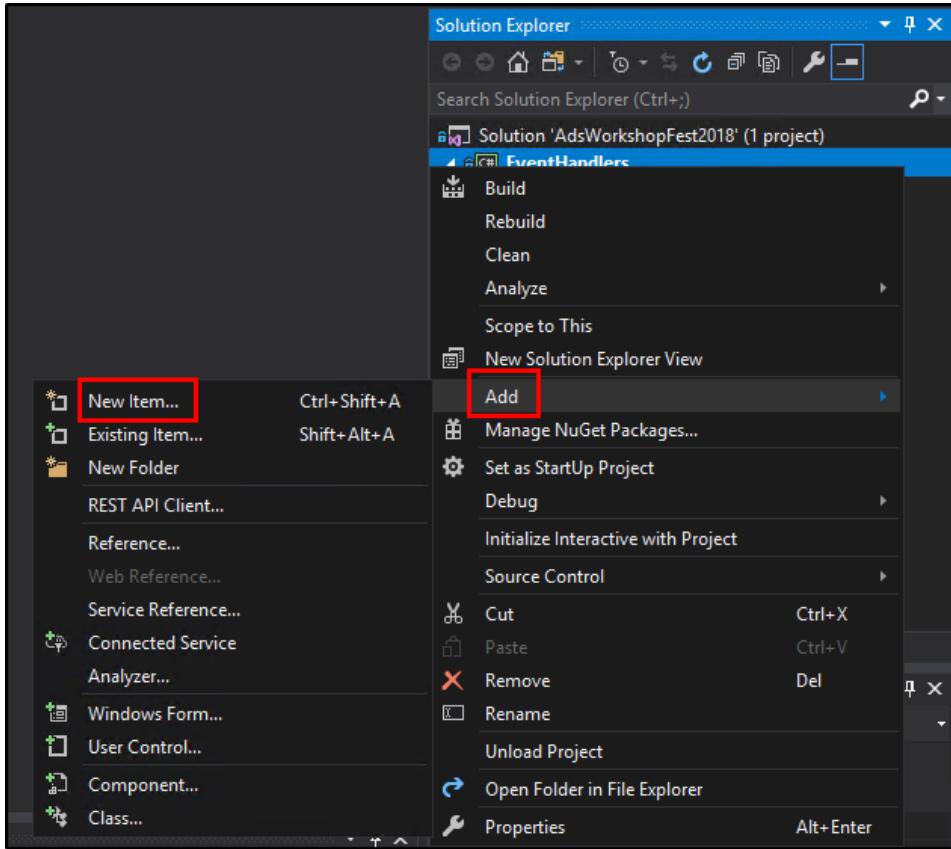


2. Verify the **Relativity Templates** are installed. You may need to **restart** Visual Studio.
3. **Tools — Extensions and Updates — Search "Relativity"**

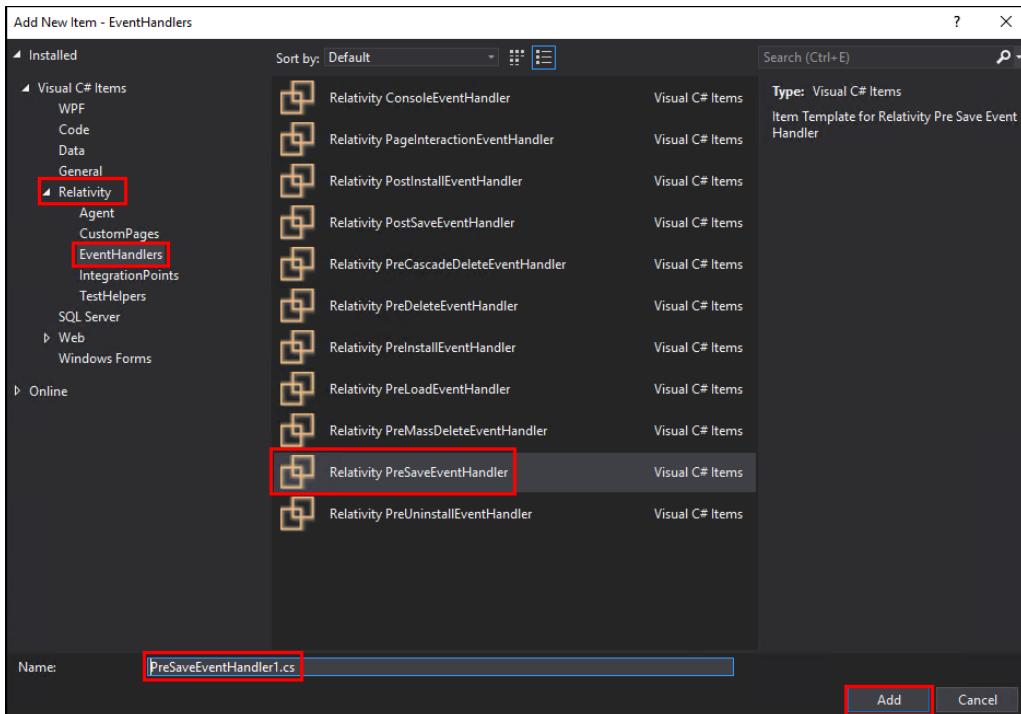




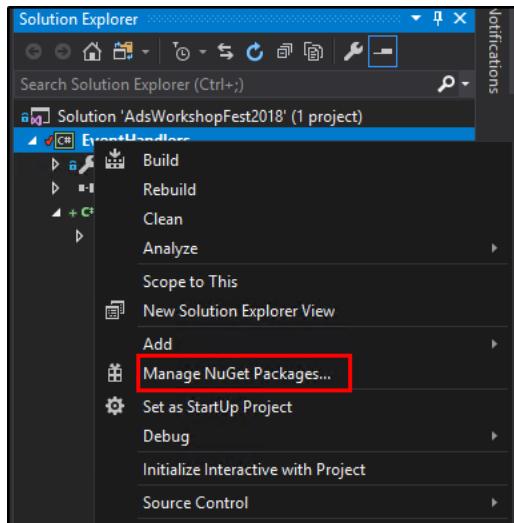
4. Right click the **EventHandlers** Project in the solution explorer. Select Add — New Item.



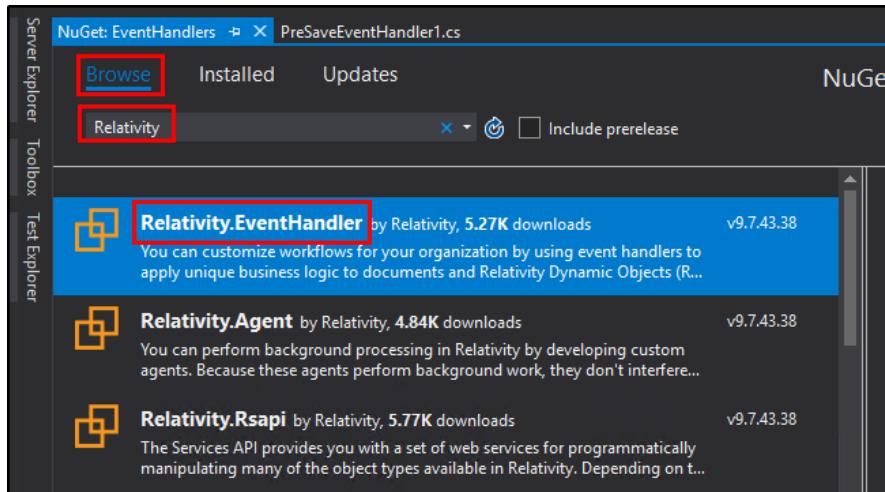
5. Under the Relativity Section, select the pre-save event handler template.



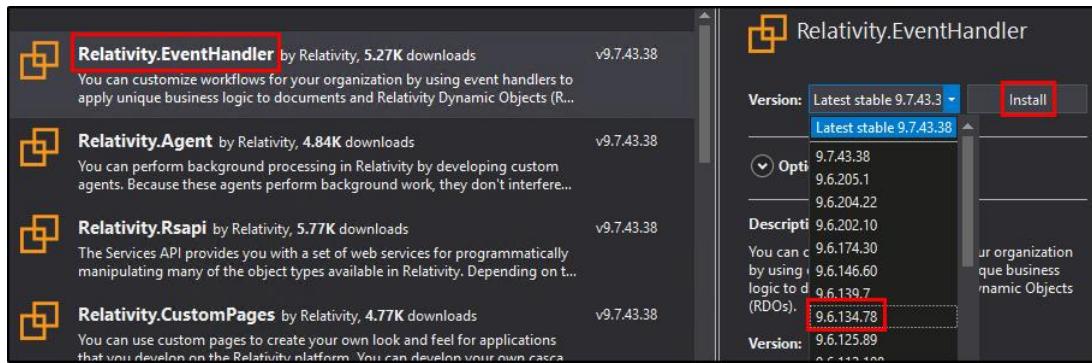
6. Click **Add**.
7. Install the 9.6.134.78 **Relativity.EventHandler** NuGet package.
 - a. Right click on the Project.
 - b. Select **Manage NuGet Packages** option.



- c. Switch to **Browse** tab.
- d. Search for “**Relativity**”.
- e. Select the **Relativity.EventHandler** NuGet package.



- f. From the **Version** drop-down select the **9.6.134.78** option.
- g. Click **Install**.



8. Update the event handler response.

- Success* – false
- Message* – Message of your choice

The screenshot shows a code editor in Visual Studio with the file 'PreSaveEventHandler1.cs' open. The code defines a class 'PreSaveEventHandler1' that inherits from 'kCura.EventHandler.PreSaveEventHandler'. The 'Execute' method is overridden. Inside the method, a response object is constructed with default values. A try block then executes some logic. If successful, it sets 'RetVal.Success' to true and 'RetVal.Message' to an empty string. If it fails, it sets 'RetVal.Success' to false and 'RetVal.Message' to the string 'FirstTime'. Both the 'Success' and 'Message' assignments are highlighted with a red box.

```

using Relativity.Services.Objects;
namespace EventHandlers
{
    [kCura.EventHandler.CustomAttributes.Description("Pre Save EventHandler")]
    [System.Runtime.InteropServices.Guid("4e1adc09-4c8e-443a-986d-487c475dc8ec")]
    public class PreSaveEventHandler1 : kCura.EventHandler.PreSaveEventHandler
    {
        public override Response Execute()
        {
            //Construct a response object with default values.
            kCura.EventHandler.Response retVal = new kCura.EventHandler.Response();
            retVal.Success = true;
            retVal.Message = string.Empty;
            try
            {
                Int32 currentWorkspaceArtifactID = Helper.GetActiveCaseID();

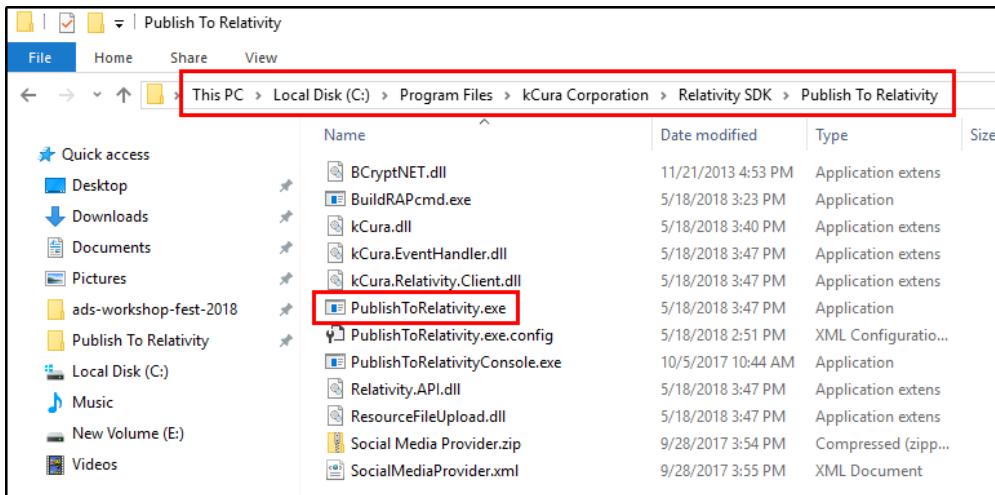
                //The Object Manager is the newest and preferred way to interact with Relativity instead of the RSAPI. The RSAPI will be scheduled for depreciation after the Object Manager reaches feature parity.
                using (IObjectManager objectManager = this.Helper.GetServiceManager().CreateProxy<IObjectManager>())
                {
                    retVal.Success = false;
                    retVal.Message = "FirstTime";
                }

                //Setting up an RSAPI Client
                using (IRSAPIClient proxy =
                    Helper.GetServiceManager().CreateProxy<IRSAPIClient>(ExecutionIdentity.System))
                {
                    //Set the proxy to use the current workspace
                    proxy.APIOptions.WorkspaceID = currentWorkspaceArtifactID;
                    //Add code for working with RSAPIClient
                }
            }
        }
    }
}

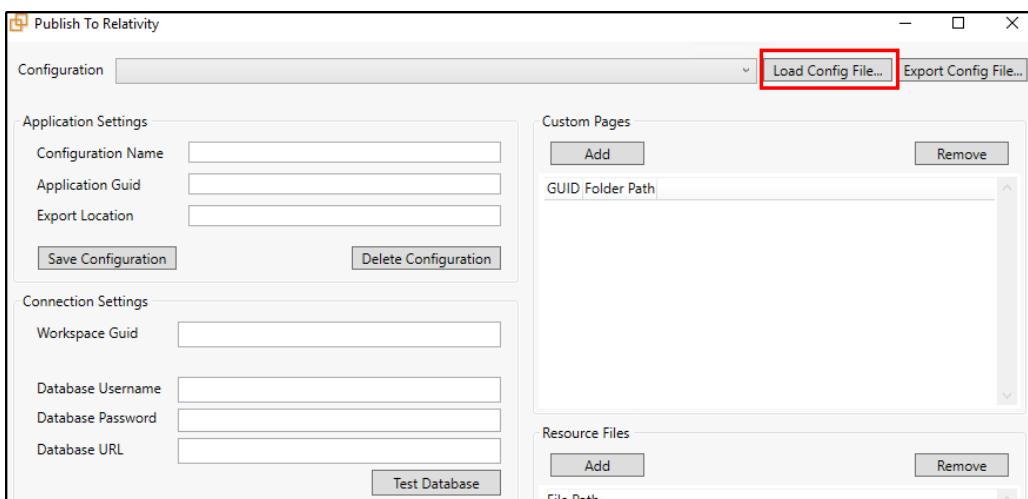
```

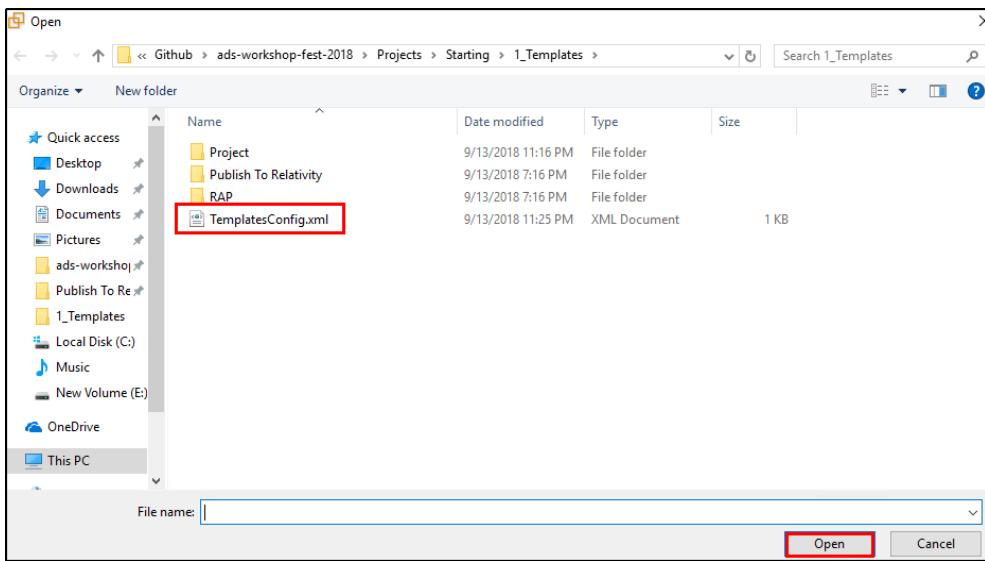
3.3 Publish

1. Build the solution.
2. **Build — Build Solution**
3. Run the Publish to Relativity executable.



4. Click the **Load Config File** button and select the **TemplatesConfig.xml** file in the **E:\Github\ads-workshop-fest-2018\Projects\Starting\1_Templates** folder.





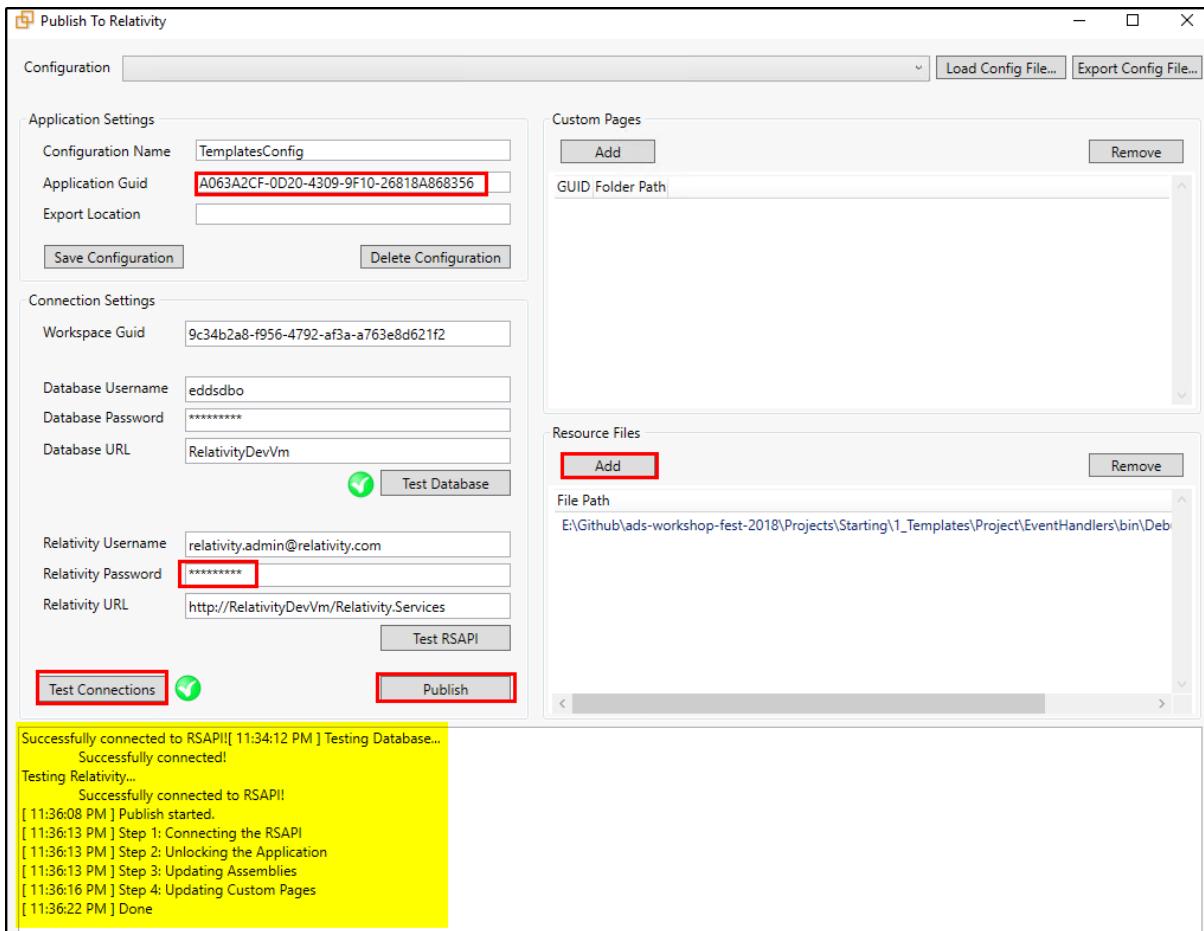
5. Enter the **Relativity Password** (Test1234!).
6. Click on **Test Connections** to validate the connections.
7. Replace the **Application GUID** with your applications GUID.
 - a. To look up your applications GUID, navigate back to Relativity Application you created in the workspace. Then select **Show Component GUIDs**.
 - b. **Developer Mode** must be active.

Application Components

Artifact ID	Artifact Type	Artifact Name	Version	Guids
1039394	Relativity Application	MyNewApp		A063A2CF-0D20-4309-9F10-26818A868356
1039403	Object Type	MyNewObject		5E886D16-184D-479F-8028-B62FB3939E63
1039400	Field	MyNewObject - Name		E6054717-3A5B-4BCD-

8. In the **Resource Files** Section click Add.
9. Navigate to **E:\Github\ads-workshop-fest-2018\Projects\Starting\1_Templates\Project\EventHandlers\bin\Debug** and select **EventHandlers.dll** file.

10. Click **Publish**.



11. Open Relativity and navigate to your application created in section 3.1.

12. **Unlock** the application.

The screenshot shows the Relativity application management interface. The 'Relativity Applications' tab is selected. On the left, there is a form for 'Application Information' with fields for Name (MyNewApp), Version (0.0.0.1), and User-friendly URL. Below it is a 'History' section showing creation and modification details. On the right, there is a sidebar titled 'RELATIVITY APPLICATION' with several buttons: 'Manage Application', 'Upgrade Application', 'Export Application', 'Export Application Schema', 'Unlock Application' (which is highlighted with a red box), and 'Push to Library'. Below the sidebar are sections for 'Application Information' with links to 'Show Application Breakdown' and 'Show Component GUIDs'.

13. Navigate to the view page of the object type created in section 3.1.

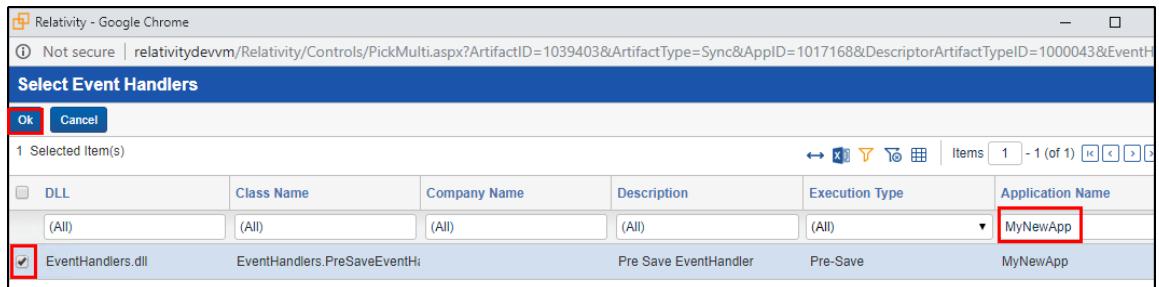
The screenshot shows the Relativity Applications view page for the object type 'MyNewApp'. The top navigation bar includes tabs for Documents, Review Batches, Reporting, Case Admin, Job Admin, Workspace Admin, Indexing & Analytics, Persistent Lists, Relativity Applications (which is highlighted with a red box), Custom Pages, and History. The main content area displays 'Application Information' with fields for Name (MyNewApp), Version (0.0.0.1), and User-friendly URL. Below this is a 'History' section showing creation and modification details. A table lists 'Object Type (Relativity Applications)' with one item: 'MyNewObject'. The 'Edit' button for this item is highlighted with a red box. On the right side, there's a sidebar titled 'RELATIVITY APPLICATION' with options like Manage Application, Upgrade Application, Export Application, Export Application Schema, Unlock Application, Push to Library, Application Information, Show Application Breakdown, Show Component GUIDs, Show Errors, and Refresh Page.

14. Click **New** on the Event handler associated item list.

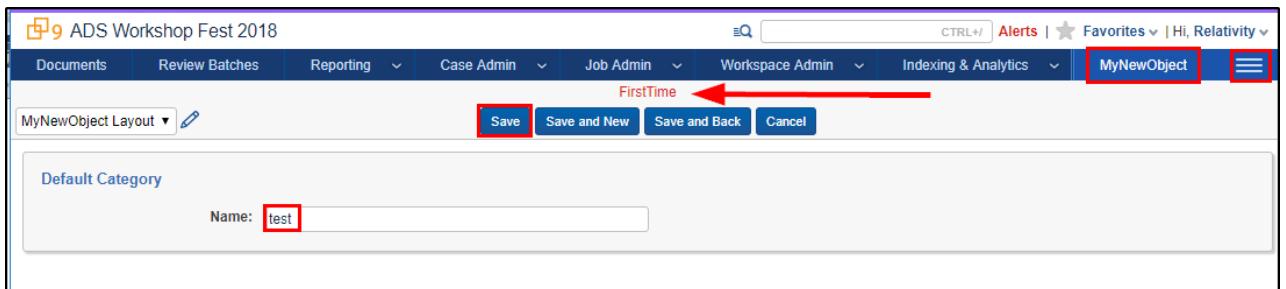
The screenshot shows the Event Handlers list page for the object type 'MyNewApp'. The top navigation bar includes tabs for Documents, Review Batches, Reporting, Case Admin, Job Admin, Workspace Admin, Indexing & Analytics, Persistent Lists, Relativity Applications (highlighted), Custom Pages, and History. The main content area shows a message about exceeding system limits. Below this is a 'Record History' section with creation and modification details. The 'Event Handlers' section contains a 'New' button, which is highlighted with a red box. A table lists event handlers with columns for DLL, Class Name, Company Name, Description, and Execution Type. The 'New' button is located at the top left of this table.

15. Find the newly created Event Handler associated to your application.

- Filter by Application Name.
- Select it from the list and click **Ok**.



16. Test your event handler by creating a new instance of your object type.

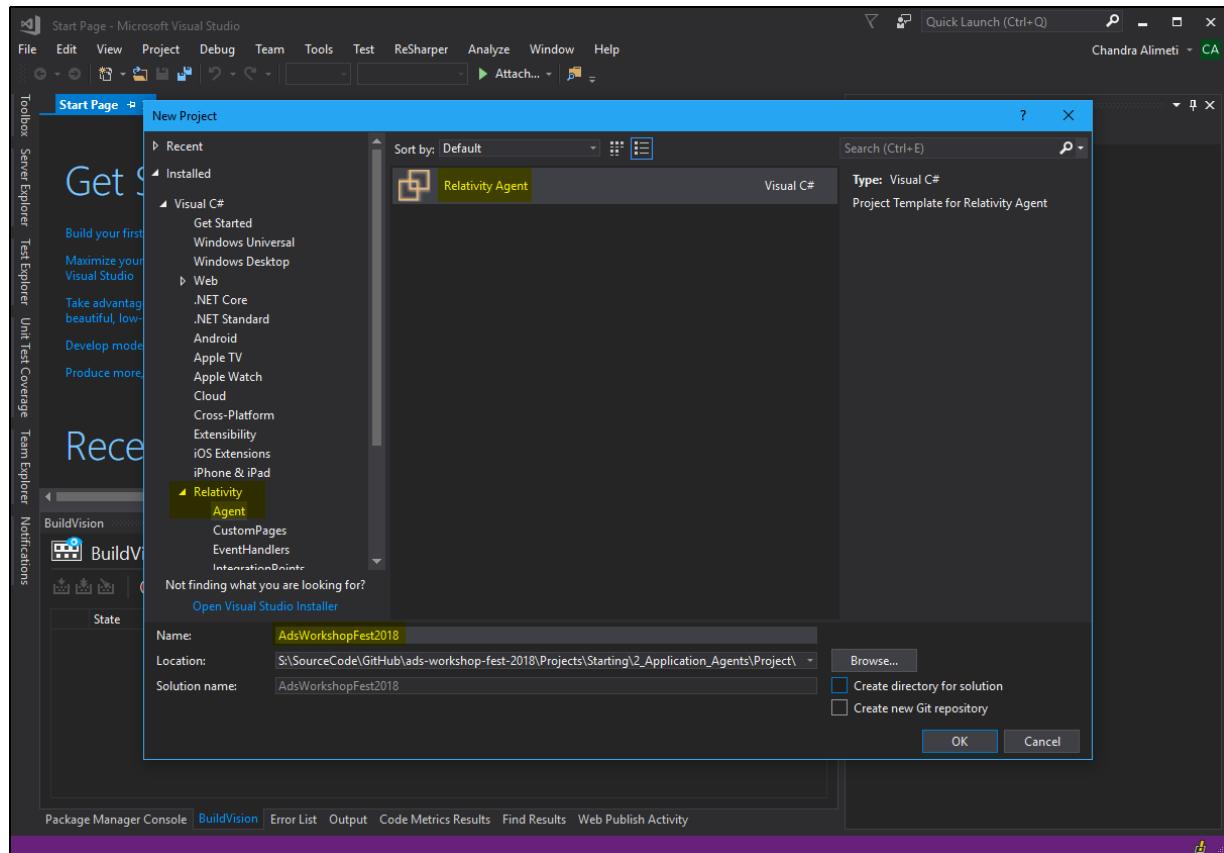


3.4 New Agent

Note: Documentation link for Agents -

https://platform.relativity.com/9.6/Content/Background_processing/Building_your_first_agent.htm?Highlight=agents

1. Create a new Visual Studio solution named **AdsWorkshopFest2018** in the **E:\Github\ads-workshop-fest-2018\Projects\Starting\2_Application_Agents\Project** folder using the **Relativity Agent** template.



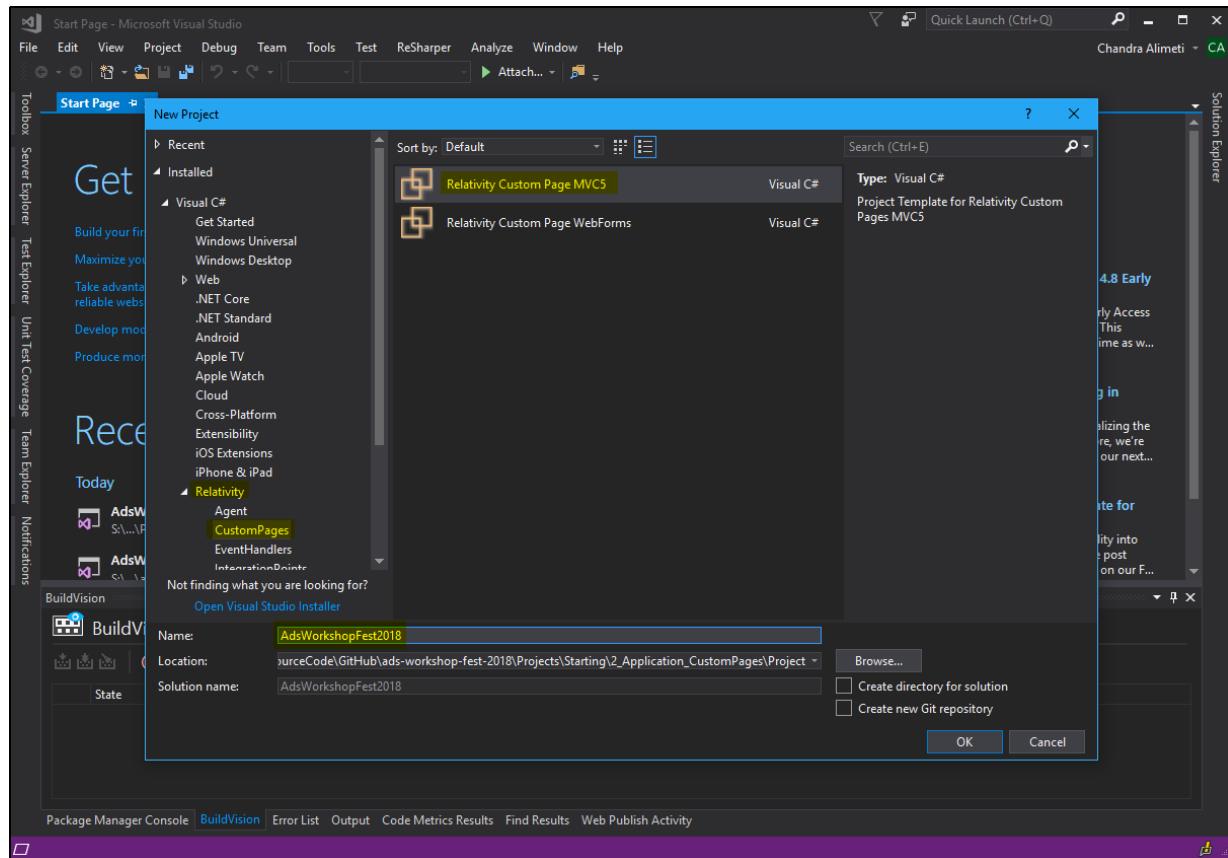
2. Open the **Relativity Agent.cs** class and take a look at the **Execute()** method where you will be adding business logic for your custom agents. It already comes with some commented code which is some of the most commonly used code in Agents to jump start your development.
3. Build the solution you just created in Visual Studio.
4. Use the Publish to Relativity tool you used in the previous section to add the Agents project DLL and then push it to the environment.
5. Once the changes have been published, go to the **Agents** tab to see the current time being displayed which is refreshed based on the agent interval.

3.5 New CustomPage

Note: Documentation link for Custom pages -

[https://platform.relativity.com/9.6/Content/Customizing_the_UI/Building_your_first_custom_page.htm
?Highlight=custom%20page](https://platform.relativity.com/9.6/Content/Customizing_the_UI/Building_your_first_custom_page.htm?Highlight=custom%20page)

1. Create a new Visual Studio solution named **AdsWorkshopFest2018** in the **E:\Github\ads-workshop-fest-2018\Projects\Starting\2_Application_CustomPages\Project** folder using the **Relativity Custom Page MVC5** template.



2. Open the **HomeController.cs** class under the **Controllers** folder and take a look at the **Index()** method containing some commented code which is some of the most commonly used Helper code in Custom pages to jump start your development.
3. Build the solution you just created in Visual Studio.
4. Let's add some custom code to count the number of workspaces in DevVM.

- Add the following code in the using statement for IRSAPIClient in the **Index()** method in **HomeController.cs** file under **Controllers** folder.

Code to add:

```
/Setting up an RSAPI Client

using (IRSAPIClient rsapiClient = Relativity.CustomPages.ConnectionHelper.Helper().
GetServicesManager().CreateProxy<IRSAPIClient>(ExecutionIdentity.CurrentUser))
{
    //Set the proxy to use the Admin case
    rsapiClient.APIOptions.WorkspaceID = -1;

    kCura.Reliability.Client.DTOs.Query<kCura.Reliability.Client.DTOs.Workspace> workspaceQuery =
        new kCura.Reliability.Client.DTOs.Query<kCura.Reliability.Client.DTOs.Workspace>();

    workspaceQuery.Fields.Add(new kCura.Reliability.Client.DTOs.FieldValue(kCura.Reliability.Client.DTOs.WorkspaceFieldNames.TextIdentifier));

    kCura.Reliability.Client.DTOs.QueryResultSet<kCura.Reliability.Client.DTOs.Workspace>
    workspaceQueryResultSet = new kCura.Reliability.Client.DTOs.QueryResultSet<kCura.Reliability.Client.DTOs.Workspace>();

    try
    {

        workspaceQueryResultSet = rsapiClient.Repositories.Workspace.Query(workspaceQuery,
0);
    }
    catch (Exception ex)
    {

        throw new Exception("An error occurred when querying for workspaces. Query.", ex);
    }

    int workspaceCount = workspaceQueryResultSet.Results.Count;
    ViewBag.WorkspaceCount = workspaceCount;
}
```

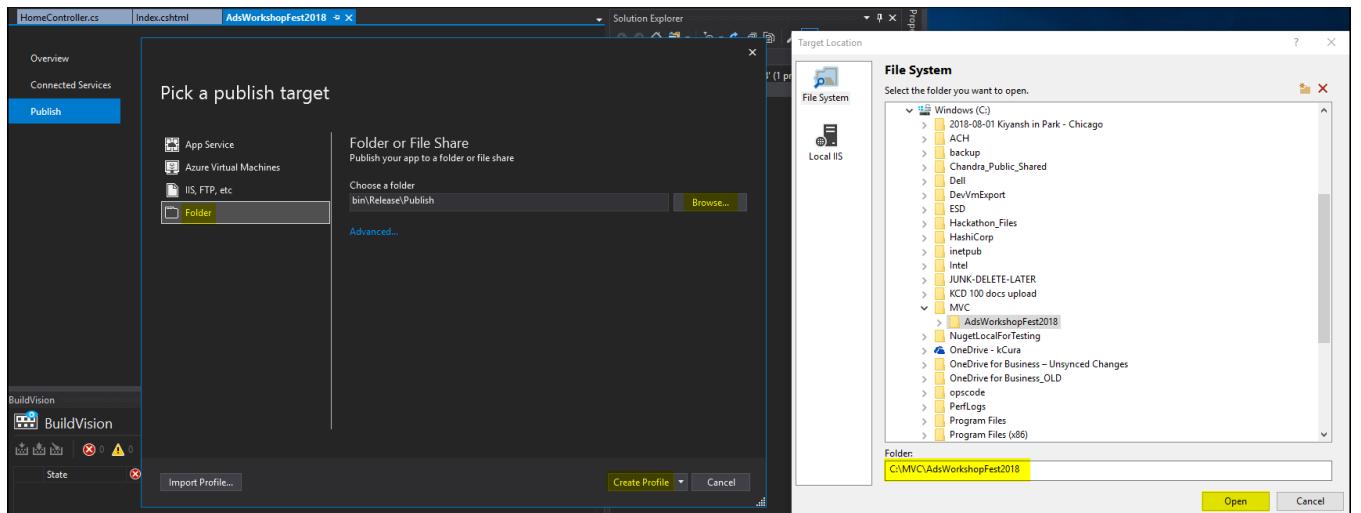
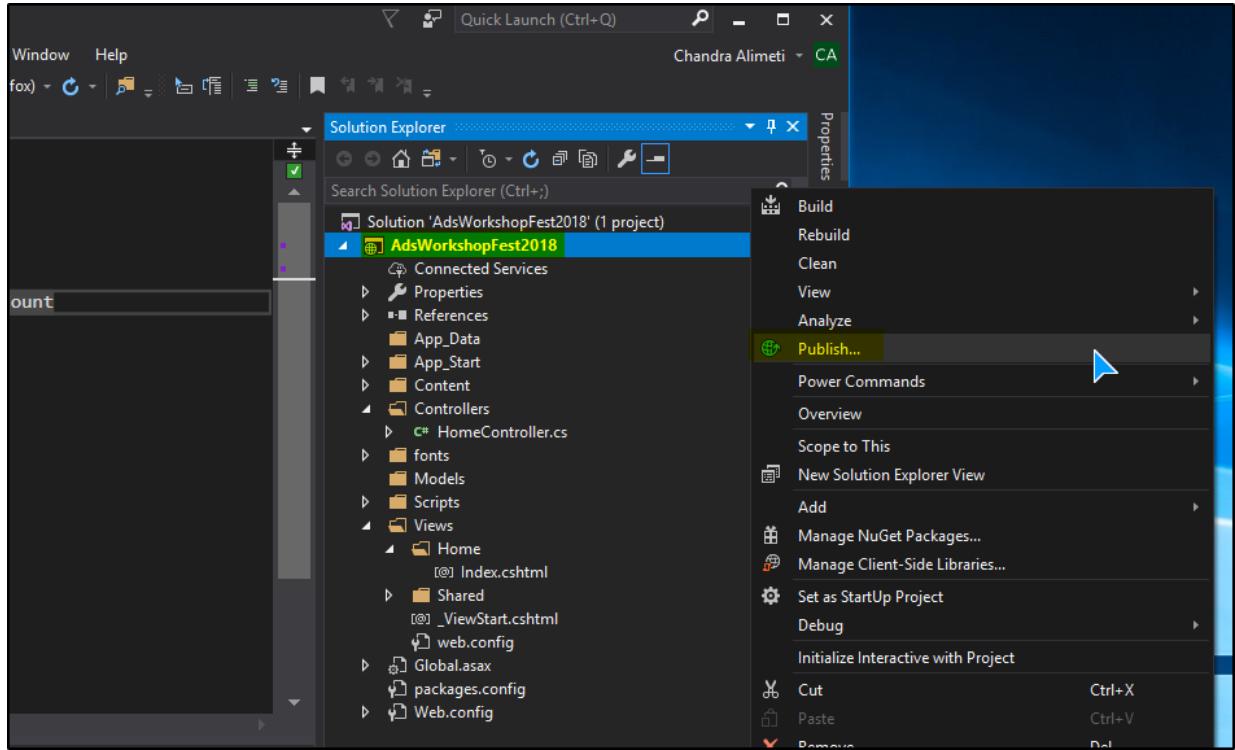
- Add the following code to the **Index.cshtml** file under **Views/Home** folder.

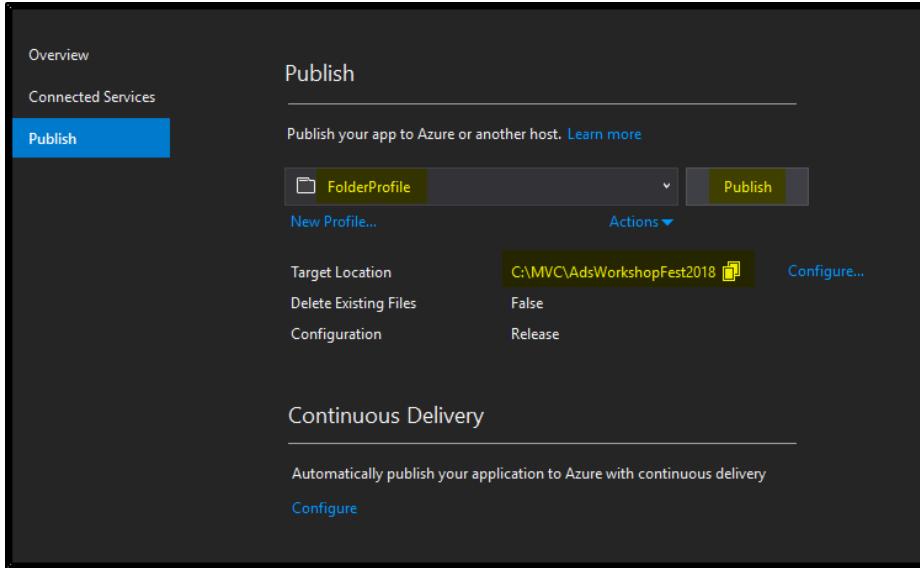
Code to add:

```
<label>WorkspaceCount:</label> @ViewBag.WorkspaceCount
```

- Build the solution you just created in Visual Studio.
- Create a folder (**C:\MVC\AdsWorkshopFest2018**) in C drive to publish the custom page code.

9. Publish the custom page to the folder created above using the **Folder** option.





10. Add the Custom page to your custom application in the workspace from front-end Relativity.
11. Once attached to the application, use the **Push to Library** button to deploy the custom page on the Relativity web server.
12. Create a new tab for the custom page using the below path.
%applicationPath%/CustomPages/<YourRelativityApplicationGUID>/Home/Index
13. Link the tab to your custom application
14. Navigate to the custom page tab to see its contents.

4 Sample Application Overview

The functionality of the Sample Relativity application (ADS Workshop Fest 2018) is to calculate the selected Instance metrics and write them to the Instance Metrics Job.

The ADS Workshop Fest 2018 Relativity application consists of the following:

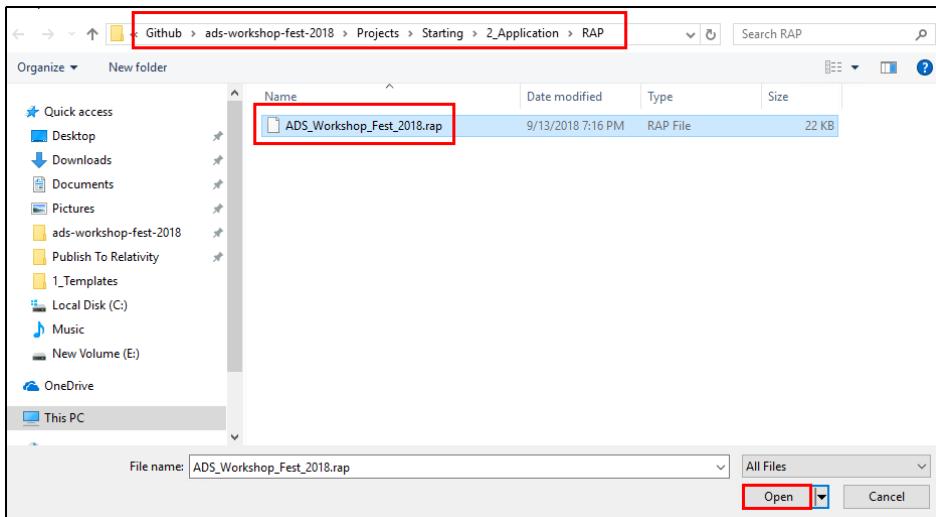
- **Custom Object**
 - Instance Metrics Job
 - Fields
 - Metrics
 - Status
 - Workspaces Count
 - Users Count
 - Groups Count
 - Errors
- **Event Handler**
 - PreSaveStatusUpdate
- **Agent**
 - Instance Metrics Calculator Agent

4.1 Install Sample Application

1. Go to the **Application Library** tab.
2. Click on the **Upload Application** button

#	Artifact ID	Name	Version	Last Modified On
1	1017170	MyNewApp	0.0.0.1	09/13/2018 11:36 PM CDT
2	1017110	Data Sampler	2.8.0.2	06/12/2018 11:47 AM CDT
3	1017099	Smoke Test	2.0.0.30	06/12/2018 11:46 AM CDT

3. Click on **Choose File** button.
4. Navigate to **E:\Github\ads-workshop-fest-2018\Projects\Starting\2_Application\RAP** folder and select the **ADS_Workshop_Fest_2018.rap** file.



5. Click the **Save** button.
6. Under the **Workspaces Installed** section, click the **Install** button.
7. Select the **ADS Workshop Fest 2018** workspace and click the **Save** button.

Workspace	Version	Status	Message	Case Artifact ID	Installed By	Installed On
ADS Workshop Fest 2018	0.0.23	Installed			1017168 Admin, Relativity	9/14/2018 12:22 AM CDT
Admin Case	0.0.23	Installed			-1 Admin, Relativity	9/14/2018 12:21 AM CDT

4.2 Run Sample Application

- Once the application is installed, navigate to the **Agents** tab and create an **Instance Metrics Calculator Agent** agent.

The screenshot shows the Relativity 9 interface with the 'Agents' tab selected. A modal window titled 'Agent Information' is open, containing fields for 'Agent Type' (set to 'Instance Metrics Calculator Agent'), 'Number of Agents' (1), 'Agent Server' (RELATIVITYDEVVM), 'Run interval' (10 seconds), and 'Logging level of event details' (Log critical errors only). Below these are sections for 'Status' (Enabled: Yes) and 'Other' (Keywords: empty, Notes: empty). At the top right of the modal are 'Save', 'Save and New', 'Save and Back', and 'Cancel' buttons, with 'Save' highlighted. A secondary modal window titled 'Select Agent Type' is displayed, listing agents by name, company name, and application name. The 'Instance Metrics Calculator Agent' is selected and highlighted with a red box. The 'Application Name' column shows 'ads' for this agent.

The screenshot shows the Relativity 9 interface with the 'Agents' tab selected. A table lists four agents: 'Instance Metrics Calculator Agent (1)', 'Data Grid Migration Manager (1)', 'Data Grid Migration Worker (1)', and 'Data Grid Ingestion Agent (1)'. The 'Name' column for the first agent is highlighted with a red box. The table includes columns for '#', 'Artifact ID', 'Name', 'Enabled', and 'Application'. The 'Enabled' column shows 'Yes' for all agents, and the 'Application' column shows 'ADS Workshop Fest 2018' for the first agent.

- Now navigate the **Instance Metrics Job** tab in **ADS Workshop Fest 2018** workspace.
- Create a new **Instance Metrics Job** and click on the **Save** button.

Instance Metrics Job

Instance Metrics Job Layout ▾

Save **Save and New** **Save and Back** **Cancel**

Default Category

Name:	Job1
Metrics:	<input checked="" type="checkbox"/> Workspaces <input checked="" type="checkbox"/> Users <input checked="" type="checkbox"/> Groups <input type="checkbox"/> Manage
Status:	
Errors:	

Statistics

Workspaces Count:	
Users Count:	
Groups Count:	

Status

System Created By:		System Created On:	
System Last Modified By:		System Last Modified On:	

Instance Metrics Job

Instance Metrics Job Layout ▾

Edit **Delete** **Back** **Edit**

Default Category

Name:	Job1
Metrics:	Workspaces; Users; Groups;
Status:	New
Errors:	

Statistics

Workspaces Count:	
Users Count:	
Groups Count:	

4. The **Instance Metrics Calculator Agent** which is running every 10 seconds will pick up the job and calculate the metrics and save it to the job.
5. Keeping refreshing the windows every few seconds until the job status is set to **Completed**.

Instance Metrics Job

Instance Metrics Job Layout

Edit Delete Back Edit Permissions View Audit

Default Category

Name: Job1
Metrics: Workspaces; Users; Groups;
Status: **Completed**
Errors:

Statistics

Workspaces Count: 7
Users Count: 2
Groups Count: 9

Status

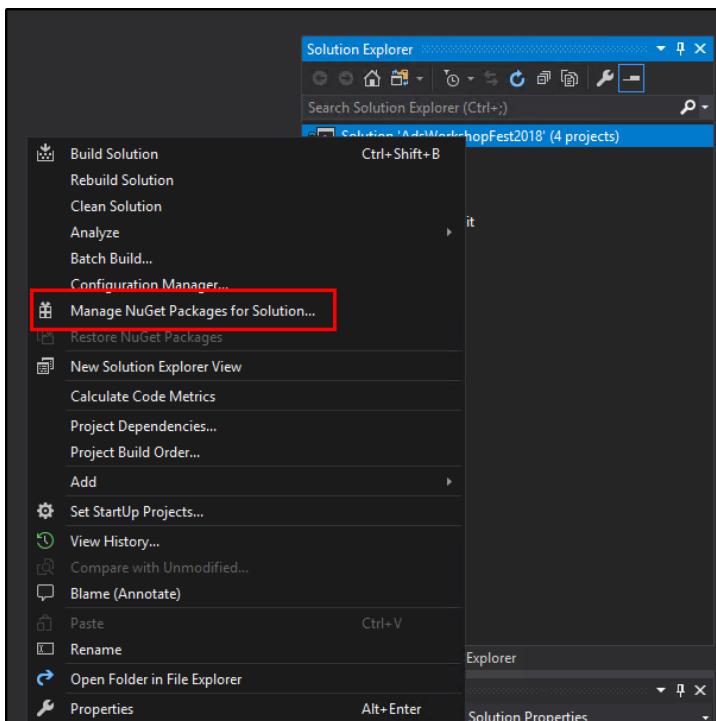
System Created By: Admin, Relativity
System Last Modified By: Service Account, Relativity

System Created On: 9/14/2018 12:29 AM
System Last Modified On: 9/14/2018 12:30 AM

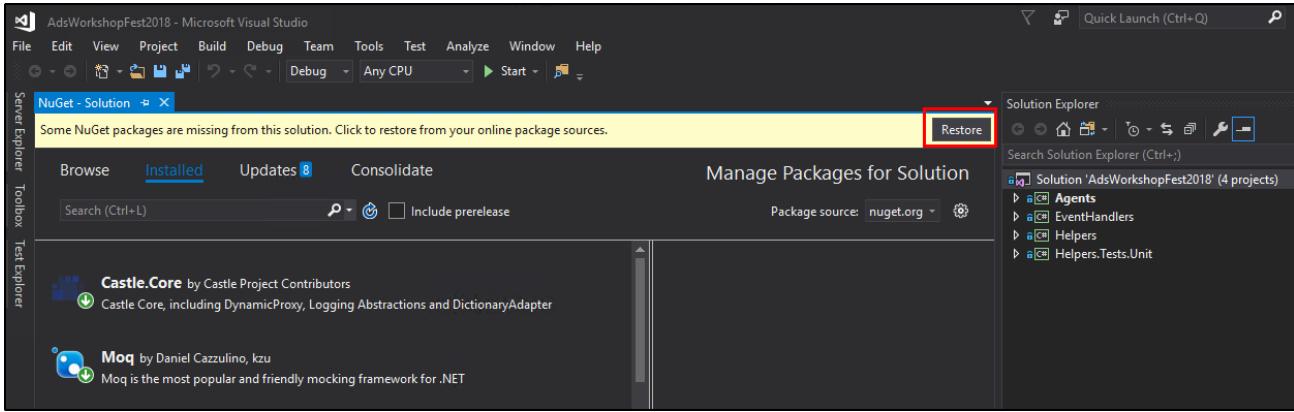
5 Unit Tests

5.1 Writing Unit Tests

1. Navigate to the `E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\Project` folder and open `AdsWorkshopFest2018.sln` file.
2. In this solution, you will find 4 projects.
 - a. Agents
 - b. EventHandlers
 - c. Helpers
 - d. Helpers.Tests.Unit
3. In this section we will be working in the `Helpers.Tests.Unit` project.
4. In `Helpers` project, there is an `RsapiHelper.cs` file which contains RSAPI calls used in the sample application.
5. Our goal for this section is to write couple unit tests for the `RetrieveJobsInWorkspaceWithStatus` in `RsapiHelper` class.
6. Right click on the solution and select **Manage NuGet Packages for Solution** option.



7. Next click the **Restore** button.



8. Now all the NuGet packages in the solution should be restored.
9. Right click on the **Helpers.Tests.Unit** project and add a new class named **RsapiHelperTests.cs**.
10. Overwrite the default class structure with the Unit Test structure we will be using.
11. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\1_Structure.txt" file replacing the entire **RsapiHelperTests.cs** class.

```

RsapiHelperTests.cs
1  using Kcura.Relativity.Client;
2  using Kcura.Relativity.Client.DTOs;
3  using Kcura.Relativity.Client.Repositories;
4  using Moq;
5  using NUnit.Framework;
6  using System;
7  using System.Collections.Generic;
8  using Choice = Kcura.Relativity.Client.DTOs.Choice;
9  using User = Kcura.Relativity.Client.DTOs.User;
10
11 namespace Helpers.Tests.Unit
12 {
13     [TestFixture]
14     [Description("Rsapi Helper Tests")]
15     public class RsapiHelperTests
16     {
17         #region SetUp and TearDown
18
19         #endregion
20
21         #region TestConstants
22
23         #endregion
24
25         #region Tests
26
27         #endregion
28
29         #region Mocks
30
31         #endregion
32
33         #region Verify
34
35         #endregion
36
37         #endregion
38
39     }
40

```

12. Add the **RsapiHelper** class reference which we will be our System under Test (SUT) for which we are writing unit tests.
13. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\2_System_Under_Test.txt" file at the beginning of the **RsapiHelperTests.cs** class.

```

[TestFixture]
[Description("Rsapi Helper Tests")]
public class RsapiHelperTests
{
    public RsapiHelper Sut { get; set; }

```

14. Next add the Mocks for various RDO repositories which are used in the Relativity application.
15. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\3_Mocks.txt" file at the beginning of the **RsapiHelperTests.cs** class.

```

[TestFixture]
[Description("Rsapi Helper Tests")]
public class RsapiHelperTests
{
    public RsapiHelper Sut { get; set; }
    public Mock<IGenericRepository<RDO>> MockRdoRepository { get; set; }
    public Mock<IGenericRepository<Choice>> MockChoiceRepository { get; set; }
    public Mock<IGenericRepository<Workspace>> MockWorkspaceRepository { get; set; }
    public Mock<IGenericRepository<User>> MockUserRepository { get; set; }
    public Mock<IGenericRepository<Group>> MockGroupRepository { get; set; }

```

16. Next add the SetUp and TearDown methods which run once before and after all the tests execution.
17. The SetUp method is utilized to initialize any data needed by tests.
18. The TearDown method is utilized to clean up the data used by the tests.
19. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\4_SetUp_And_TearDown.txt" file under the **SetUp** and **TearDown** region in the **RsapiHelperTests.cs** class.

```

#region SetUp and TearDown

[SetUp]
public void SetUp()
{
    MockRdoRepository = new Mock<IGenericRepository<RDO>>();
    MockChoiceRepository = new Mock<IGenericRepository<Choice>>();
    MockWorkspaceRepository = new Mock<IGenericRepository<Workspace>>();
    MockUserRepository = new Mock<IGenericRepository<User>>();
    MockGroupRepository = new Mock<IGenericRepository<Group>>();
    APIOptions rsapiApiOptions = new APIOptions
    {
        WorkspaceID = -1
    };

    Sut = new RsapiHelper(
        rsapiApiOptions: rsapiApiOptions,
        rdoRepository: MockRdoRepository.Object,
        choiceRepository: MockChoiceRepository.Object,
        workspaceRepository: MockWorkspaceRepository.Object,
        userRepository: MockUserRepository.Object,
        groupRepository: MockGroupRepository.Object);
}

[TearDown]
public void TearDown()
{
    MockRdoRepository = null;
    MockChoiceRepository = null;
    MockWorkspaceRepository = null;
    MockUserRepository = null;
    MockGroupRepository = null;
    Sut = null;
}

#endregion

```

20. Next add the Test constants used in the Tests.
21. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\5_Test_Constants.txt" file under the **TestConstants** region in the **RsapiHelperTests.cs** class.

```
#region TestConstants
private const int TestWorkspaceArtifactId = 123;
private const string TestStatus = "New";
#endregion
```

22. Next add the Mock Stubs for an RSAPI call used in the Tests. Here we will be stubbing out the Query method on the RDO repository in RSAPI.
23. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\6_Mock_Stub_Golden_Flow.txt" file under the **Mocks** region in the **RsapiHelperTests.cs** class.

```
#region Mocks
private void Mock_RdoRepository_Query_Works(int rdoCount)
{
    List<Result<RDO>> results = new List<Result<RDO>>();
    for (int i = 1; i <= rdoCount; i++)
    {
        Result<RDO> newResult = new Result<RDO>
        {
            Artifact = new RDO(i),
            Message = string.Empty,
            Success = true
        };
        results.Add(newResult);
    }
    QueryResultSet<RDO> rdoQueryResultSet = new QueryResultSet<RDO>
    {
        Success = true,
        Results = results
    };

    MockRdoRepository
        .Setup(x => x.Query(It.IsAny<Query<RDO>>(), It.IsAny<int>()))
        .Returns(rdoQueryResultSet);
}
```

24. Next add the Verify method for the previously added Mock Stubs. Use this method we can verify if our stub was called during our test execution.
25. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\7_Mock_Stub_Golden_Flow_Verify.txt" file under the **Verify** region in the **RsapiHelperTests.cs** class.

```
#region Verify
private void Verify_RdoRepository_Query_Works_Was_Called(int timesCalled)
{
    MockRdoRepository
        .Verify(x => x.Query(It.IsAny<Query<RDO>>(), It.IsAny<int>()),
               , Times.Exactly(timesCalled));
}

#endregion
```

26. Next add the Golden flow test.

27. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\8_UnitTest_Golden_Flow.txt" file under the **Tests** region in the **RsapiHelperTests.cs** class.

```
#region Tests

[Test]
public void RetrieveJobsInWorkspaceWithStatus_GoldenFlow()
{
    //Arrange
    int rdoCount = 5;
    Mock_RdoRepository_Query_Works(rdoCount);

    //Act
    List<int> jobsList = Sut.RetrieveJobsInWorkspaceWithStatus(TestWorkspaceArtifactId, TestStatus);

    //Assert
    Verify_RdoRepository_Query_Works_Was_Called(1);
    Assert.That(jobsList.Count, Is.EqualTo(rdoCount));
}
```

28. Next add the Mock Stubs for an RSAPI call to simulate an exception in the RSAPI call.
29. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\9_Mock_Stub_Rsapi_Fails.txt" file under the **Mocks** region in the **RsapiHelperTests.cs** class.

```
private void Mock_RdoRepository_Query_Throws_Exception()
{
    MockRdoRepository
        .Setup(x => x.Query(It.IsAny<Query<RDO>>(), It.IsAny<int>()))
        .Throws<Exception>();
}

#endregion
```

30. Next add the RSAPI Failure test.
31. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\3_UnitTests\TextFiles\10_UnitTest_Rsapi_Fails.txt" file under the **Tests** region in the **RsapiHelperTests.cs** class.

```
[Test]
public void RetrieveJobsInWorkspaceWithStatus_Rsapi_Fails()
{
    //Arrange
    Mock_RdoRepository_Query_Throws_Exception();

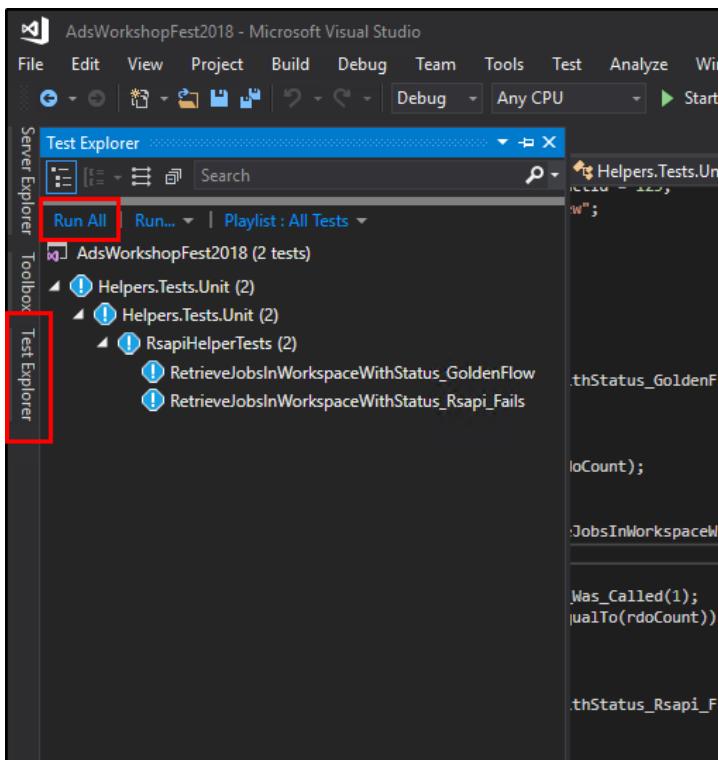
    //Act
    Exception exception = Assert.Throws<Exception>(() => Sut.RetrieveJobsInWorkspaceWithStatus(TestWorkspaceArtifactId, TestStatus));

    //Assert
    StringAssert.Contains(Constants.ErrorMessages.QUERY_APPLICATION_JOBS_ERROR, exception.ToString());
    Verify_RdoRepository_Query_Works_Was_Called(1);
}

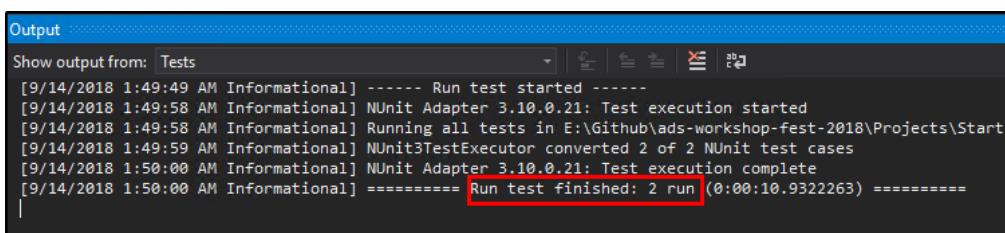
#endregion
```

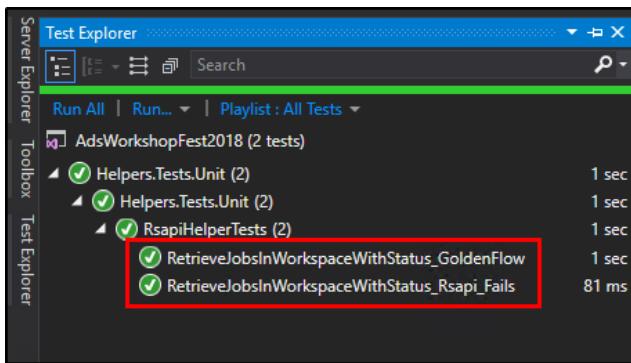
5.2 Running Unit Tests

1. Build the solution to successfully build all the projects.
2. Open the **Test Explorer** by click on the **Test Explorer** tab on the left side bar of Visual Studio.
3. Click on the **Run All** link to run the 2 unit tests we just created.



4. On successful run, click on the **Test Explorer** to see the status of the tests. Both the tests should be green.

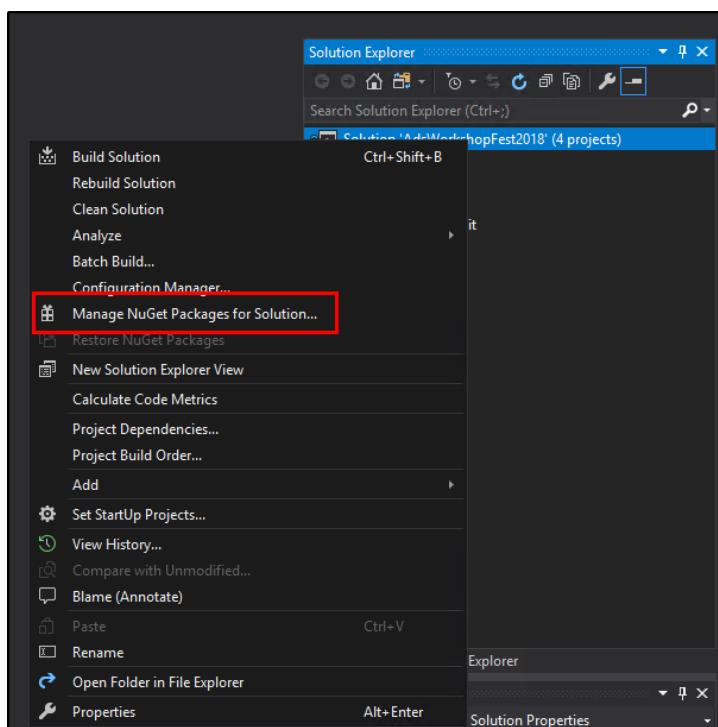




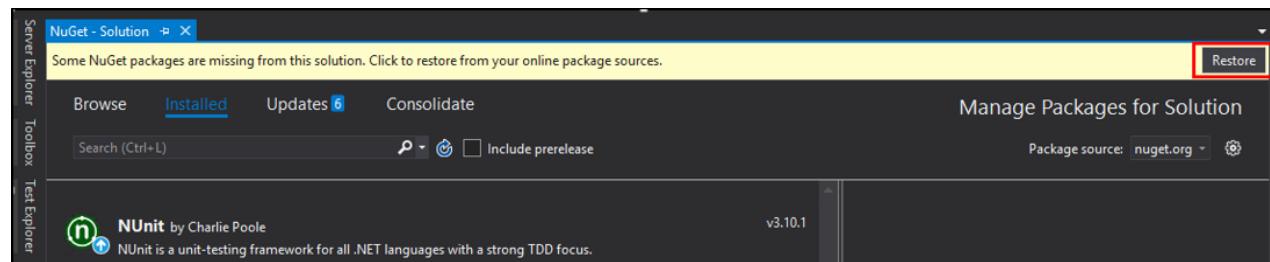
6 Integration Tests

6.1 Writing Integration Tests

1. Navigate to the E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project folder and open **AdsWorkshopFest2018.sln** file.
2. In this solution, you will find 4 projects.
 - a. Agents
 - b. Agents.Tests.Integration
 - c. EventHandlers
 - d. Helpers
3. In this section we will be working in the **Agents.Tests.Integration** project.
4. Right click on the solution and select **Manage NuGet Packages for Solution** option.



5. Next click the **Restore** button.



6. Now all the NuGet packages in the solution should be restored.
7. Right click on the **Agents.Tests.Integration** project and add a new class named **AgentsTests.cs**.

8. Overwrite the default class structure with the Integration Test structure we will be using.
9. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\TextFiles\1_Structure.txt" file replacing the entire AgentsTests.cs class.

```

AgentTests.cs NuGet - Solution
Agents.Tests.Integration
1  using kCura.Relativity.Client;
2  using kCura.Relativity.Client.DTOs;
3  using NUnit.Framework;
4  using Relativity.API;
5  using Relativity.Services.ServiceProxy;
6  using Relativity.Test.Helpers;
7  using System;
8  using System.Collections.Generic;
9  using System.Threading;
10 using UsernamePasswordCredentials = Relativity.Services.ServiceProxy.UsernamePasswordCredentials;
11
12 namespace Agents.Tests.Integration
13 {
14     [TestFixture]
15     [Description("Agent Tests")]
16     public class AgentTests
17     {
18         private ServiceFactory _serviceFactory;
19         private IServicesMgr _servicesManager;
20         private IRSAPIClient _rsapiClient;
21         private IDBContext _edsDbContext;
22         private AgentUtility _agentUtility;
23         private TestHelper _testHelper;
24         private int _workspaceArtifactId;
25     }
26 }
27

```

10. Add the **OneTimeSetUp** method and its dependent methods to the **AgentsTests** class.
11. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\TextFiles\2_OneTimeSetUp.txt" to the **AgentsTests** class.

```

[OneTimeSetUp]
public void OneTimeSetUp()...

private void SetupTestVariables()...
private void SetupApiEndpoints()...
private int CreateWorkspace(string workspaceName)...
private void InstallApplicationInWorkspace()...

```

12. Add the **OneTimeTearDown** method and its dependent methods to the **AgentsTests** class.
13. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\TextFiles\3_OneTimeTearDown.txt" to the **AgentsTests** class.

```

[OneTimeTearDown]
public void OneTimeTearDown()...

private bool DoesWorkspaceExists(int workspaceArtifactId)...
private void DeleteWorkspace(int? workspaceArtifactId)...

```

14. Add the **Job_ShouldBePickedUp_WhenStatusIsSetToNew_And_RunSuccessfully** integration test method and its dependent methods to the **AgentsTests** class.

15. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\TextFiles\4_Job_ShouldBePickedUp_WhenStatusIsSetToNew_And_RunSuccessfully.txt" to the **AgentsTests** class.

```
[Test]
[Description("Job_ShouldBePickedUp_WhenStatusIsSetToNew_And_RunSuccessfully")]
public void Job_ShouldBePickedUp_WhenStatusIsSetToNew_And_RunSuccessfully()...

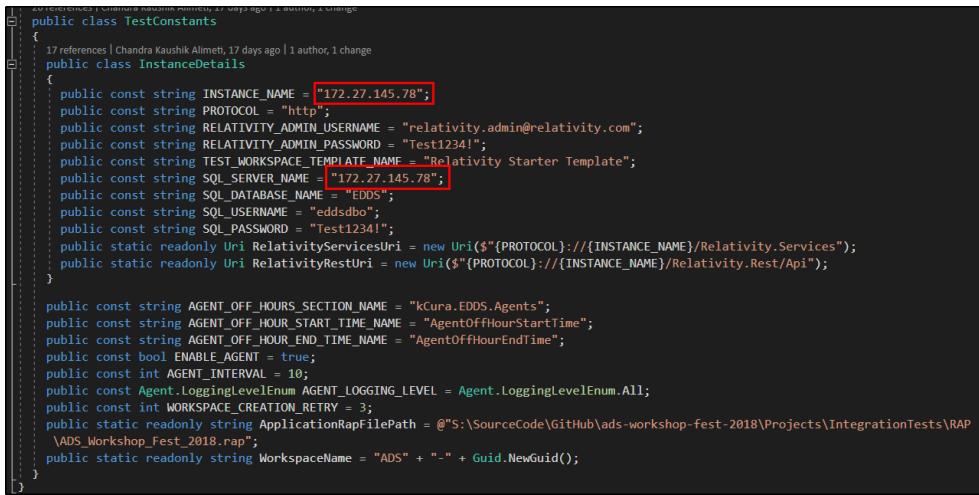
private int CreateUser()...
private bool DoesUserExists(int userArtifactId)...
private void DeleteUser(int? userArtifactId)...
private int CreateGroup(string groupName)...
private bool DoesGroupExists(int groupArtifactId)...
private void DeleteGroup(int? groupArtifactId)...
private int CreateJob(string jobStatus)...
private bool DoesJobExists(int jobArtifactId)...
private void DeleteJob(int? jobArtifactId)...
private string RetrieveJobStatus(int jobArtifactId)...
```

16. Add the **Job_ShouldNotBePickedUp_WhenStatusIsNotSetToNew** integration test method and its dependent methods to the **AgentsTests** class.
17. Copy and paste code from "E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\TextFiles\5_Job_ShouldNotBePickedUp_WhenStatusIsNotSetToNew.txt" to the **AgentsTests** class.

```
[Test]
[Description("Job_ShouldNotBePickedUp_WhenStatusIsNotSetToNew")]
public void Job_ShouldNotBePickedUp_WhenStatusIsNotSetToNew()...
```

6.2 Running Integration Tests

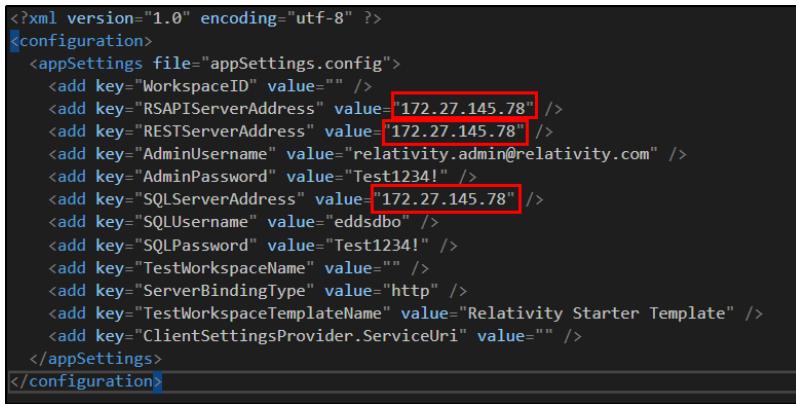
1. Open **TestConstants.cs** file in **Agents.Tests.Integration** project.
2. Replace the **192.168.137.95** IP address with the DevVM IP address.



```
public class TestConstants
{
    public class InstanceDetails
    {
        public const string INSTANCE_NAME = "172.27.145.78";
        public const string PROTOCOL = "http";
        public const string RELATIVITY_ADMIN_USERNAME = "relativity.admin@relativity.com";
        public const string RELATIVITY_ADMIN_PASSWORD = "Test1234!";
        public const string TEST_WORKSPACE_TEMPLATE_NAME = "Relativity Starter Template";
        public const string SQL_SERVER_NAME = "172.27.145.78";
        public const string SQL_DATABASE_NAME = "EDDS";
        public const string SQL_USERNAME = "eddsdbo";
        public const string SQL_PASSWORD = "Test1234!";
        public static readonly Uri RelativityServicesUri = new Uri($"{PROTOCOL}://{INSTANCE_NAME}/Relativity.Services");
        public static readonly Uri RelativityRestUri = new Uri($"{PROTOCOL}://{INSTANCE_NAME}/Relativity.Rest/Api");

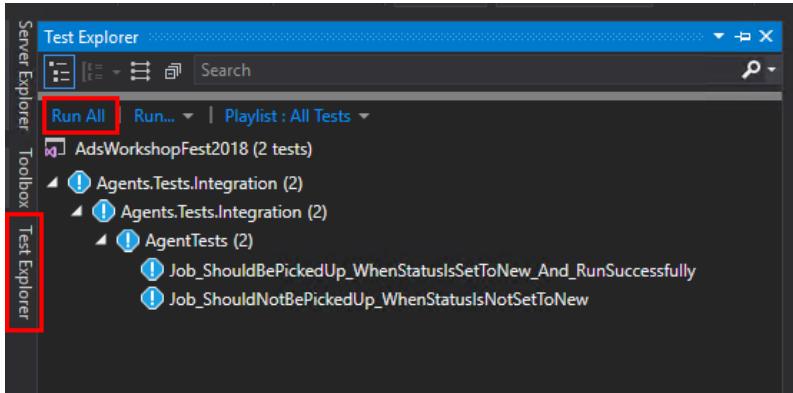
        public const string AGENT_OFF_HOURS_SECTION_NAME = "kCurA.EDDS.Agents";
        public const string AGENT_OFF_HOUR_START_TIME_NAME = "AgentOffHourStartTime";
        public const string AGENT_OFF_HOUR_END_TIME_NAME = "AgentOffHourEndTime";
        public const bool ENABLE_AGENT = true;
        public const int AGENT_INTERVAL = 10;
        public const Agent.LoggingLevelEnum AGENT_LOGGING_LEVEL = Agent.LoggingLevelEnum.All;
        public const int WORKSPACE_CREATION_RETRY = 3;
        public static readonly string ApplicationRapFilePath = @"S:\SourceCode\GitHub\ads-workshop-fest-2018\Projects\IntegrationTests\RAP\ADS_Workshop_Fest_2018.rap";
        public static readonly string WorkspaceName = "ADS" + "-" + Guid.NewGuid();
    }
}
```

3. Open **App.config** file in **Agents.Tests.Integration** project.
4. Replace the **192.168.137.95** IP address with the DevVM IP address.

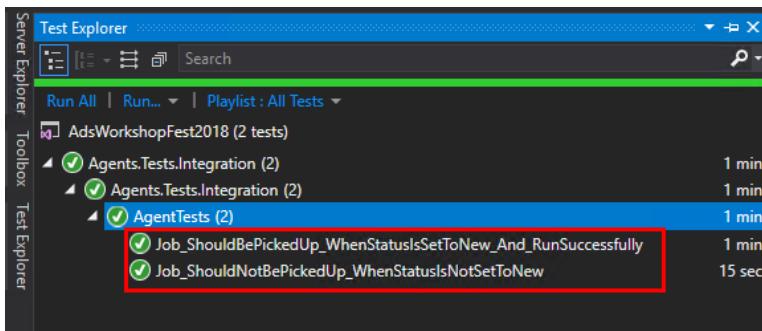
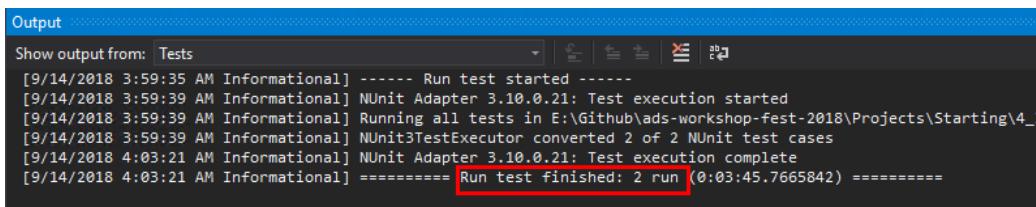


```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <appSettings file="appSettings.config">
        <add key="WorkspaceID" value="" />
        <add key="RSAPIServerAddress" value="172.27.145.78" />
        <add key="RESTServerAddress" value="172.27.145.78" />
        <add key="AdminUsername" value="relativity.admin@relativity.com" />
        <add key="AdminPassword" value="Test1234!" />
        <add key="SQLServerAddress" value="172.27.145.78" />
        <add key="SQLUsername" value="eddsdbo" />
        <add key="SQLPassword" value="test1234!" />
        <add key="TestWorkspaceName" value="" />
        <add key="ServerBindingType" value="http" />
        <add key="TestWorkspaceTemplateName" value="Relativity Starter Template" />
        <add key="ClientSettingsProvider.ServiceUri" value="" />
    </appSettings>
</configuration>
```

5. Build the solution to successfully build all the projects.
6. Open the **Test Explorer** by click on the **Test Explorer** tab on the left side bar of Visual Studio.
7. Click on the **Run All** link to run the 2 unit tests we just created.



8. On successful run, click on the **Test Explorer** to see the status of the tests. Both the tests should be green.



6.2 Running Integration Tests as CI

1. Open the E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\CI_Scripts folder
2. Here you will find 3 files and 1 folder which we will be using to run Integration Tests as a CI (Continuous Integration) process.
3. Copy all the files and folders from the CI_Scripts folder to E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project folder.

(E:) > Github > ads-workshop-fest-2018 > Projects > Starting > 4_IntegrationTests > Project >			
Name	Date modified	Type	Size
.vs	9/14/2018 3:36 AM	File folder	
Agents	9/14/2018 3:36 AM	File folder	
Agents.Tests.Integration	9/14/2018 3:56 AM	File folder	
EventHandlers	9/14/2018 3:36 AM	File folder	
Helpers	9/14/2018 3:36 AM	File folder	
packages	9/14/2018 3:36 AM	File folder	
AdsWorkshopFest2018.sln	9/13/2018 7:16 PM	Visual Studio Solu...	3 KB

(E:) > Github > ads-workshop-fest-2018 > Projects > Starting > 4_IntegrationTests > Project >			
Name	Date modified	Type	Size
.vs	9/14/2018 3:36 AM	File folder	
Agents	9/14/2018 3:36 AM	File folder	
Agents.Tests.Integration	9/14/2018 3:56 AM	File folder	
buildtools	9/14/2018 4:11 AM	File folder	
EventHandlers	9/14/2018 3:36 AM	File folder	
Helpers	9/14/2018 3:36 AM	File folder	
packages	9/14/2018 3:36 AM	File folder	
AdsWorkshopFest2018.sln	9/13/2018 7:16 PM	Visual Studio Solu...	3 KB
Build.ps1	9/13/2018 7:27 PM	Windows PowerS...	3 KB
defaultBuildTest.ps1	9/13/2018 7:27 PM	Windows PowerS...	5 KB

4. Open the **Build.ps1** script and update the **NuGet_URL** to use the one which is compatible with Visual Studio 2017.

```
#set up variables
$BASE_DIR = Resolve-Path .
Write-Verbose "BASE_DIR resolves to: $BASE_DIR"
#VS 2017
#$NUGET_URL = "https://dist.nuget.org/win-x86-commandline/latest/nuget.exe"
#VS2015
$NUGET_URL = "https://dist.nuget.org/win-x86-commandline/v3.3.0/nuget.exe"
Write-Verbose "nuget URL: $NUGET_URL"
```

```
#set up variables
$BASE_DIR = Resolve-Path .
Write-Verbose "BASE DIR resolves to: $BASE_DIR"
#VS 2017
#$NUGET_URL = "https://dist.nuget.org/win-x86-commandline/latest/nuget.exe"
#VS2015
#$NUGET_URL = "https://dist.nuget.org/win-x86-commandline/v3.3.0/nuget.exe"
Write-Verbose "nuget URL: $NUGET_URL"
```

5. Open the **defaultBuildTest.ps1** script and update the project paths.

- a. Replace line 10 with the following code.

Code:

```
$solution = Join-Path $root "..\AdsWorkshopFest2018.sln"
```

Before:

10	\$solution = Join-Path \$root "..\RelativityAgent\RelativityAgent.sln"
----	--

After:

10	\$solution = Join-Path \$root "..\AdsWorkshopFest2018.sln"
----	--

- b. Replace line 15 with the following code.

Code:

```
$testAssembly = $testAssembly = Join-Path $root  
"..\Project\Agents.Tests.Integration\bin\Debug\Agents.Tests.Integration.dll"
```

Before:

15	\$testAssembly = \$testAssembly = Join-Path \$root "..\RelativityAgent\AgentNunitIntegrationTest\bin\Debug\AgentNunitIntegrationTest.dll"
----	--

After:

15	\$testAssembly = \$testAssembly = Join-Path \$root "..\Project\Agents.Tests.Integration\bin\Debug\Agents.Tests.Integration.dll"
----	--

- c. Comment lines 64 – 66.

64	# task UnitTest -Alias Test -Depends TestInitialize -Description "Run NUnit unit tests" {
65	# exec { & \$nunit_exe \$solution --where "class=~/.+\.UnitTests\...+\$/"
66	--result="\$test_logs\UnitTests.xml;format=nunit2" } -errorMessage "Unit tests failed!"

- d. Replace lines 69 - 74 with the following code.

Code:

```
$testDir = Join-Path $root "..\Agents.Tests.Integration"  
$testDir = Join-Path $root "..\Agents.Tests.Integration"  
$configSource = "..\Project\Agents.Tests.Integration\App.config"  
Write-Host "configSource is : $configSource"  
$configDestination = Join-Path $root  
"..\Project\Agents.Tests.Integration\bin\Debug\Agents.Tests.Integration.dll.config"  
Write-Host "Test assembly : $testAssembly"  
Copy-Item $configSource $configDestination -Verbose:$VerbosePreference
```

Before:

54	#Собираем проекты в папку с тестами - передаете путь к папке
55	Мы это делаем для : скомпилированного
56	"..\Библиотеки\Agents\Agents\bin\Debug\Agents.dll"
57	#Создаем папку для тестов = тот-то это
58	#Мы это делаем : скомпилированное
59	# скомпилированное = "..\Библиотеки\Agents\Agents\bin\Debug\Agents.dll"
60	# скомпилированное = тот-то это

After:

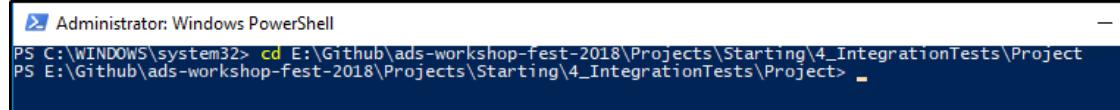
```
69 $testDir = Join-Path $root "..\Agents.Tests.Integration"
70 $configSource = "..\Project\Agents.Tests.Integration\App.config"
71 Write-Host "configSource is : $configSource"
72 $configDestination = Join-Path $root
73     "..\Project\Agents.Tests.Integration\bin\Debug\Agents.Tests.Integration.dll.config"
74 Copy-Item $configSource $configDestination -Verbose:$VerbosePreference
```

6. Open PowerShell and run the following command.

Set-ExecutionPolicy Unrestricted

7. When prompted type **A** and press **Enter**.
8. Close the current PowerShell window and open a new PowerShell window.
9. Navigate to “**E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project**” folder by running the following command.

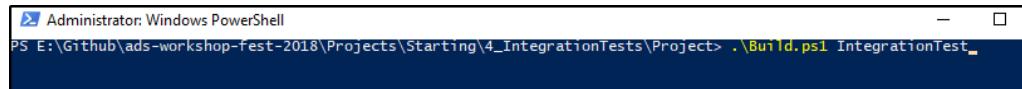
```
cd E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project
```



```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> cd E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project
PS E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project>
```

10. Run the following command to run the Integration Tests.

```
.\Build.ps1 IntegrationTest
```



```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> .\Build.ps1 IntegrationTest
PS E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project> .\Build.ps1 IntegrationTest
```

11. You will see the following output when the Integration tests are run successfully.

```

PS E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project> .\Build.ps1 IntegrationTest
Restoring tools from NuGet...
Using E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project\buildtools\packages.config...
Feeds used:
  C:\Users\csadmin\.nuget\packages\
  https://api.nuget.org/v3/index.json
  C:\Program Files (x86)\Microsoft SDKs\NuGetPackages\

All packages listed in E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project\buildtools\packages.config are already installed.
----- Executing Task: TestInitialize -----

  Directory: E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project\Artifacts

Mode          LastWriteTime      Length Name
----          -----        -----
d----  9/14/2018   4:45 AM           TestLogs
----- Executing Task: IntegrationTest -----
config source is : ..\Project\Agents.Tests.Integration\App.config
Test assembly : E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project\..\Project\Agents.Tests.Integration\bin\Debug\Agents.Tests.Integration.dll
NUnit Console Runner 3.6.0
Copyright (C) 2017 Charlie Poole

Runtime Environment
  OS Version: Microsoft Windows NT 10.0.17134.0
  CLR Version: 4.0.30319.42000

Test Files
  E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project\..\Project\Agents.Tests.Integration\bin\Debug\Agents.Tests.Integration.dll

=> Agents.Tests.Integration.AgentTests.Job_ShouldBePickedUp_WhenStatusIsSetToNew_And_RunSuccessfully
Start - ARRANGE
numberOfWorkspacesBeforeJobRun= 12
numberOfUsersBeforeJobRun= 2
numberOfGroupsBeforeJobRun= 9
Start - Create Workspace.
Creating workspace.
Workspace created [WorkspaceArtifactId= 1017194]
End - Create Workspace.
testWorkspaceArtifactId= 1017194
Start - Creating User.
Creating new User.
End - Creating User.
testUserArtifactId= 1017195
Start - Creating Group.
Creating new Group.
New Group Created.
End - Creating Group.
testGroupArtifactId= 1017197
End - ARRANGE
Start - ACT
Start - Creating Job.
End - Creating Job.
jobArtifactId= 1039422
attempt= 1
Start - Retrieving Job Status.
End - Retrieving Job Status.
jobStatus= New
attempt= 2
Start - Retrieving Job Status.
End - Retrieving Job Status.
jobStatus= Completed
numberOfWorkspacesAfterJobRun= 13
numberOfUsersAfterJobRun= 3
numberOfGroupsAfterJobRun= 10

```

```

numberOfGroupsAfterJobRun= 10
End - ACT
Start - ASSERT
End - ASSERT
Start - Clean up
Start - Deleting User.
Start - Checking if User exists.
End - Checking if User exists.
Deleting User.
User Deleted.
End - Deleting User.
Start - Deleting Group.
Start - Checking if Group exists.
End - Checking if Group exists.
Deleting Group.
Group Deleted.
End - Deleting Group.
Start - Deleting Job.
Start - Checking if Job exists.
End - Checking if Job exists.
End - Deleting Job.
End - Clean up
=> Agents.Tests.Integration.AgentTests.Job_ShouldNotBePickedUp_WhenStatusIsNotSetToNew

```

```

Administrator: Windows PowerShell
End - Checking if Job exists.
End - Deleting Job.
End - Clean up
=> Agents.Tests.Integration.AgentTests.Job_ShouldNotBePickedUp_WhenStatusIsNotSetToNew
Start - ARRANGE
End - ARRANGE
Start - ACT
Start - Creating Job.
End - Creating Job.
jobArtifactId= 1039423
attempt= 1
Start - Retrieving Job Status.
End - Retrieving Job Status.
jobStatus=
attempt= 2
Start - Retrieving Job Status.
End - Retrieving Job Status.
jobStatus=
attempt= 3
Start - Retrieving Job Status.
End - Retrieving Job Status.
jobStatus=
End - ACT
Start - ASSERT
End - ASSERT
Start - Deleting Job.
Start - Checking if Job exists.
End - Checking if Job exists.
End - Deleting Job.
=> Agents.Tests.Integration.AgentTests
Start - OneTimeSetUp
Start - Setup Test Variables.
End - Setup Test Variables.
Start - Setup API Endpoints.
End - Setup API Endpoints.
Start - Create Workspace.
Creating workspace.
Workspace created [WorkspaceArtifactId= 1017193]
End - Create Workspace.
Start - Install Application in Workspace.
Starting Import Application.....
Querying for Application artifact id.....
Application artifactid is 1039394
Exiting Import Application method.....
End - Install Application in Workspace.
Agent exists in the Instance. Skipped creation.
End - OneTimeSetup
Start - OneTimeTearDown
Deleting Agent.
Agent(s) Deleted. Count = 1
End - OneTimeTearDown

Run Settings
DisposeRunners: True
WorkDirectory: E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project
ImageRuntimeVersion: 4.0.30319
ImageTargetFrameworkName: .NETFramework,Version=v4.6.2
ImageRequiresX86: False
ImageRequiresDefaultAppDomainAssemblyResolver: False
NumberOfTestWorkers: 4

Test Run Summary
Overall result: Passed
Test Count: 2, Passed: 2, Failed: 0, Warnings: 0, Inconclusive: 0, Skipped: 0
Start time: 2018-09-14 09:45:03Z
End time: 2018-09-14 09:47:37Z
Duration: 154.201 seconds

Results (nunit2) saved as E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project\Artifacts\TestLogs\IntegrationTest.xml

Build Succeeded!

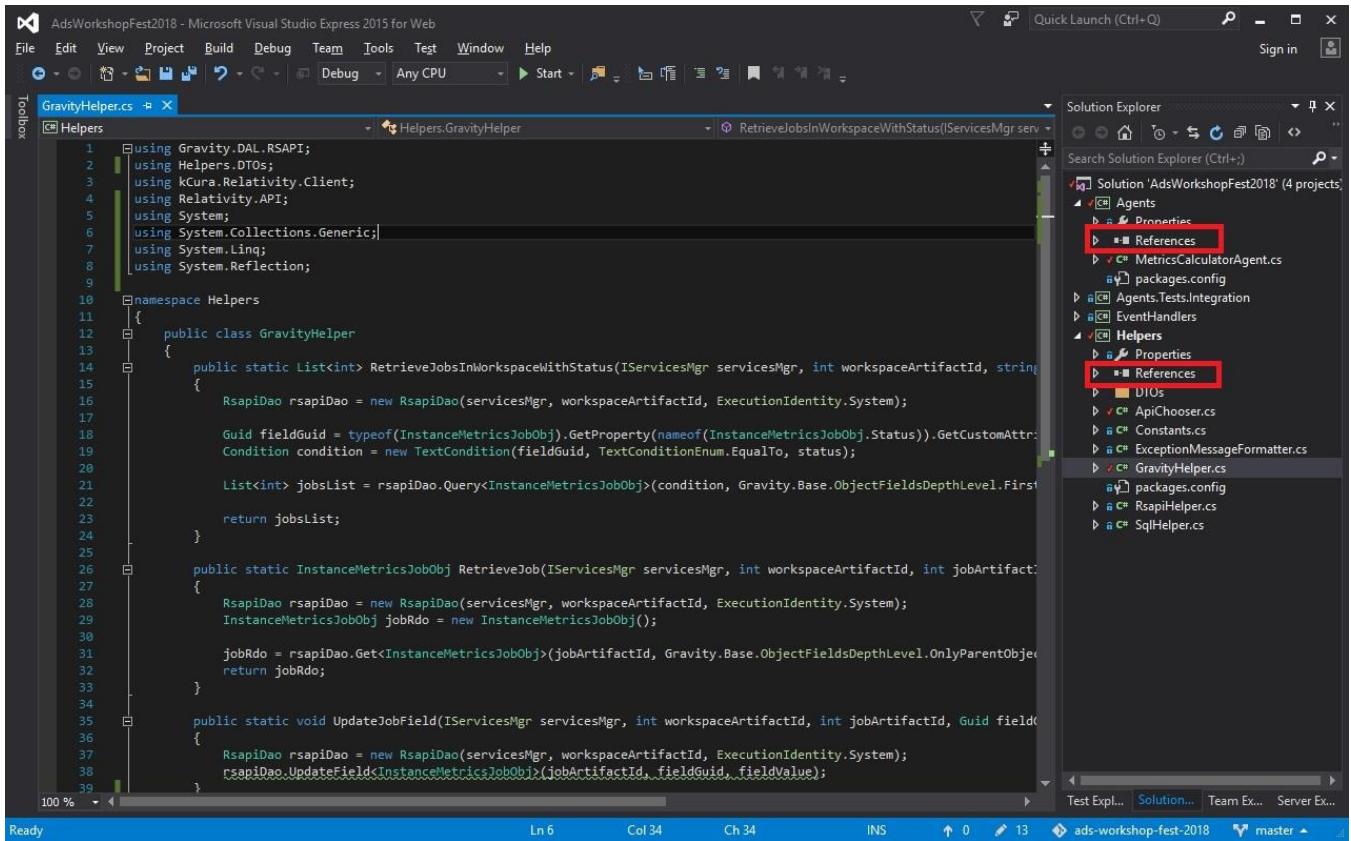
-----
Build Time Report
-----
Name          Duration
---           ---
TestInitialize 00:00:00.0352480
IntegrationTest 00:02:36.5150248
Total:         00:02:36.5632791

PS E:\Github\ads-workshop-fest-2018\Projects\Starting\4_IntegrationTests\Project>

```

7 Gravity API

- 1 In this section, we will be using the Gravity open source API instead of RSAPI to read and update RDO values. You can read more about the Gravity API at this link:
[https://github.com/relativitydev\(Gravity](https://github.com/relativitydev(Gravity)
- 2 To use the **Gravity** API, first go to Visual Studio and add Gravity.dll as new reference for **Helpers** project and **Agents** project. You can find Gravity.dll in **ads-workshop-fest-2018\DLLs\Gravity**



- 3 To use the **Gravity** API, first go to Visual Studio and create enum MetricsChoices under folder **DTOs** in **Helpers** project. This enum represents the Choices for Metrics field.
*Note: To create the enum right-click on the folder **DTOs** and select **Add -> New Item**. On the new window select **Visual C# Items -> Class**. Name the file **MetricsChoices.cs**.*

```
public enum MetricsChoices
{
    [RelativityObject("9715EB01-97F0-496A-9640-2494AD7CAA35")]
    Workspaces = 1,

    [RelativityObject("49BE6FCC-DB19-4BA3-A849-712DD2A72650")]
    Users = 2,

    [RelativityObject("E9BCE5CE-EC87-4A46-AB61-E8157DC5BA57")]
    Groups = 3
}
```

Code to add:

```
public enum MetricsChoices
{
    [RelativityObject("9715EB01-97F0-496A-9640-2494AD7CAA35")]
    Workspaces = 1,

    [RelativityObject("49BE6FCC-DB19-4BA3-A849-712DD2A72650")]
    Users = 2,

    [RelativityObject("E9BCE5CE-EC87-4A46-AB61-E8157DC5BA57")]
    Groups = 3
}
```

- 4 Then create class for the RDO with its fields as properties under folder DTOs in Helpers project.

*Note: To create the class right-click on the folder **DTOs** and select **Add -> New Item**. On the new window select **Visual C# Items -> Class**. Name the file **InstanceMetricsJobObj.cs**.*

```
[Serializable]
[RelativityObject("07FCE2E4-3318-4A00-9EF4-566FFCD7C198")]
public class InstanceMetricsJobObj : BaseDto
{
    [RelativityObjectField("7D1DFEDD-36A2-41A2-97D3-C1537DCD0598", RdoFieldType.FixedLengthText)]
    public override string Name { get; set; }

    [RelativityObjectField("065F1211-5A65-4DAC-AA26-EEED9007DCA9", RdoFieldType.LongText)]
    public string Status { get; set; }

    [RelativityObjectField("70401A4A-94CC-45BB-A6CA-808F6754F114", RdoFieldType.MultipleChoice)]
    public IList<MetricsChoices> Metrics { get; set; }

    [RelativityObjectField("8435115F-894F-43C3-978E-8E9CF42AB2DB", RdoFieldType.LongText)]
    public string WorkspacesCount { get; set; }

    [RelativityObjectField("45D186F4-A9BB-4427-A6DA-DAA4AD639024", RdoFieldType.LongText)]
    public string UsersCount { get; set; }

    [RelativityObjectField("9B56A63F-5F42-4CBD-BE53-761E8AD32CA0", RdoFieldType.LongText)]
    public string GroupsCount { get; set; }

    [RelativityObjectField("B9A22B34-0D87-4527-AD7F-B070AF4470AE", RdoFieldType.LongText)]
    public string Errors { get; set; }
}
```

Code to add:

```
using Gravity.Base;
using System;
using System.Collections.Generic;

namespace Helpers.DTOs
{
    [Serializable]
```

```

[RelativityObject("07FCE2E4-3318-4A00-9EF4-566FFCD7C198")]
public class InstanceMetricsJobObj : BaseDto
{
    [RelativityObjectField("7D1DFEDD-36A2-41A2-97D3-C1537DCD0598",
RdoFieldType.FixedLengthText)]
    public override string Name { get; set; }

    [RelativityObjectField("065F1211-5A65-4DAC-AA26-EED9007DCA9",
RdoFieldType.LongText)]
    public string Status { get; set; }

    [RelativityObjectField("70401A4A-94CC-45BB-A6CA-808F6754F114",
RdoFieldType.MultipleChoice)]
    public IList<MetricsChoices> Metrics { get; set; }

    [RelativityObjectField("8435115F-894F-43C3-978E-8E9CF42AB2DB",
RdoFieldType.LongText)]
    public string WorkspacesCount { get; set; }

    [RelativityObjectField("45D186F4-A9BB-4427-A6DA-DAA4AD639024",
RdoFieldType.LongText)]
    public string UsersCount { get; set; }

    [RelativityObjectField("9B56A63F-5F42-4CBD-BE53-761E8AD32CA0",
RdoFieldType.LongText)]
    public string GroupsCount { get; set; }

    [RelativityObjectField("B9A22B34-0D87-4527-AD7F-B070AF4470AE",
RdoFieldType.LongText)]
    public string Errors { get; set; }
}

```

- 5 To use Gravity, you have to create RsapiDao. You will see this for the methods below.

```
RsapiDao rsapiDao = new RsapiDao(servicesMgr, workspaceArtifactId, ExecutionIdentity.System);
```

- 6 When you have created the RsapiDao, you can use it for various CRUDQ operations.

```
List<int> jobsList = rsapiDao.Query<InstanceMetricsJobObj>(condition, Gravity.Base.ObjectFieldsDepthLevel.FirstLevelOnly).Select(x => x.ArtifactId).ToList();
```

- 7 If you want to retrieve the GUID for a field, you can do that by using Reflection

```
Guid fieldGuid = typeof(InstanceMetricsJobObj).GetProperty(nameof(InstanceMetricsJobObj.Status)).GetCustomAttribute<RelativityObjectFieldAttribute>().FieldGuid;
```

- 8 Open the **GravityHelper.cs** file in **Helpers** project and add the following **using** statements

```
using Gravity.DAL.RSAPI;
using Helpers.DTOs;
using kCura.Relativity.Client;
using Relativity.API;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
```

Code to add:

```
using Gravity.DAL.RSAPI;
using Helpers.DTOs;
using kCura.Relativity.Client;
using Relativity.API;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
```

- 9 Next declare method to retrieve all jobs in a workspace for specific status

```
public static List<int> RetrieveJobsInWorkspaceWithStatus(IServicesMgr servicesMgr, int workspaceArtifactId, string status)
{
    RsapiDao rsapiDao = new RsapiDao(servicesMgr, workspaceArtifactId, ExecutionIdentity.System);

    Guid fieldGuid = typeof(InstanceMetricsJobObj).GetProperty(nameof(InstanceMetricsJobObj.Status)).GetCustomAttribute<RelativityObjectFieldAttribute>().FieldGuid;
    Condition condition = new TextCondition(fieldGuid, TextConditionEnum.EqualTo, status);

    List<int> jobsList = rsapiDao.Query<InstanceMetricsJobObj>(condition, Gravity.Base.ObjectFieldsDepthLevel.FirstLevelOnly).Select(x => x.ArtifactId).ToList();
}

return jobsList;
```

Code to add:

```
public static List<int> RetrieveJobsInWorkspaceWithStatus(IServicesMgr servicesMgr, int
workspaceArtifactId, string status)
{
    RsapiDao rsapiDao = new RsapiDao(servicesMgr, workspaceArtifactId,
ExecutionIdentity.System);

    Guid fieldGuid =
typeof(InstanceMetricsJobObj).GetProperty(nameof(InstanceMetricsJobObj.Status)).GetCustomAttribute<RelativityObjectFieldAttribute>().FieldGuid;
    Condition condition = new TextCondition(fieldGuid, TextConditionEnum.EqualTo,
status);

    List<int> jobsList = rsapiDao.Query<InstanceMetricsJobObj>(condition,
Gravity.Base.ObjectFieldsDepthLevel.FirstLevelOnly).Select(x => x.ArtifactId).ToList();
```

```
        return jobsList;
    }
```

- 10 In the same **GravityHelper.cs**, declare method to retrieve one specific job

```
public static InstanceMetricsJobObj RetrieveJob(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId)
{
    RsapiDao rsapiDao = new RsapiDao(servicesMgr, workspaceArtifactId, ExecutionIdentity.System);
    InstanceMetricsJobObj jobRdo = new InstanceMetricsJobObj();

    jobRdo = rsapiDao.Get<InstanceMetricsJobObj>(jobArtifactId, Gravity.Base.ObjectFieldsDepthLevel.OnlyParentObject);
    return jobRdo;
}
```

Code to add:

```
public static InstanceMetricsJobObj RetrieveJob(IServicesMgr servicesMgr, int workspaceArtifactId,
int jobArtifactId)
{
    RsapiDao rsapiDao = new RsapiDao(servicesMgr, workspaceArtifactId,
ExecutionIdentity.System);
    InstanceMetricsJobObj jobRdo = new InstanceMetricsJobObj();

    jobRdo = rsapiDao.Get<InstanceMetricsJobObj>(jobArtifactId,
Gravity.Base.ObjectFieldsDepthLevel.OnlyParentObject);
    return jobRdo;
}
```

- 11 In the same **GravityHelper.cs**, declare method to update specific field

```
public static InstanceMetricsJobObj RetrieveJob(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId)
{
    RsapiDao rsapiDao = new RsapiDao(servicesMgr, workspaceArtifactId, ExecutionIdentity.System);
    InstanceMetricsJobObj jobRdo = new InstanceMetricsJobObj();

    jobRdo = rsapiDao.Get<InstanceMetricsJobObj>(jobArtifactId, Gravity.Base.ObjectFieldsDepthLevel.OnlyParentObject);
    return jobRdo;
}
```

Code to add:

```
public static void UpdateJobField(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId,
Guid fieldGuid, object fieldValue)
{
    RsapiDao rsapiDao = new RsapiDao(servicesMgr, workspaceArtifactId,
ExecutionIdentity.System);
    rsapiDao.UpdateField<InstanceMetricsJobObj>(jobArtifactId, fieldGuid, fieldValue);
}
```

- 12 Open the **ApiChooser.cs** file in **Helpers** project and invoke the method **GravityHelper.RetrieveJobsInWorkspaceWithStatus** (declared in step 4) in method public **List<int> RetrieveJobsInWorkspaceWithStatus(IServicesMgr servicesMgr, int workspaceArtifactId, string status)**

```
public List<int> RetrieveJobsInWorkspaceWithStatus(IServicesMgr servicesMgr, int workspaceArtifactId, string status)
{
    List<int> jobsList;

    if (_apiType.Equals(Constants.ApiType.Rsapi))
    {
        jobsList = RsapiHelper.RetrieveJobsInWorkspaceWithStatus(servicesMgr, workspaceArtifactId, status);
    }
    else if (_apiType.Equals(Constants.ApiType.Gravity))
    {
        jobsList = GravityHelper.RetrieveJobsInWorkspaceWithStatus(servicesMgr, workspaceArtifactId, status);   
    }
    else
    {
        throw new Exception(Constants.ErrorMessages.INVALID_API_TYPE_ERROR);
    }

    return jobsList;
}
```

Code to comment:

```
jobsList = new List<int>();
```

Code to add:

```
jobsList = GravityHelper.RetrieveJobsInWorkspaceWithStatus(servicesMgr, workspaceArtifactId, status);
```

- 13 Next, create a method to invoke **GravityHelper.RetrieveJob**

```
public InstanceMetricsJobObj RetrieveJobWithGravity(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId)
{
    InstanceMetricsJobObj jobRdo = null;

    jobRdo = GravityHelper.RetrieveJob(servicesMgr, workspaceArtifactId, jobArtifactId);

    return jobRdo;
}
```

Code to add:

```
public InstanceMetricsJobObj RetrieveJobWithGravity(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId)
{
    InstanceMetricsJobObj jobRdo = null;
```

```

        jobRdo = GravityHelper.RetrieveJob(servicesMgr, workspaceArtifactId, jobArtifactId);

        return jobRdo;
    }

```

- 14 Now invoke **GravityHelper.UpdateJobField** in method **public void UpdateJobField(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId, Guid fieldGuid, object fieldValue)**

```

public void UpdateJobField(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId, Guid fieldGuid, object fieldValue)
{
    if (_apiType.Equals(Constants.ApiType.Rsapi))
    {
        RsapiHelper.UpdateJobField(servicesMgr, workspaceArtifactId, jobArtifactId, fieldGuid, fieldValue);
    }
    else if (_apiType.Equals(Constants.ApiType.Gravity))
    {
        GravityHelper.UpdateJobField(servicesMgr, workspaceArtifactId, jobArtifactId, fieldGuid, fieldValue);
    }
    else
    {
        throw new Exception(Constants.ErrorMessages.INVALID_API_TYPE_ERROR);
    }
}

```

Code to add:

GravityHelper.UpdateJobField(servicesMgr, workspaceArtifactId, jobArtifactId, fieldGuid, fieldValue);

- 15 Now you are ready to use the ApiChooser methods in the Agents project. Go to Agents project and open **MetricsCalculatorAgent.cs**. Create overload method **ProcessAllMetrics(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId, InstanceMetricsJobObj jobRdo)**.

```

private void ProcessAllMetrics(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId, InstanceMetricsJobObj jobRdo)
{
    try
    {
        foreach (MetricsChoices metric in jobRdo.Metrics)
        {
            Guid metricGuid = metric.GetRelativityObjectAttributeValue();

            ProcessSingleMetric(servicesMgr, workspaceArtifactId, jobArtifactId, metricGuid);
        }
    }
    catch (Exception ex)
    {
        throw new Exception(Constants.ErrorMessages.PROCESS_ALL_JOB_METRICS_ERROR, ex);
    }
}

```

Code to add:

```

private void ProcessAllMetrics(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId,
InstanceMetricsJobObj jobRdo)
{
    try
    {
        foreach (MetricsChoices metric in jobRdo.Metrics)

```

```

        {
            Guid metricGuid = metric.GetRelativityObjectAttributeValue();

            ProcessSingleMetric(servicesMgr, workspaceArtifactId, jobArtifactId,
                metricGuid);
        }
    }
    catch (Exception ex)
    {
        throw new
        Exception(Constants.ErrorMessages.PROCESS_ALL_JOB_METRICS_ERROR, ex);
    }
}

```

16 Use RetrieveJobWithGravity to retrieve and process the job in **private void ProcessJob(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId)**

```

private void ProcessJob(IServicesMgr servicesMgr, int workspaceArtifactId, int jobArtifactId)
{
    try
    {
        //Update job status to In Progress
        _apiChooser.UpdateJobField(servicesMgr, workspaceArtifactId, jobArtifactId, Constants.Guids.Fields.InstanceMetricsJob.Status_LongText, Constants.JobStatus.IN_PROGRESS);

        //Update job metrics
        if(selectedApiType == Helpers.Constants.ApiType.Rsapi)
        {
            RDO jobRdo = _apiChooser.RetrieveJob(servicesMgr, workspaceArtifactId, jobArtifactId);
            RaiseMessage("Calculating metrics for the job", 10);
            ProcessAllMetrics(servicesMgr, workspaceArtifactId, jobArtifactId, jobRdo);
            RaiseMessage("Calculated metrics for the job", 10);
        }
        else if(selectedApiType == Helpers.Constants.ApiType.Gravity)
        {
            InstanceMetricsJobObj jobRdo = _apiChooser.RetrieveJobWithGravity(servicesMgr, workspaceArtifactId, jobArtifactId);
            RaiseMessage("Calculating metrics for the job", 10);
            ProcessAllMetrics(servicesMgr, workspaceArtifactId, jobArtifactId, jobRdo);
            RaiseMessage("Calculated metrics for the job", 10);
        }

        //Update job status to Completed
        _apiChooser.UpdateJobField(servicesMgr, workspaceArtifactId, jobArtifactId, Constants.Guids.Fields.InstanceMetricsJob.Status_LongText, Constants.JobStatus.COMPLETED);
    }
    catch (Exception ex)
    {
        //Update job status to Error
        string errorMessage = ErrorMessageFormatter.GetInnerMostExceptionMessage(ex);
        _apiChooser.UpdateJobField(servicesMgr, workspaceArtifactId, jobArtifactId, Constants.Guids.Fields.InstanceMetricsJob.Status_LongText, Constants.JobStatus.COMPLETED);
        _apiChooser.UpdateJobField(servicesMgr, workspaceArtifactId, jobArtifactId, Constants.Guids.Fields.InstanceMetricsJob.Errors_LongText, errorMessage);
    }
}

```

Code to add:

InstanceMetricsJobObj jobRdo = _apiChooser.RetrieveJobWithGravity(servicesMgr, workspaceArtifactId, jobArtifactId);

RaiseMessage("Calculating metrics for the job", 10);

ProcessAllMetrics(servicesMgr, workspaceArtifactId, jobArtifactId, jobRdo);

RaiseMessage("Calculated metrics for the job", 10);

- 17 To switch from RSAPI to Gravity change the field **private static readonly Constants.ApiType selectedApiType** in **MetricsCalculatorAgent.cs**

```
[kCura.Agent.CustomAttributes.Name(Constants.Names.AGENT_INSTANCE_METRICS_CALCULATOR)]
[System.Runtime.InteropServices.Guid("20530D16-D825-4FA5-9A7E-6760579EB07B")]
public class MetricsCalculatorAgent : kCura.Agent.AgentBase
{
    private IAPILog _logger;

    public override string Name => Constants.Names.AGENT_INSTANCE_METRICS_CALCULATOR;
    private static readonly Constants.ApiType selectedApiType = Constants.ApiType.Gravity;
    private readonly Apichoose _apichoose = new Apichoose(selectedApiType);

    public override void Execute()
    {
        RaiseMessage("Enter Agent", 10);

        try
        {
            _logger = Helper.GetLoggerFactory().GetLogger();
            IServicesMgr servicesMgr = Helper.GetServicesManager();
            IDBContext eddsDbContext = Helper.GetDbContext(-1);
```