



# How to Create Solutions for Relativity & ADS Solutions Workshop

September 12, 2018 - Version 1.0

# Contents

## 1 Course Overview3

1.1 What You'll Learn3

1.2 Required Development Software3

## 2 Getting Started4

2.1 Visual Studio Project4

2.2 DevVM Setup6

## 3 Starting Development11

3.1 Application Setup11

3.2 New Event Handler14

3.3 Publish17

## 4 Logging21

4.1 Adding Logging21

4.2 Viewing Logs22

## 5 Customize New Button Override & Override Edit Link URL Object Rules25

5.1 Custom Page25

5.2 Visual Studio Post-Build Event & Publish To Relativity Console35

5.3 Object Rules42

5.4 ObjectManager API49

5.5 Gravity API50

## 6 Unit Tests57

# 1 Course Overview

ADS is the standard way to build applications on the Relativity Platform. As a developer, you want a good process for building apps so you can focus on delivering business value quickly. In this session, we'll review common patterns for setting up an effective ADS development solution. Topics will include: setting up a Visual Studio build workflow; versioning; which assemblies to include in your application; source code management; Relativity developer tools; and more. We'll also highlight recent improvements to ADS that'll make your life easier.

## 1.1 What You'll Learn

- Faster Development
- Publish to Relativity
- Logging
- How to Override New Button Link for a RDO
- How to Override Edit Link for a RDO
- Publish to Relativity Console
- Visual Studio Post-Publish Event
- Object Manager API
- Gravity API
- Unit Tests
- Versioning

## 1.2 Required Development Software

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

- Relativity 9.5
- Visual Studio 2017
- Relativity Visual Studio Templates
- .NET Framework 4.6.2
- SQL Server 2012

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

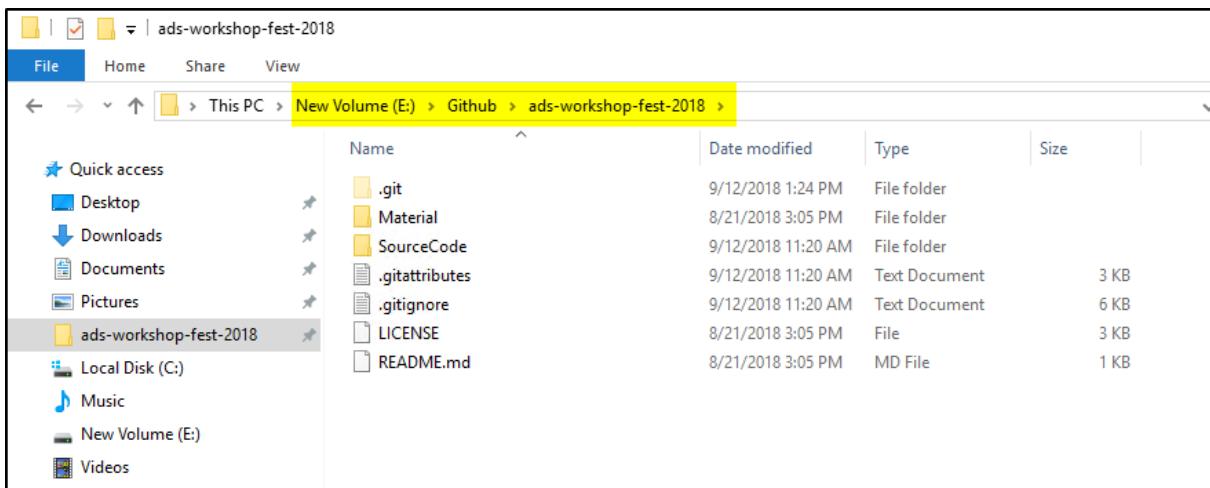
## 2 Getting Started

### 2.1 Visual Studio Project

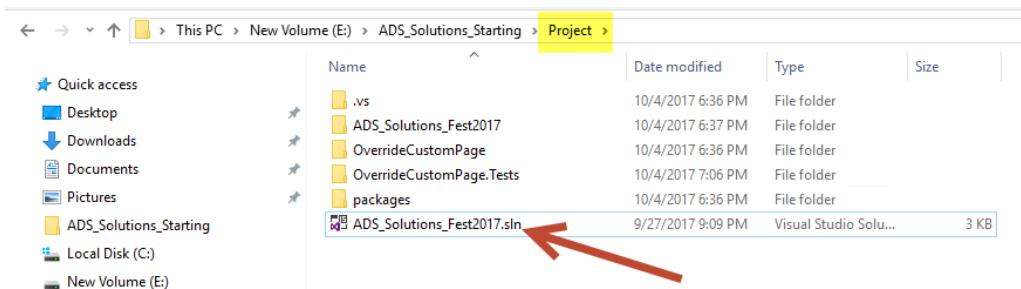
1. Make sure you have a folder shortcut on Desktop with name **ads-workshop-fest-2018**.



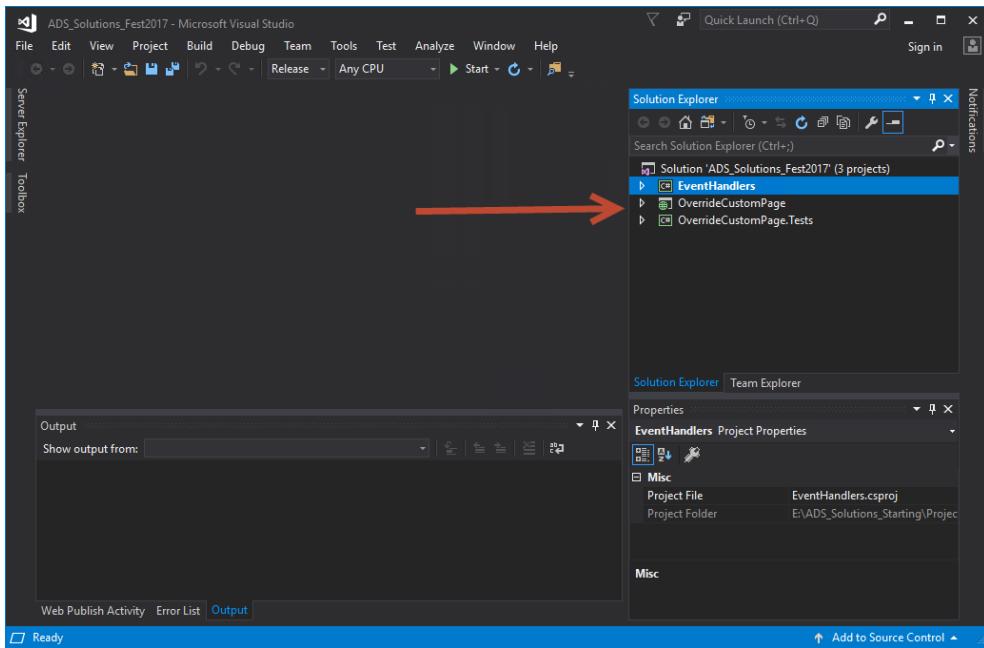
2. Double click on the **ads-workshop-fest-2018** folder to open it. Make sure you see the files as shown in the below screenshot.



3. Go to the **Project** folder and double click the **ADS\_Solutions\_Fest2017.sln** file to open the Visual Studio solution.



4. **Visual Studio 2017** will be opened and it contains three projects listed in the **Solution Explorer** as shown in the below screenshot.

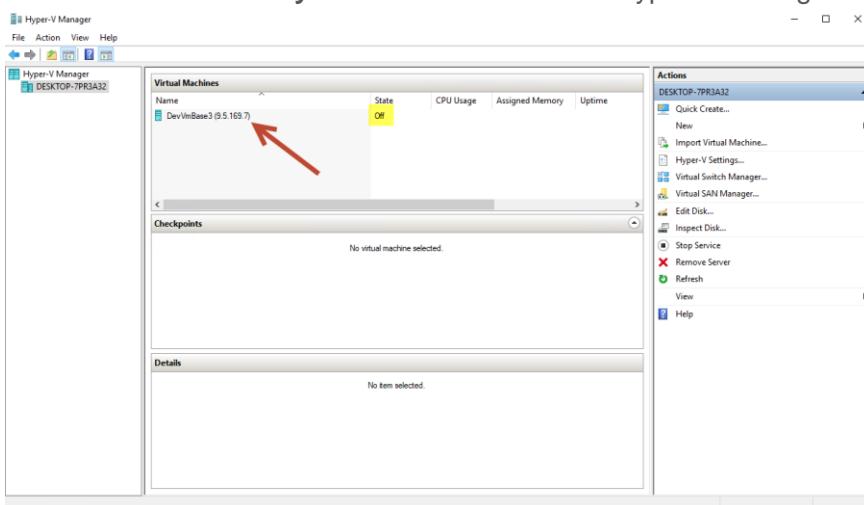


## 2.2 DevVM Setup

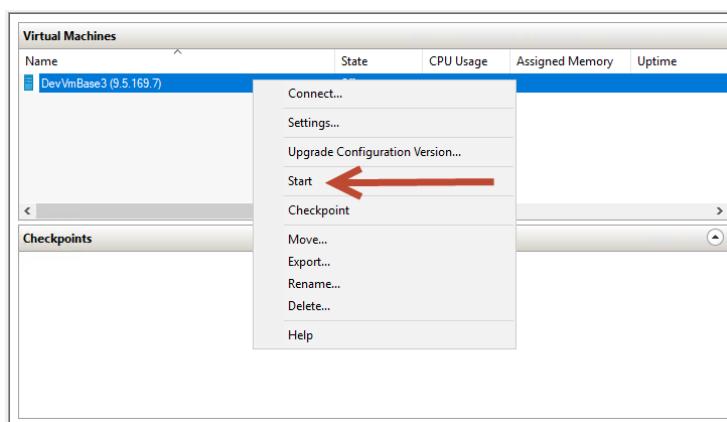
1. Open the **Hyper-V Manager** application by clicking the blue icon in the taskbar.



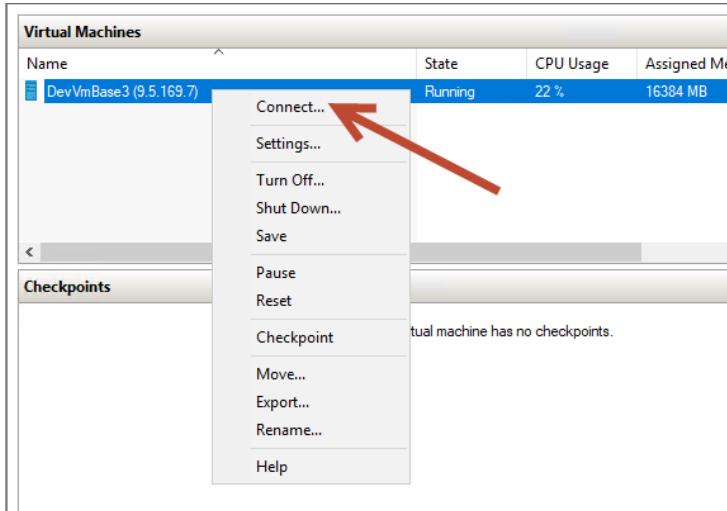
2. You will see a **Relativity 9.5 DevVM** listed in the Hyper-V Manager Virtual Machine section.



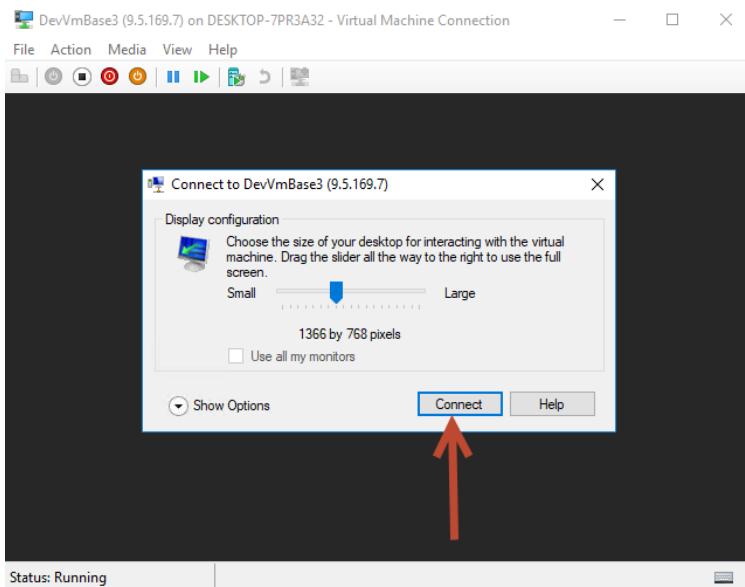
3. Right click on the **DevVM** and select the **Start** option.



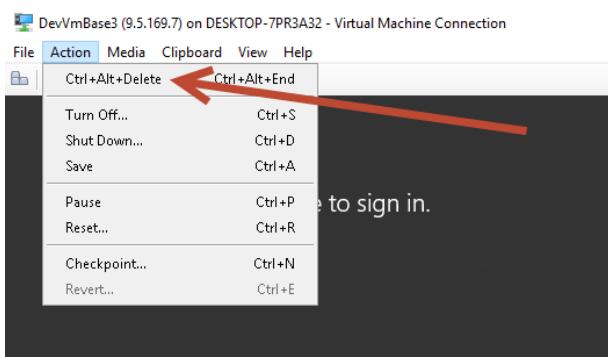
4. Right click on the **DevVM** and select the **Connect** option.



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. Go to the **ADS\_Solutions\_Starting** folder shortcut on Desktop and open the **Credentials and Paths.txt** file.

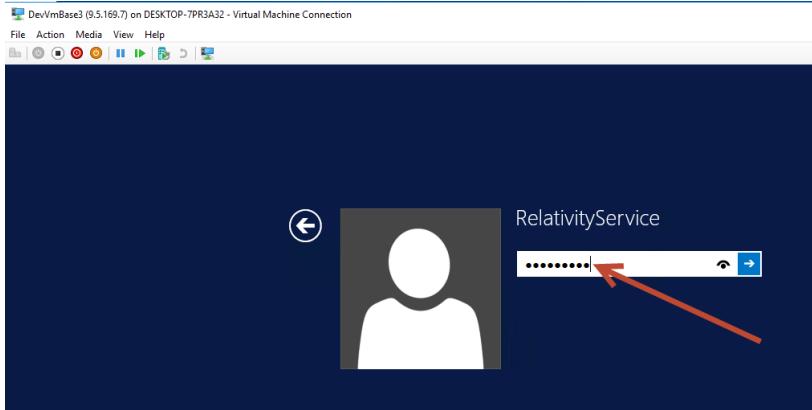
Volume (E:) > ADS_Solutions_Starting >				
Name	Date modified	Type	Size	
9.5.169.7 SDK 64bit	10/4/2017 6:45 PM	File folder		
Project	10/4/2017 6:38 PM	File folder		
<b>Credentials and Paths.txt</b>	10/4/2017 7:02 PM	Text Document	1 KB	
PublishToRelativityConsole.exe	9/28/2017 10:17 PM	Application	13 KB	
Visual Studio Post Publish Build Comma...	10/1/2017 7:25 PM	Text Document	1 KB	

8. **Credentials and Paths** file has the credentials for the DevVM, Relativity and SQL.

```
Credentials and Paths.txt - Notepad
File Edit Format View Help
DevVM:
Windows login: RelativityService
Windows password: Test1234!

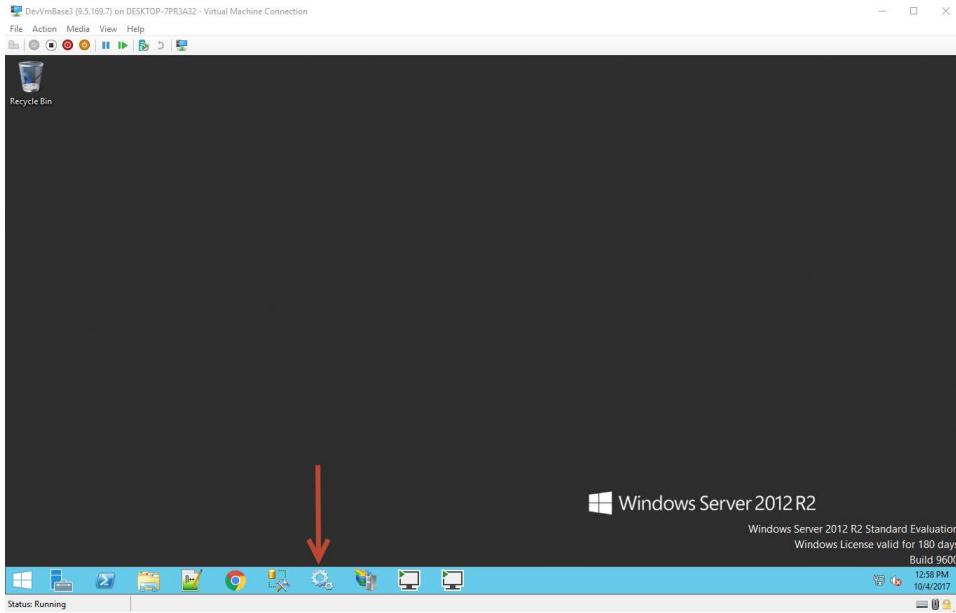
Relativity:
Admin login: relativity.admin@kcura.com
Admin password: Test1234!
```

9. Use the DevVM credentials to login to DevVM



10. 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.

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



12. 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,

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

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...
<b>kCura Service Host Manager</b>	<b>Hosts Relati...</b>	<b>Automatic (D...</b>	<b>Automatic (D...</b>	<b>\Relativity...</b>
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				

13. Open **Chrome** on your Workshop machine (Not the DevVM) and go to <http://devvmbase3/Relativity>. Use the Relativity admin credentials from the **Credentials and Paths.txt** file to login. Verify that you can login to Relativity.

#	Case Artifact ID	Name	Client Name	Matter Name
1	1018783	Workspace1	Relativity Template	Relativity Template
2	1015024	kCura Starter Template	Relativity Template	Relativity Template
3	1014823	New Case Template	Relativity Template	Relativity Template

# 3 Starting Development

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 **Chrome** on your Workshop machine (Not the DevVM) and go to <http://devvmbase3/Relativity>. Use the Relativity admin credentials from the Credentials and Paths.txt file to login.
2. Enter **Workspace 1** and navigate to the **Relativity Applications** Tab.

#	Edit	Name
1	<input type="checkbox"/>	OCR
2	<input type="checkbox"/>	Search Terms Report
3	<input type="checkbox"/>	Transform Set
4	<input type="checkbox"/>	Imaging
5	<input type="checkbox"/>	Analytics Core
6	<input type="checkbox"/>	Lists
7	<input type="checkbox"/>	Document Viewer
8	<input type="checkbox"/>	Production
9	<input type="checkbox"/>	Relativity List Page
10	<input type="checkbox"/>	Environment Statistics
11	<input type="checkbox"/>	Choice Editor
12	<input type="checkbox"/>	Data Grid Core
13	<input type="checkbox"/>	Field Catalog
14	<input type="checkbox"/>	Relativity Color Map
15	<input type="checkbox"/>	Processing
16	<input type="checkbox"/>	ADS Solutions Fest 2017

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

**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.XX

Revision number assigned automatically:

User-friendly URL:

Buttons: Save, Save and New, Save and Back, Cancel

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

**Application Information**

Name: MyNewApp

Version:

User-friendly URL:

**History**

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

System Created On: 10/6/2017 8:39 AM  
System Last Modified On: 10/6/2017 8:39 AM

**Object Type (Relativity Applications)**

**Field (Relativity Applications)**

Buttons: New, Link, Unlock, Edit, Delete, Back, Edit Permissions, View Audit

Right Sidebar:

- RELATIVITY APPLICATION Manage Application
  - Upgrade Application
  - Export Application
  - Export Application Schema
  - Lock Application
  - Push to Library
- Application Information
  - Show Application Breakdown
  - Show Component GUIDs
  - Show Errors
- Refresh Page

Workspace1 - Relativity

devvmbase3/Relativity/Case/ObjectType/Edit.aspx?AppID=1018783&AssociatedArtifactID=1043056&ConnectorFieldArtifactID=1036636&ParentArtifactID=1003663

Apps Relativity

Developer Mode Is Active

9 Workspace1

ADS Solutions Fest 2017 Custodians Documents Review Batches Reporting Case Admin Job Admin Workspace Admin Indexing & Analytics Persistent List

Workspace Details Search Indexes Object Type Fields Choices Choices List Layouts Views Tabs Relativity Applications Custom Pages History User Status

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  Clear

Save Save and New Save and Back Cancel

**Object Type Information**

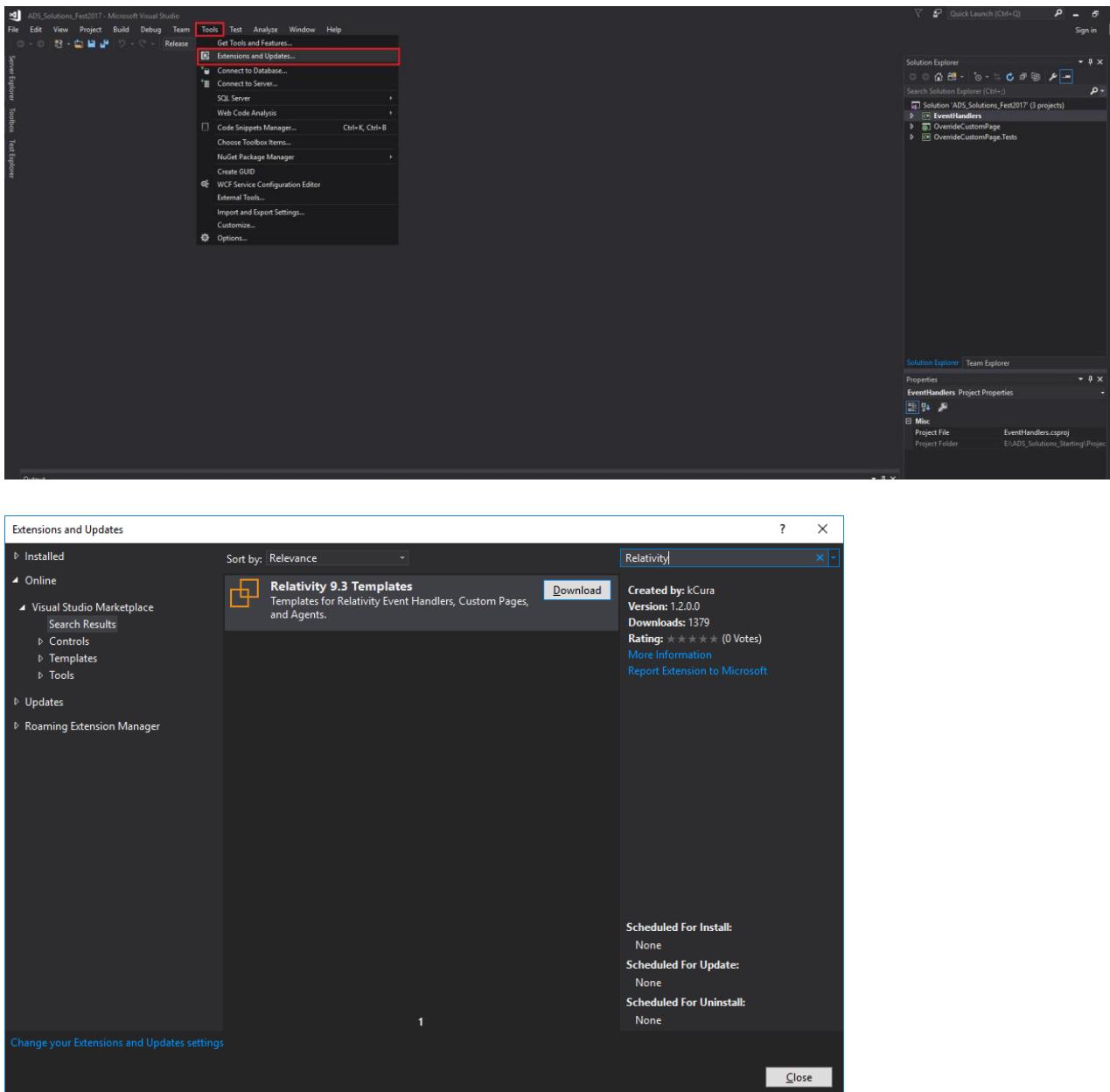
This screenshot shows the 'Object Type Information' page in Relativity's admin interface. The main title is 'Object Type Information' and the sub-section is 'Workspace1 - Relativity'. The URL in the browser is 'devvmbase3/Relativity/Case/ObjectType/Edit.aspx?AppID=1018783&AssociatedArtifactID=1043056&ConnectorFieldArtifactID=1036636&ParentArtifactID=1003663'. The top navigation bar includes links for ADS Solutions Fest 2017, Custodians, Documents, Review Batches, Reporting, Case Admin, Job Admin, Workspace Admin, Indexing & Analytics, and Persistent List. Below the navigation is a secondary menu with links for Workspace Details, Search Indexes, Object Type (which is selected), Fields, Choices, Choices List, Layouts, Views, Tabs, Relativity Applications, Custom Pages, History, and User Status. At the bottom of the page are buttons for Save, Save and New, Save and Back, and Cancel.

The main content area contains several configuration sections:

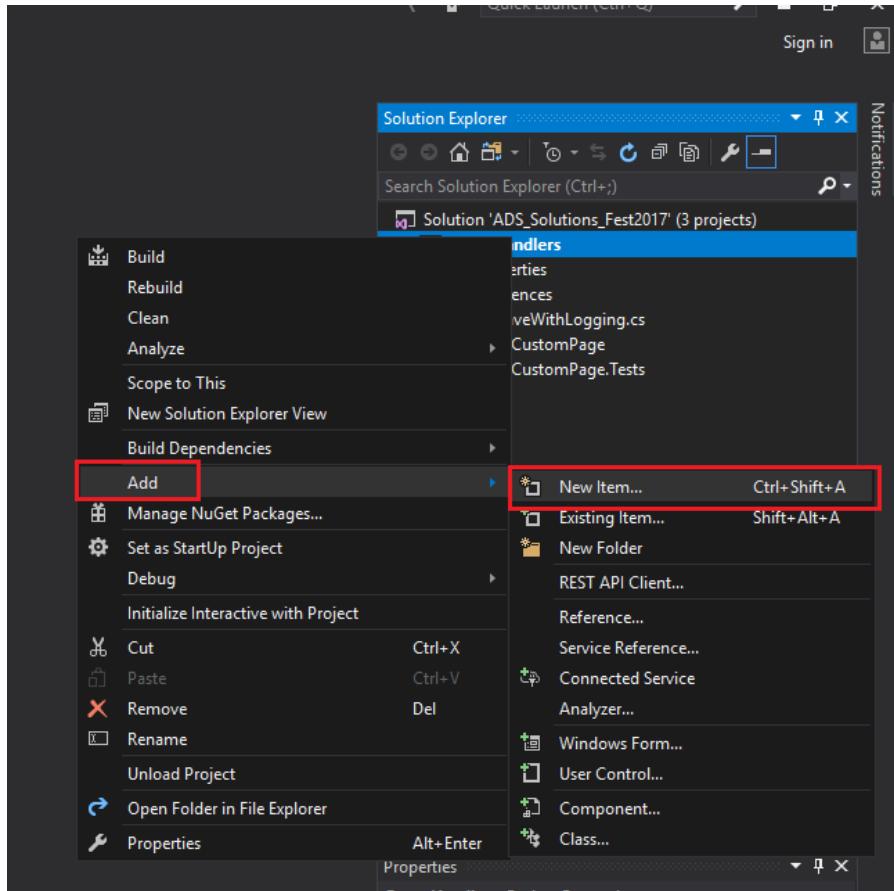
- Name:** MyNewObject
- Parent Object Type:** Workspace
- Dynamic:** Yes
- Enable Snapshot Auditing On**
  - Delete:** Yes
  - Pivot:** Enabled
  - Sampling:** Enabled
  - Lists:** Disabled
- Copy Instances On Workspace**
  - Creation:** No
- Copy Instances On Parent**
  - Copy:** No
- Relativity Applications:** MyNewApp  Clear

## 3.2 New Event Handler

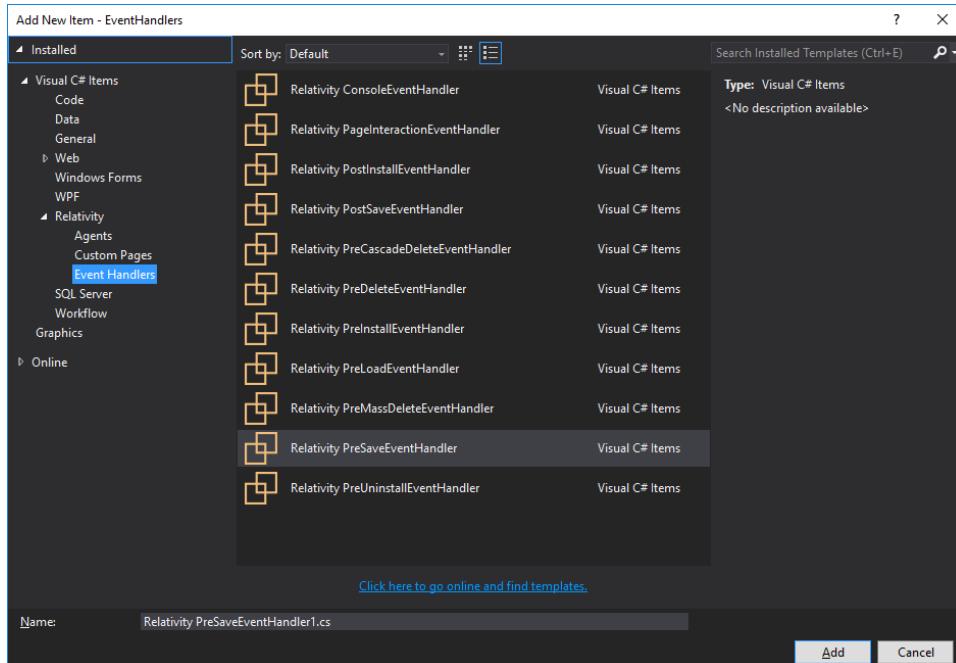
1. Open the Visual Studio solution **ADS\_Solutions\_Fest2017.sln** in the **ADS\_Solutions\_Starting** folder.
2. Verify the **Relativity Templates** are installed. You may need to **restart** visual studio.
3. Tools — Extensions and Updates — Search "Relativity"



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



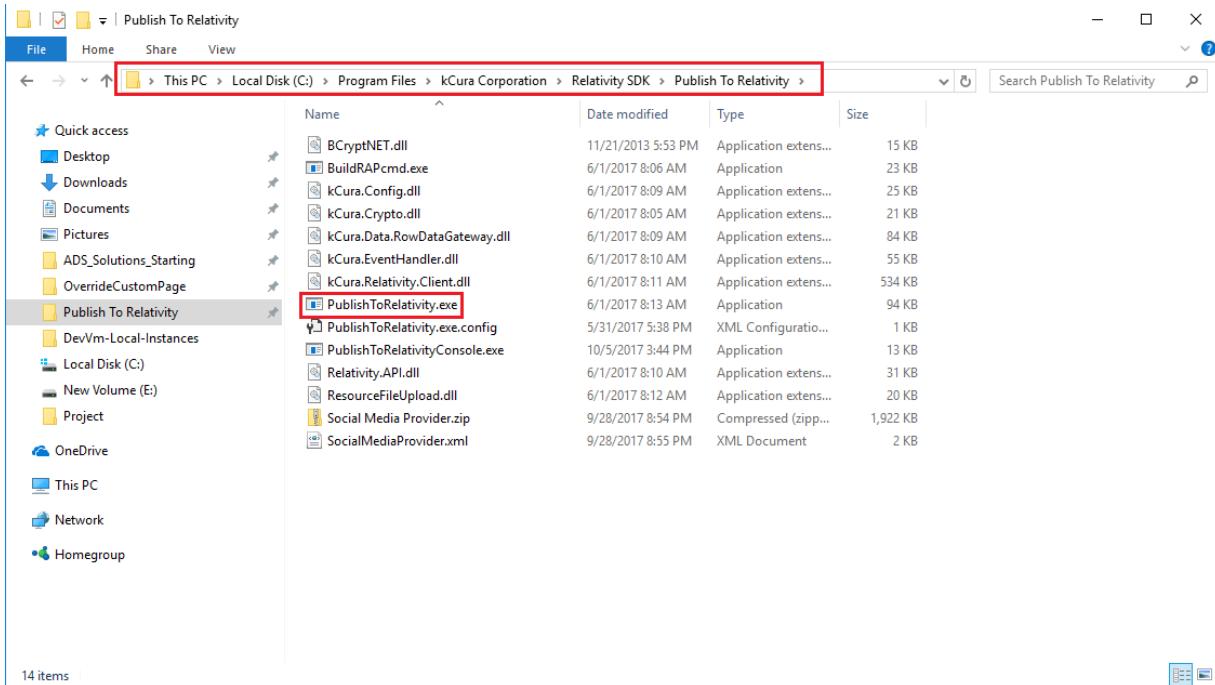
5. Under the Relativity Section, select the **pre-save event handler template**.



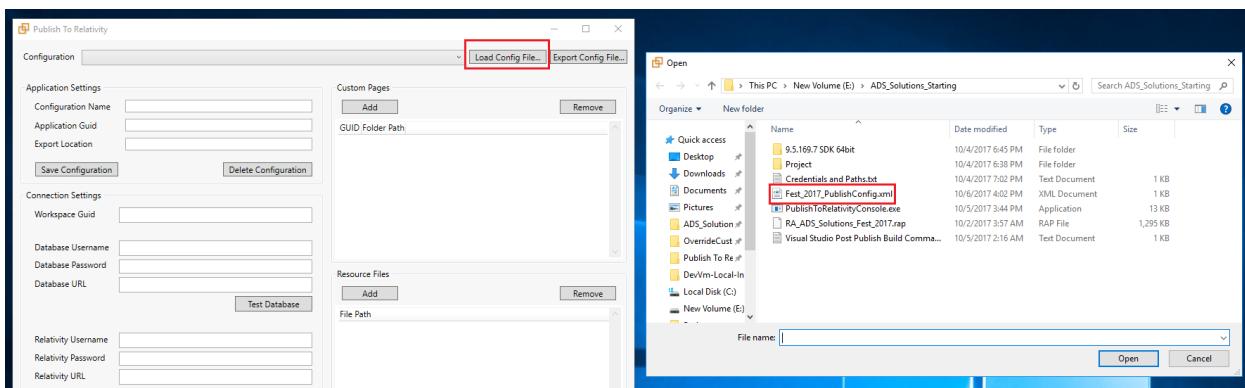
6. Click **Add**.
7. Update the event handler response.
8. Success – false
9. Message – Message of your choice

## 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 **Fest\_2017\_PublishConfig.xml** in the **ADS\_Solutions\_Startng** folder.



5. Replace the **Application GUID** with your applications GUID.

- To look up your applications GUID, navigate back to Relativity Application you created in the workspace. Then select **Show Component GUIDs**. **Developer Mode** must be active.

The screenshot shows the Relativity Application Components screen. A modal window titled "Application Components" is open, displaying a table of artifacts. One row in the table is highlighted with a red border, showing the artifact ID 1043056, artifact type "Relativity Application", artifact name "MyNewApp", version "(All)", and GUID "CA81E1FE-1061-4D29-B0C8-D0396FFA5348". The right sidebar has a section titled "RELATIVITY APPLICATION" with a button "Show Component GUIDs" highlighted with a red box.

Artifact ID	Artifact Type	Artifact Name	Version	GUIDs
1043056	Relativity Application	MyNewApp	(All)	CA81E1FE-1061-4D29-B0C8-D0396FFA5348
1043065	Object Type	MyNewObject		25211519-C02F-48A1-86B9-66059013480
1043062	Field	MyNewObject - Name		1E0F9EC-C-0E5-42F5-B3E0-D490E82864E
1043063	View	MyNewObject - All MyNewObjects		4C9781C3-0E5-42F5-B3E0-D490E82864E
1043064	Layout	MyNewObject - MyNewObject Layout		D490E82864E-7618-4BD4-8E1B-693C7252E
1043066	Tab	MyNewObject - MyNewObject		E07A7099-1616-4115-8F33-4BCAA392E39F

- In the **Resource File** Section click Add.
- Navigate to **E:\ADS\_Solutions\_Starting\Project\ADS\_Solutions\_Fest2017\bin\Release** and select **ADS\_Solutions\_Fest2017.dll**
- Click **Publish**.
- Open Relativity and navigate to your application created in section 3.1.
- Unlock** the application.

The screenshot shows the Relativity Application screen. A modal window titled "Application Information" is open, showing basic application details like name, version, and URL. The right sidebar has a section titled "RELATIVITY APPLICATION" with a button "Unlock Application" highlighted with a red box.

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

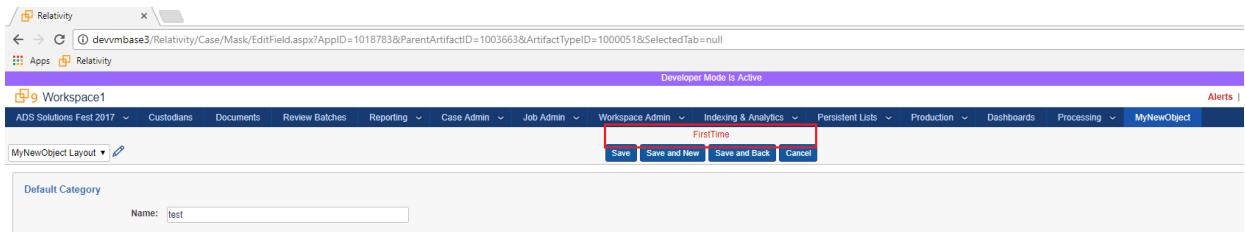
	Name	System	Dynamic	Created On	Created By
<input type="checkbox"/>	(All) <input type="button" value="Edit"/> MyNewObject	(All)	(All)	(All)	(All)

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

DLL	Class Name	Company Name	Description
(All)	(All)	(All)	(All)

14. Find the newly created Event Handler associated to your application. Filter by Application Name.

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



## 4 Logging

Relativity logging enables developers to gather runtime diagnostic information of their application. It can give detailed insight into the system to help troubleshooting application problems. This section covers adding a logging statement to a Relativity extensibility point.

### 4.1 Adding Logging

1. Open the visual studio solution ADS\_Solutions\_Fest2017 and open the EventHandlers project.
2. Create a new pre-save event handler using a Visual Studio Template, or the event handler created in section 3.
3. Update the Execute function to do the following (see screenshot for sample):
4. Throws a new exception with a message.
5. Handles the thrown exception with a try/catch.
6. Add a log statement during exception handling. Log the exception and a helpful message.

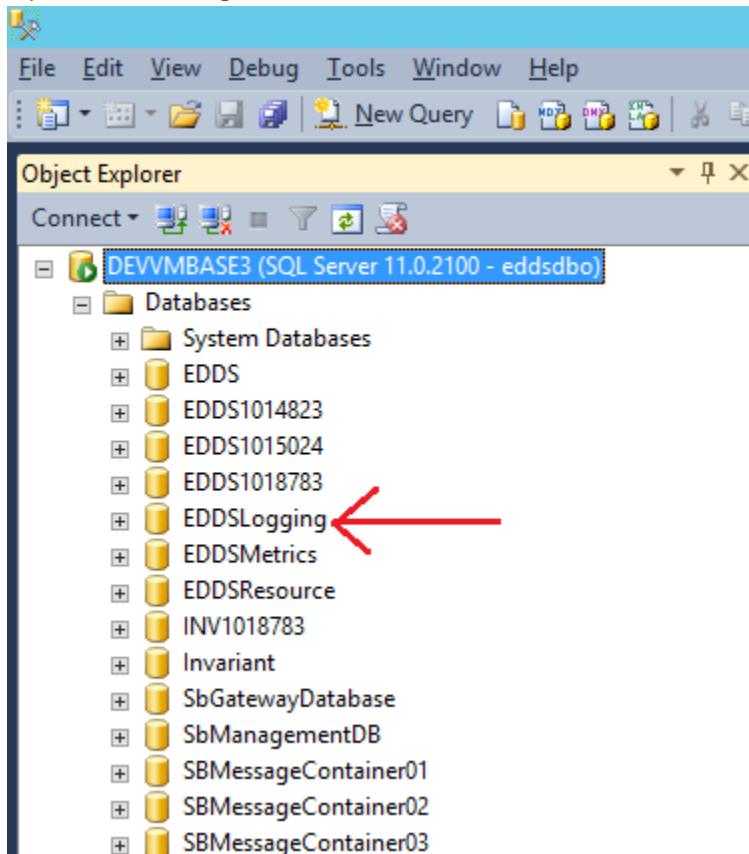
```
public override kCura.EventHandler.Response Execute()
{
    Response retVal = new kCura.EventHandler.Response();
    try
    {
        Helper.GetLoggerFactory().GetLogger().LogDebug("About to try something!");
        throw new System.Exception("Boom");
    }
    catch (System.Exception ex)
    {
        Helper.GetLoggerFactory().GetLogger()..LogError("That definitely did not work", ex);
        retVal.Success = false;
        retVal.Message = ex.ToString();
    }

    return retVal;
}
```

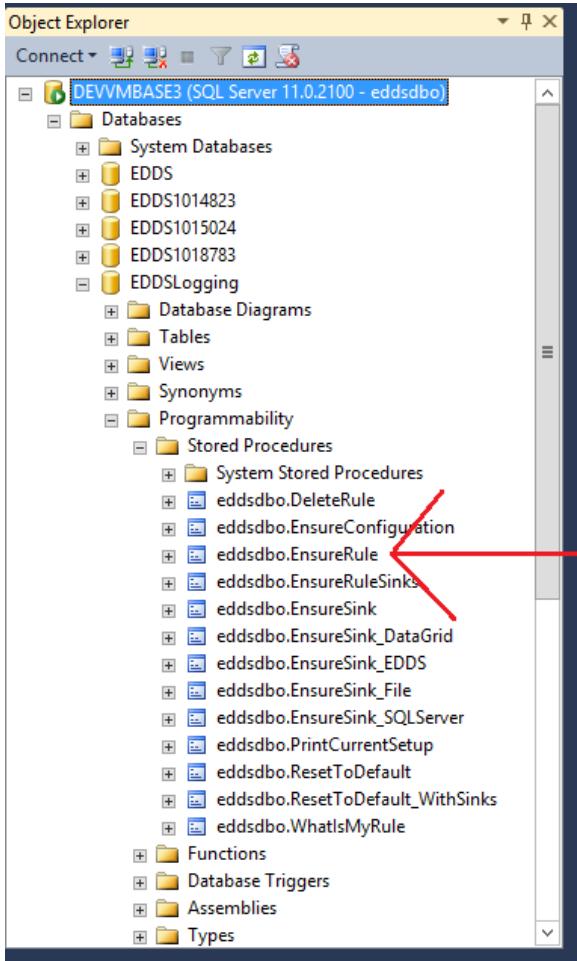
7. Build the project.
8. Click Publish in the Publish to Relativity Tool.

## 4.2 Viewing Logs

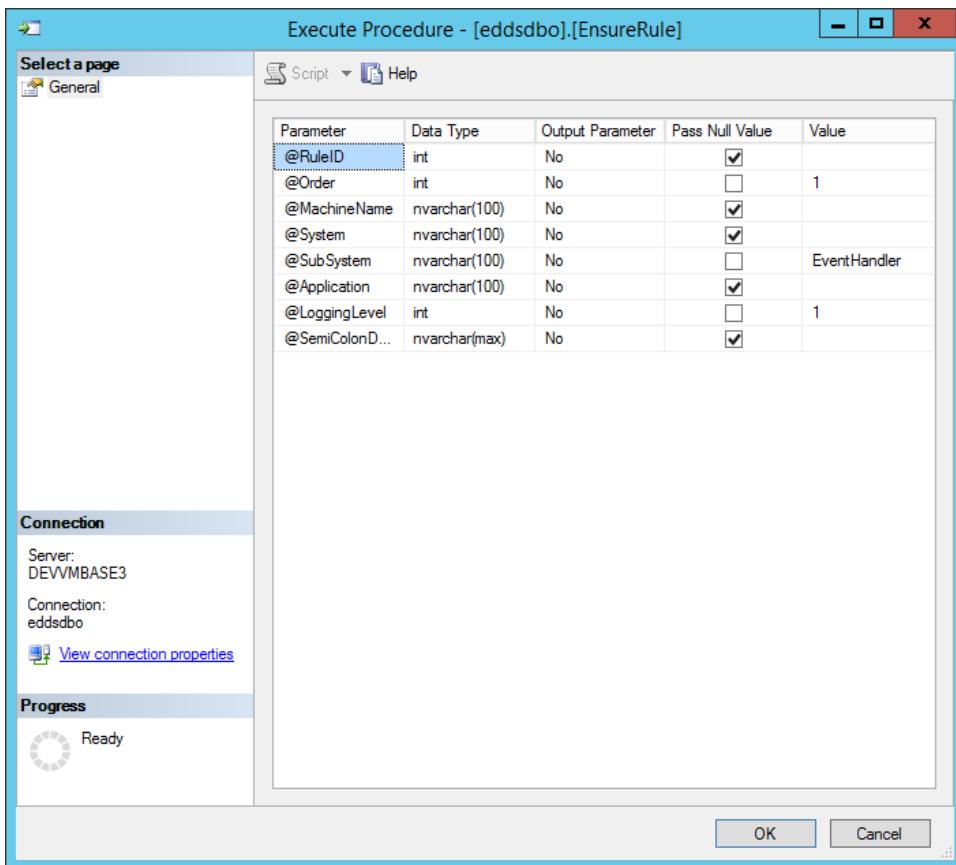
1. Log into the Relativity 9.5 DevVM.
2. Open SQL Management Studio.



3. Right click the EnsureRule stored procedure and execute.



4. Enter the following values. This sets a debug log level for all event handlers.
  - a. Order – 1
  - b. SubSystem – EventHandler
  - c. LoggingLevel – 1



5. Click OK.
6. Execute your event handler with logging.
7. Run the following command:

```
SELECT TOP 10 * FROM [EDDSLogging].[eddsdbo].[RelativityLogs] WHERE
Properties.value('(/properties/property[@key="SubSystem"])[1]', 'varchar(max)') =
'EventHandler'
```

8. If there are a lot of logs in the system, this command can take some time to execute. It can be faster to query against Message Template if that is the case.
9. Verify the logs from your event handler appear.

	ID	Message	Level
1	4	About to try something!	Debug
2	36	That definitely did not work	Error

# 5 Customize New Button Override & Override Edit Link URL Object Rules

## 5.1 Custom page

1. In this section we will be working with a Relativity application (**ADS Solutions Fest 2017**) which consists of a custom RDO named **Override RDO**. Our goal for this session is to override the **New Button Link** and **Edit Link** for the custom RDO with a custom page implementation.
2. New Button Link

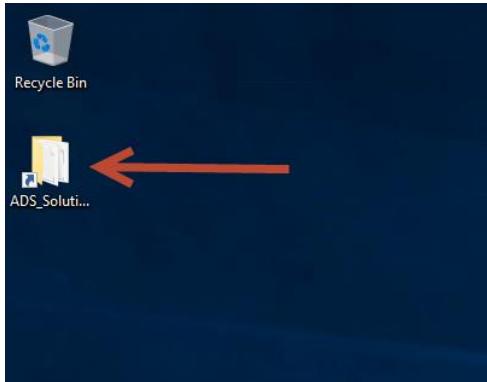
The screenshot shows a Relativity workspace titled 'Workspace1'. The top navigation bar includes 'ADS Solutions Fest 2017', 'Custodians', 'Documents', 'Review Batches', 'Reporting', and 'Case Admin'. A blue header bar indicates 'Override RDO'. Below it is a search bar with 'All Override RDOs' and a 'New Override RDO' button, which is highlighted with a red arrow. The main content area displays a table with columns: '#', 'Artifact ID', 'Name', 'Phone', and 'Email'. There are filter buttons for each column and a 'Filter' input field at the bottom of each column.

3. Edit Link

The screenshot shows the same Relativity workspace and interface as the previous screenshot. The 'Edit' button in the table row for the artifact with ID 1043055 is highlighted with a red arrow. The table data is as follows:

#	Artifact ID	Name	Phone	Email
1	1043055	test	1234567890	test@test.com

4. Go to Desktop and double click on the **ADS\_Solutions\_Starting** folder to open it.

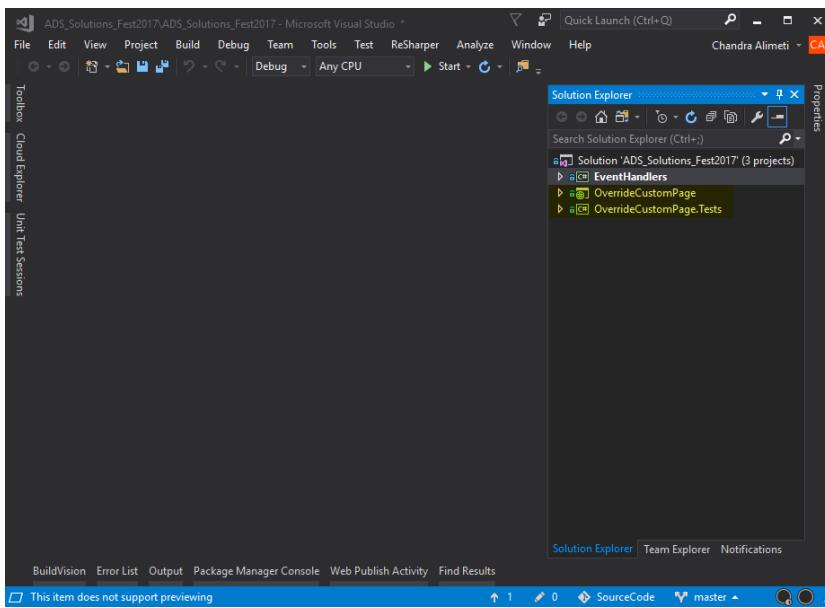


This PC > New Volume (E:) > ADS_Solutions_Starting >				
	Name	Date modified	Type	Size
Quick access	9.5.169.7 SDK 64bit	10/4/2017 6:45 PM	File folder	
Desktop	Project	10/4/2017 6:38 PM	File folder	
Downloads	Credentials and Paths.txt	10/4/2017 7:02 PM	Text Document	1 KB
Documents	PublishToRelativityConsole.exe	9/28/2017 10:17 PM	Application	13 KB
Pictures	Visual Studio Post Publish Build Comma...	10/1/2017 7:25 PM	Text Document	1 KB
ADS_Solutions_Starting				
Local Disk (C:)				
New Volume (E:)				

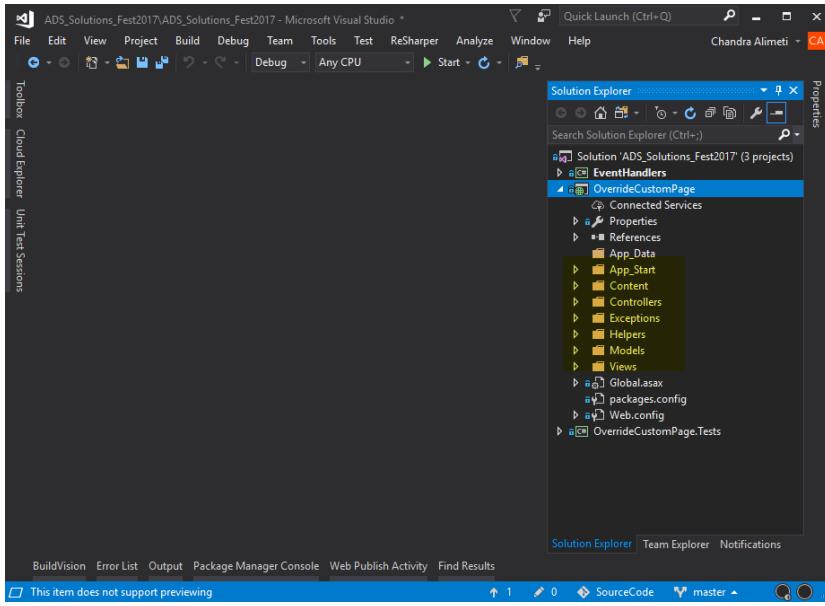
5. Go to the **Project** folder and double click the **ADS\_Solutions\_Fest2017.sln** file to open the Visual Studio solution.

This PC > New Volume (E:) > ADS_Solutions_Starting > Project >				
	Name	Date modified	Type	Size
Quick access	.vs	10/4/2017 6:36 PM	File folder	
Desktop	ADS_Solutions_Fest2017	10/4/2017 6:37 PM	File folder	
Downloads	OverrideCustomPage	10/4/2017 6:36 PM	File folder	
Documents	OverrideCustomPage.Tests	10/4/2017 7:06 PM	File folder	
Pictures	packages	10/4/2017 6:36 PM	File folder	
ADS_Solutions_Starting	ADS_Solutions_Fest2017.sln	9/27/2017 9:09 PM	Visual Studio Solu...	3 KB
Local Disk (C:)				
New Volume (E:)				

6. Visual Studio 2017 will be opened and shows you the list of the projects contained in the solution. Make sure you see the following two projects.
- o **OverrideCustomPage**
  - o **OverrideCustomPage.Tests**



7. **OverrideCustomPage** is an ASP.NET MVC 5 project. Double click the project to expand it.



8. Expand the **Controllers** folder and double click on the **HomeController.cs** file.

The screenshot shows the Microsoft Visual Studio interface. In the center, the code editor displays the `HomeController.cs` file. In the Solution Explorer on the right, the `Controllers` folder under the `OverrideCustomPage` project is expanded, with a red arrow pointing to it. The code editor shows several methods: `NewRdo`, `EditRdo`, and `editRdoModel`.

```

    1  (using ...
    2
    3  namespace OverrideCustomPage.Controllers
    4
    5  public class HomeController : Controller
    6
    7  {
    8      public TapHelper APIHelper { get; set; }
    9
   10     public HomeController()
   11     {
   12     }
   13
   14     [HttpGet]
   15     public async Task<ActionResult> NewRdo(string AppID)...
   16
   17     [HttpPost]
   18     public async Task<ActionResult> NewRdo(NewRdoModel newRdoModel)...
   19
   20     [HttpGet]
   21     public async Task<ActionResult> EditRdo(string AppID, string ArtifactID)...
   22
   23     [HttpPost]
   24     public async Task<ActionResult> editRdo(EditRdoModel editRdoModel)...
   25
   26     private method...
   27
   28 }

```

9. The first two methods with the name **NewRdo** are web pages which contain a form to create a new RDO record.

This screenshot is similar to the previous one, showing the `HomeController.cs` file in the code editor. However, two red arrows point to the `NewRdo` methods: one to the `[HttpGet]` version and another to the `[HttpPost]` version. The Solution Explorer on the right shows the same structure as the previous screenshot.

```

    1  (using ...
    2
    3  namespace OverrideCustomPage.Controllers
    4
    5  public class HomeController : Controller
    6
    7  {
    8      public TapHelper APIHelper { get; set; }
    9
   10     public HomeController()
   11     {
   12     }
   13
   14     [HttpGet]
   15     public async Task<ActionResult> NewRdo(string AppID)... ←
   16
   17     [HttpPost]
   18     public async Task<ActionResult> NewRdo(NewRdoModel newRdoModel)... ←
   19
   20     [HttpGet]
   21     public async Task<ActionResult> EditRdo(string AppID, string ArtifactID)...
   22
   23     [HttpPost]
   24     public async Task<ActionResult> editRdo(EditRdoModel editRdoModel)...
   25
   26     private method...
   27
   28 }

```

10. The next two methods with the name **EditRdo** are web pages which contains a form to edit a RDO record.

```

1  using System;
2  using System.ComponentModel.DataAnnotations;
3  namespace OverrideCustomPage.Controllers
4  {
5      public class HomeController : Controller
6      {
7          private IAPIMapper APIHelper { get; set; }
8
9          public HomeController()
10         : base()
11     {
12     }
13
14     [HttpPost]
15     public async Task<ActionResult> NewRdo(string AppID)
16     {
17         return Ok(await APIHelper.NewRdo(AppID));
18     }
19
20     [HttpPost]
21     public async Task<ActionResult> EditRdo(string AppID, string ArtifactID)
22     {
23         return Ok(await APIHelper.EditRdo(AppID, ArtifactID));
24     }
25
26     [HttpPost]
27     public async Task<ActionResult> EditRdo(EditRdoModel editRdoModel)
28     {
29         return Ok(await APIHelper.EditRdo(editRdoModel));
30     }
31
32     private void Methods()
33     {
34     }
35
36 }

```

11. Next expand the **Models** folder and double click on the **BaseRdo.cs** file. This file contains three properties **Name**, **Phone** and **Email**.

```

1  using System;
2  using System.ComponentModel.DataAnnotations;
3  namespace OverrideCustomPage.Models
4  {
5      public class BaseRdo
6      {
7          private int WorkspaceArtifactID { get; set; }
8
9          [Required(ErrorMessage = "Name is required.")]
10         public String Name { get; set; }
11
12         [Required(ErrorMessage = "Phone is required.")]
13         [RegularExpression("^([A-Za-z0-9]+[.][A-Za-z0-9]+)*@[A-Za-z0-9]+([.][A-Za-z0-9]+)*[.][A-Za-z]{2,}$")]
14         public String Phone { get; set; }
15
16         [Required(ErrorMessage = "Email is required.")]
17         [RegularExpression("^([A-Za-z0-9]+[.][A-Za-z0-9]+)*@[A-Za-z0-9]+([.][A-Za-z0-9]+)*[.][A-Za-z]{2,}$")]
18         public String Email { get; set; }
19
20     }
21
22 }

```

12. Next expand the **Helpers** folder and double click on the **Constants.cs** file. This class contains the GUID's of your application and any other static shared data for your application.

```

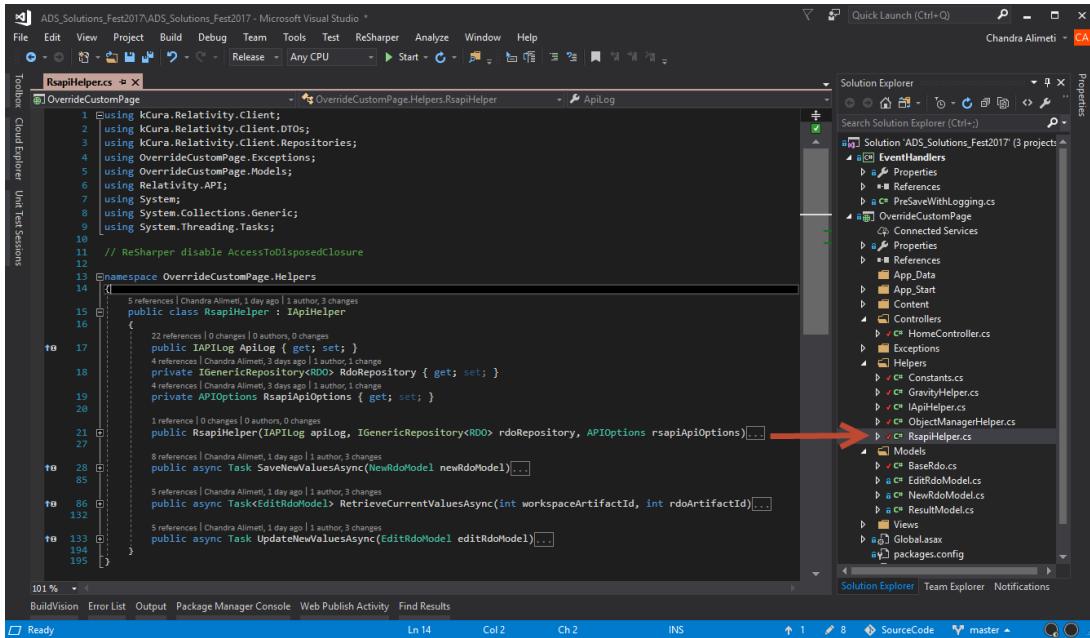
1  using System;
2  namespace OverrideCustomPage.Helpers
3  {
4      public const string ApplicationName = "ADS Solutions Fest 2017";
5      public static readonly Guid Application = new Guid("21D274EA-F22D-428D-A1CE-959F3CBDD6DC");
6      public class ObjectTypes
7      {
8          public static readonly Guid OverrideRdo = new Guid("A9B5E297-6F95-4CE8-A18A-D322D098E579");
9      }
10     public class Fields
11     {
12         public static readonly Guid ArtifacId = new Guid("24514658-CBDA-47F1-94A0-47763ECB2172");
13         public static readonly Guid Name = new Guid("61421508-19C4-41AA-97F0-68BC09F8202C");
14         public static readonly Guid Phone = new Guid("508BF393-E4E8-4B93-AE88-D7FC450AAA");
15         public static readonly Guid Email = new Guid("7C38D7AD-4B39-4DED-A7AD-A87D38A79005");
16     }
17     public class FieldNames
18     {
19         public static readonly string Name = "Name";
20         public static readonly string Phone = "Phone";
21     }
22 }
23 
```

13. Next double click on the **IApiHelper.cs** file to open it. This interface contains the methods to be implemented by the API to read and make any updates to the Override RDO.

```

1  using OverrideCustomPage.Models;
2  using Relativity.API;
3  using System.Threading.Tasks;
4
5  namespace OverrideCustomPage.Helpers
6  {
7      public interface IApiHelper
8      {
9          IAPILog ApiLog { get; set; }
10         Task SaveNewValuesAsync(NewRdoModel newRdoModel);
11         Task<EditRdoModel> RetrieveCurrentValuesAsync(int workspaceArtifactId, int rdoArtifactId);
12         Task UpdateNewValuesAsync(EditRdoModel editRdoModel);
13     }
14 }
15 
```

14. Next, double click on the **RsapiHelper.cs** file to open it. This class is the implementation for the **IApiHelper** interface using the RSAPI.



```

1  Using Kura.Reliability.Client;
2  Using Kura.Reliability.Client.OTOS;
3  Using Kura.Reliability.Client.Repositories;
4  Using OverrideCustomPage.Exceptions;
5  Using OverrideCustomPage.Models;
6  Using Relativity.API;
7  Using System;
8  Using System.Collections.Generic;
9  Using System.Threading.Tasks;
10 // ReSharper disable AccessToDisposedClosure
11
12 [Namespace OverrideCustomPage.Helpers]
13 {
14     [5 references | Chandra Alimeti, 1 day ago | 1 author; 5 changes]
15     [public class RsapiHelper : IApiHelper]
16     {
17         [22 references | 0 changes | 0 authors, 0 changes]
18         [public IAPILog ApiLog { get; set; }]
19         [4 references | Chandra Alimeti, 3 days ago | 1 author, 1 change]
20         [private IGenericRepository<RDO> RdoRepository { get; set; }]
21         [4 references | Chandra Alimeti, 3 days ago | 1 author, 1 change]
22         [private APIOptions RsapiApiOptions { get; set; }]
23
24         [1 reference | 0 changes | 0 authors, 0 changes]
25         [public RsapiHelper(IAPILog apiLog, IGenericRepository<RDO> rdoRepository, APIOptions rsapiApiOptions){...}]
26
27         [8 references | Chandra Alimeti, 1 day ago | 1 author, 3 changes]
28         [public async Task SaveNewValuesAsync(NewRdoModel newRdoModel){...}]
29
30         [5 references | Chandra Alimeti, 1 day ago | 1 author, 3 changes]
31         [public async Task<EditRdoModel> RetrieveCurrentValuesAsync(int workspaceArtifactId, int rdoArtifactId){...}]
32
33         [5 references | Chandra Alimeti, 1 day ago | 1 author, 3 changes]
34         [public async Task UpdateNewValuesAsync(EditRdoModel editRdoModel){...}]
35     }
36 }
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101

```

15. In the **HomeController.cs** file we create a new instance of the **RsapiHelper** class.

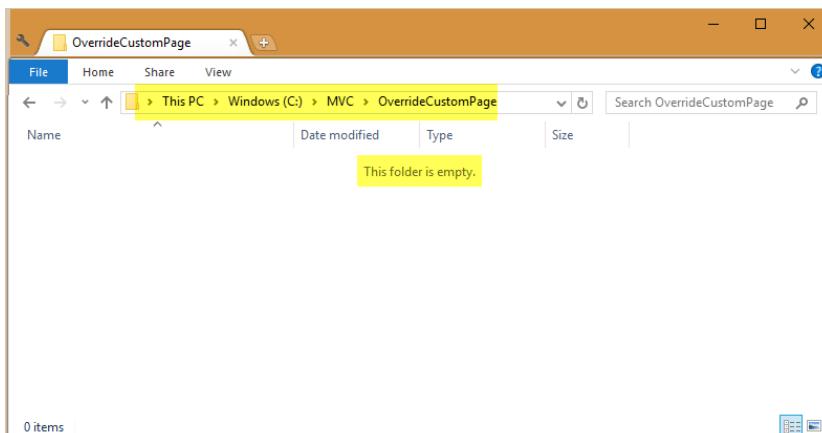
```

public HomeController()
{
    IAPILog apiLog = ConnectionHelper.Helper().GetLoggerFactory().GetLogger();
    APIOptions rsapiApiOptions = new APIOptions
    {
        WorkspaceID = -1
    };
    IRSAPIClient rsapiClient = ConnectionHelper.Helper().GetServicesManager().CreateProxy<IRSAPIClient>(ExecutionIdentity.System);
    rsapiClient.APIOptions = rsapiApiOptions;
    IGenericRepository<RDO> rdoRepository = rsapiClient.Repositories.RDO;

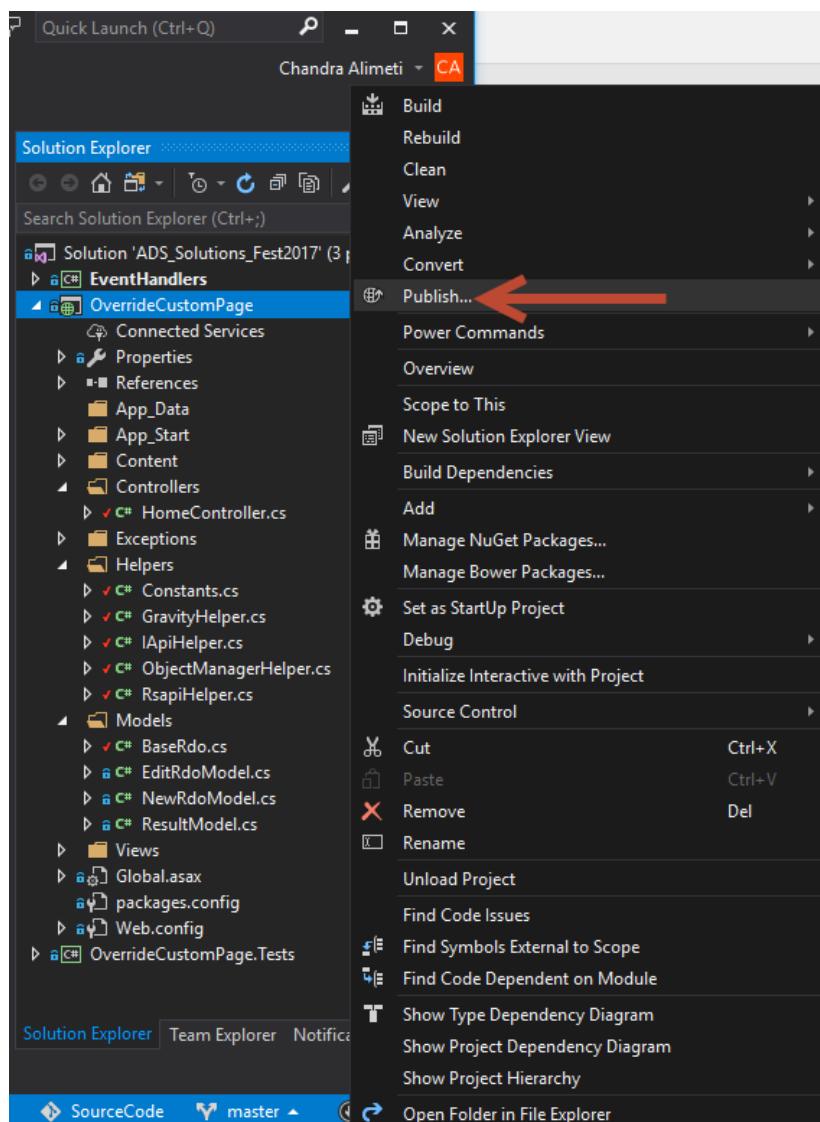
    ApiHelper = new RsapiHelper(apiLog, rdoRepository, rsapiApiOptions);
}

```

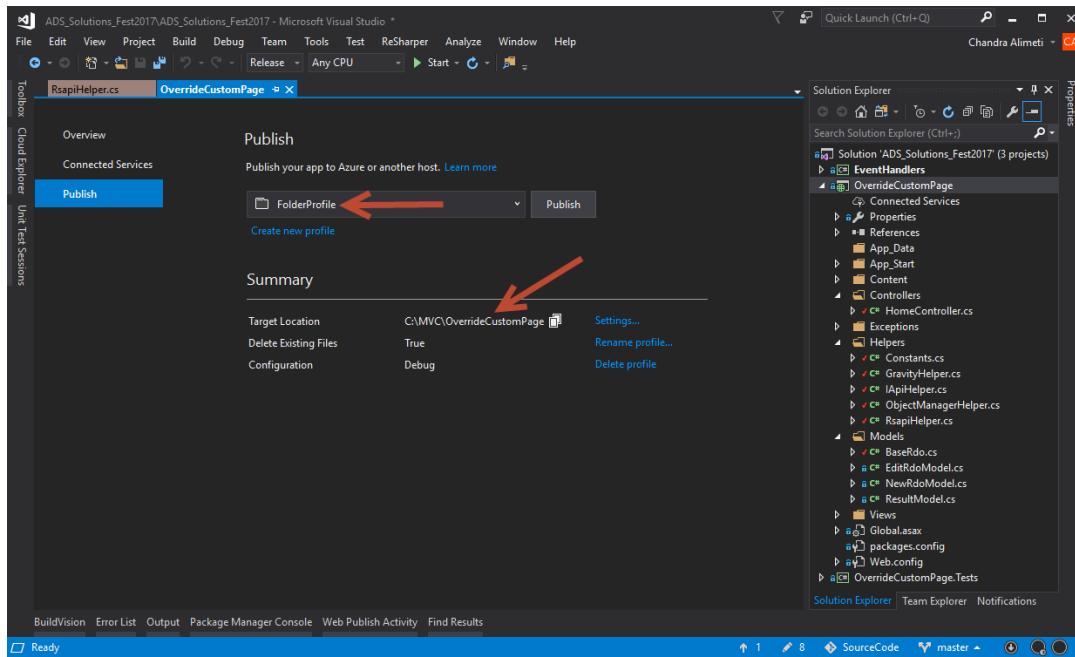
16. Next, we will publish the custom page to a **OverrideCustomPage** folder so that we could zip it up and attach it to our Relativity application. Open the **File Explorer**; make sure **C:\MVC\OverrideCustomPage** folder exists and it's empty.



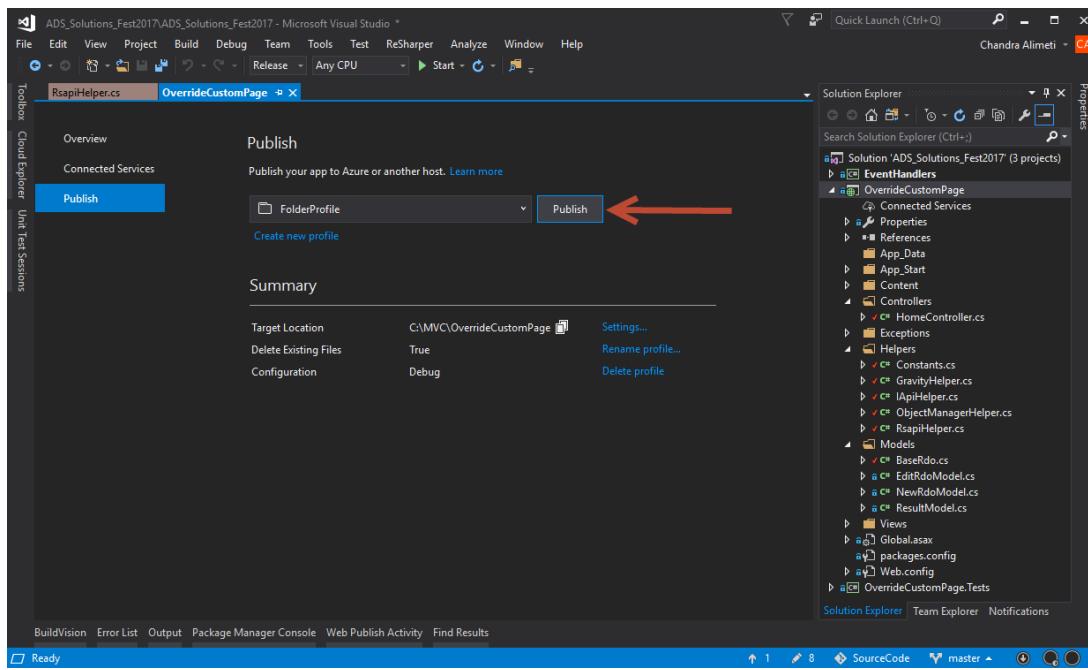
17. Right click on the **OverrideCustomPage** project and click on the **Publish** option.



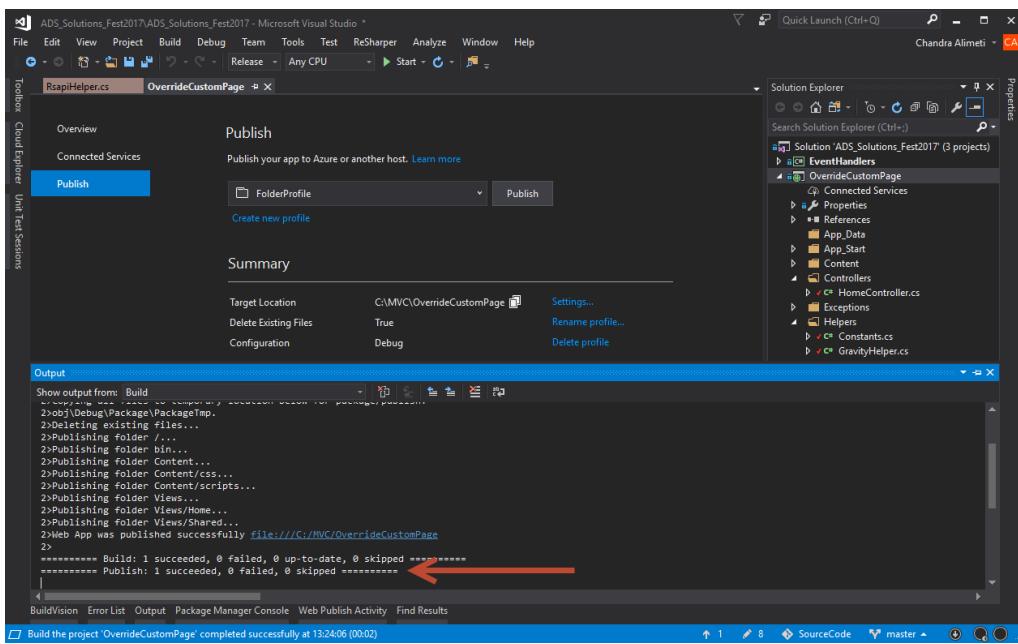
18. You should see a **Publish** settings tab opened. It already contains a **FolderProfile** which is set to publish the custom page to **C:\MVC\OverrideCustomPage** folder.



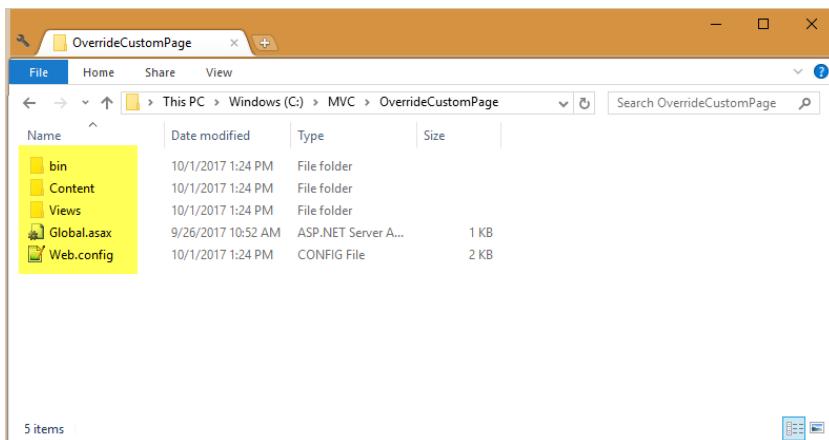
19. Click on the **Publish** button.



20. You will see a confirmation in the **Output** windows saying the custom page was successfully published.



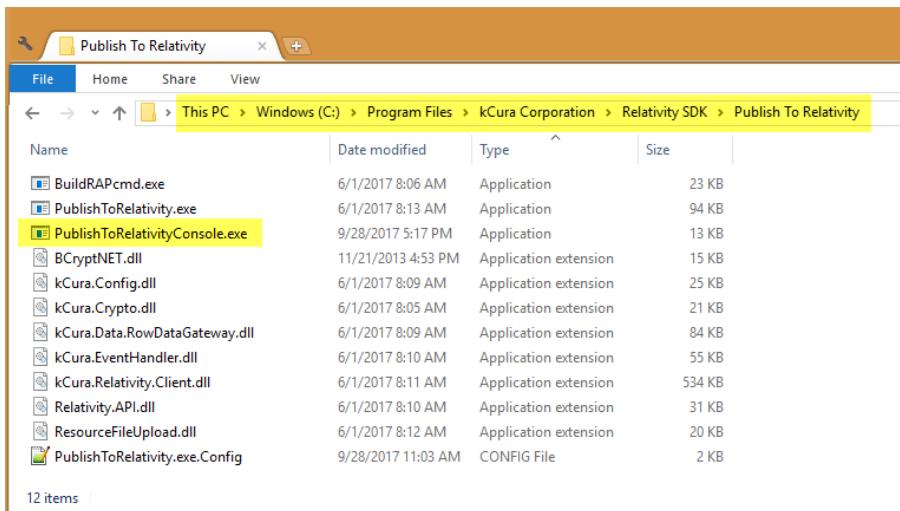
- Now go to the **C:\MVC\OverrideCustomPage** folder to verify the files are copied as shown in the following screenshot.



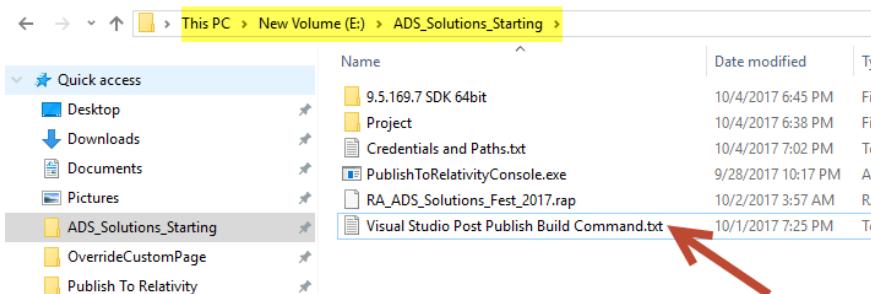
Now we have a custom page published and ready to be attached to our Relativity application.

## 5.2 Visual Studio Post-Build Event & Publish to Relativity Console

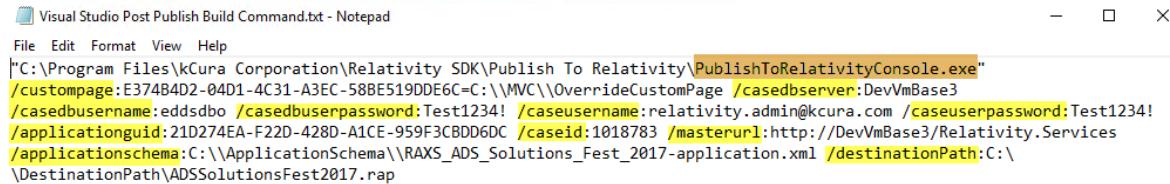
- Instead of using the Publish to Relativity GUI to publish the custom page, we will use a command line tool called **PublishToRelativityConsole** I have created to publish the custom page. The **PublishToRelativityConsole** tool is already copied to the **Publish To Relativity** folder on your workshop machine. Go the **C:\Program Files\kCura Corporation\Relativity SDK\Publish To Relativity** folder and verify the tool exists.



- PublishToRelativityConsole** is an open source project you can find on Github at the link - <https://github.com/relativitydev/publish-to-relativity-console>
- We will also be using **Visual Studio Post-Build event** to call the **PublishToRelativityConsole** tool, which in turn publishes the custom page to Relativity application. You can learn more about Visual Studio Post-Build event at this link: <https://docs.microsoft.com/en-us/visualstudio/ide/specifying-custom-build-events-in-visual-studio>
- Go to the **E:\ADS\_Solutions\_Starting** folder and make sure you see the **Visual Studio Post Publish Build Command** text file.



5. Double click on the **Visual Studio Post Publish Build Command** text file to see its contents.



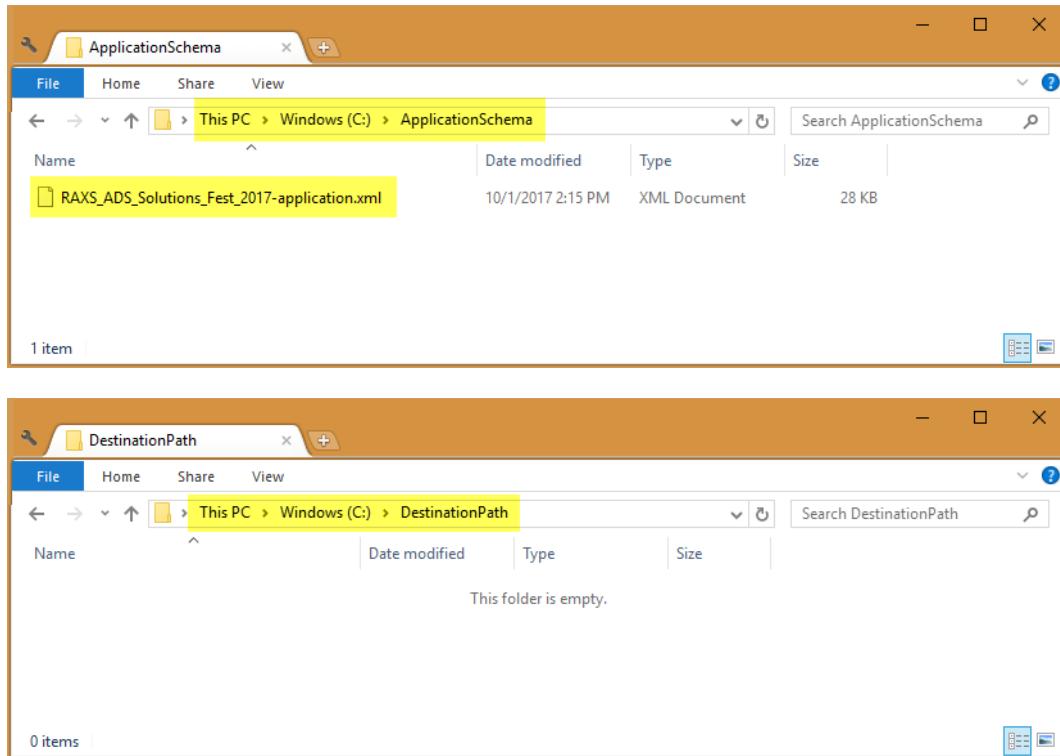
```
"C:\Program Files\kCura Corporation\Relativity SDK\Publish To Relativity\PublishToRelativityConsole.exe"
/customepage:E374B4D2-04D1-4C31-A3EC-58BE519DDE6C=C:\\MVC\\OverrideCustomPage /casedbserver:DevVmBase3
/casedbusername:eddsdbo /casedbuserpassword:Test1234! /caseusername:relativity.admin@kcura.com /caseuserpassword:Test1234!
/applicationguid:21D274EA-F22D-428D-A1CE-959F3CBDD6DC /caseid:1018783 /masterurl:http://DevVmBase3/Relativity.Services
/applicationschema:C:\\ApplicationSchema\\RAXS_ADS_Solutions_Fest_2017-application.xml /destinationPath:C:\\
\\DestinationPath\\ADSSolutionsFest2017.rap
```

6. The post publish build command contains various **arguments** for the **PublishToRelativityConsole** tool and you can find the list below.

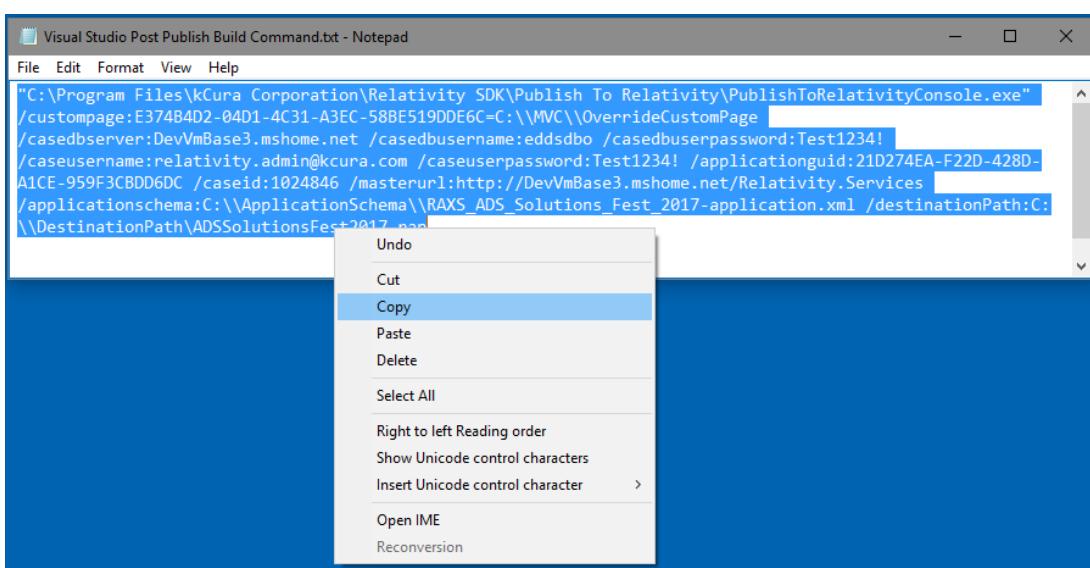
#### Console Arguments:

Argument	Description
"C:\Program Files\kCura Corporation\Relativity SDK\Publish To Relativity\PublishToRelativityConsole.exe"	Path to the PublishToRelativityConsole tool
/customepage: E374B4D2-04D1-4C31-A3EC-58BE519DDE6C=C:\\MVC\\OverrideCustomPage	Custom page GUID and path
/casedbserver:DevVmBase3.mshome.net	SQL Server name
/casedbusername:eddsdbo	SQL Server login
/casedbuserpassword:Test1234!	SQL Server password
/caseusername:relativity.admin@kcura.com	Relativity admin username
/caseuserpassword:Test1234!	Relativity admin password
/applicationguid: 21D274EA-F22D-428D-A1CE-959F3CBDD6DC	Application GUID
/caseid:1024846	Workspace Artifact ID
/masterurl:http://DevVmBase3.mshome.net/Relativity.Services	RSAPI URL
/applicationschema:C:\\ApplicationSchema\\RAXS_ADS_Solutions_Fest_2017-application.xml	Application Schema Path
/destinationPath:C:\\DestinationPath\\ADSSolutionsFest2017.rap	Generated RAP file path

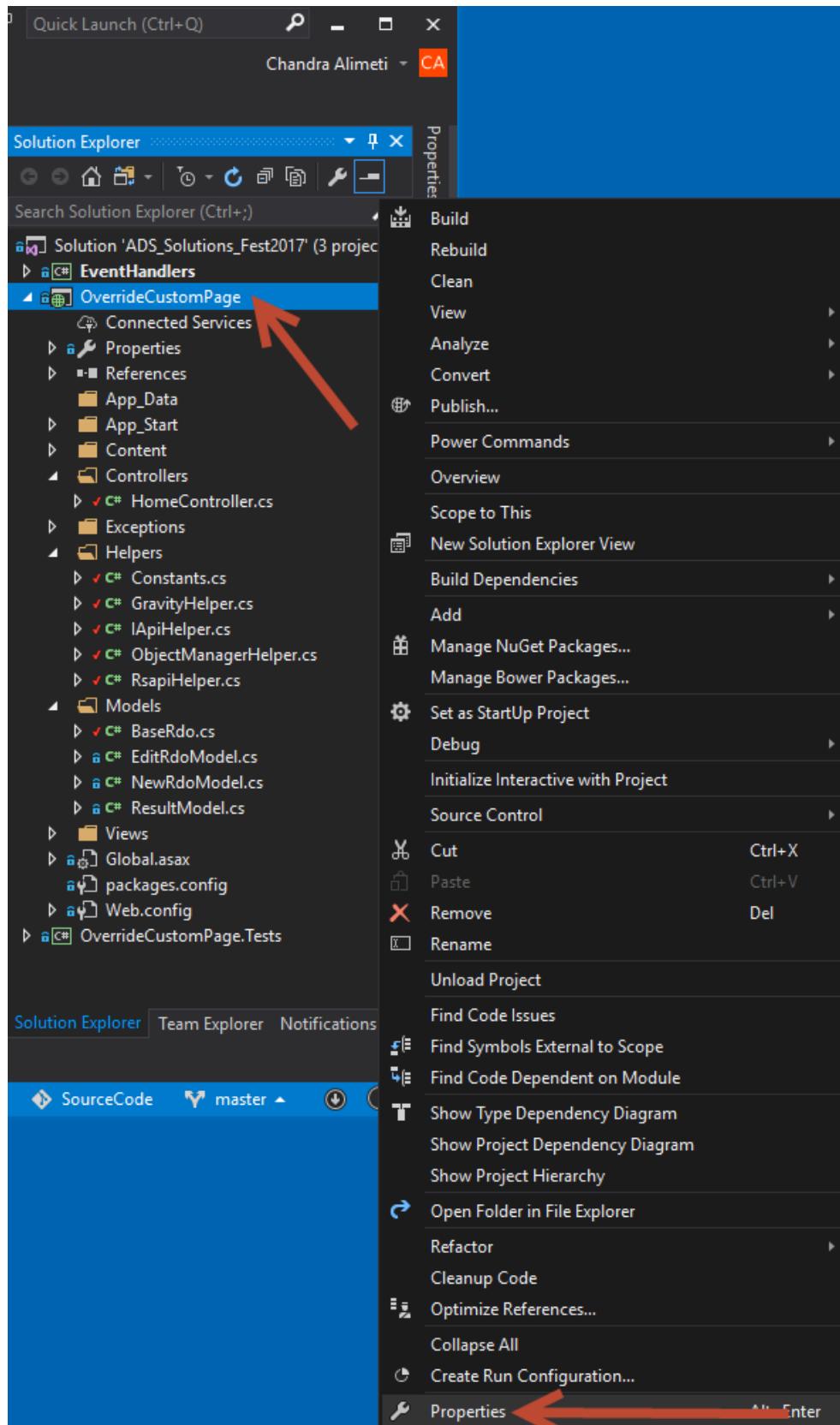
7. The **ApplicationSchema** and **DestinationPath** folders are already created on your workshop machine. The Application schema for the Relativity application is exported from the front-end and copied to the ApplicationSchema folder.



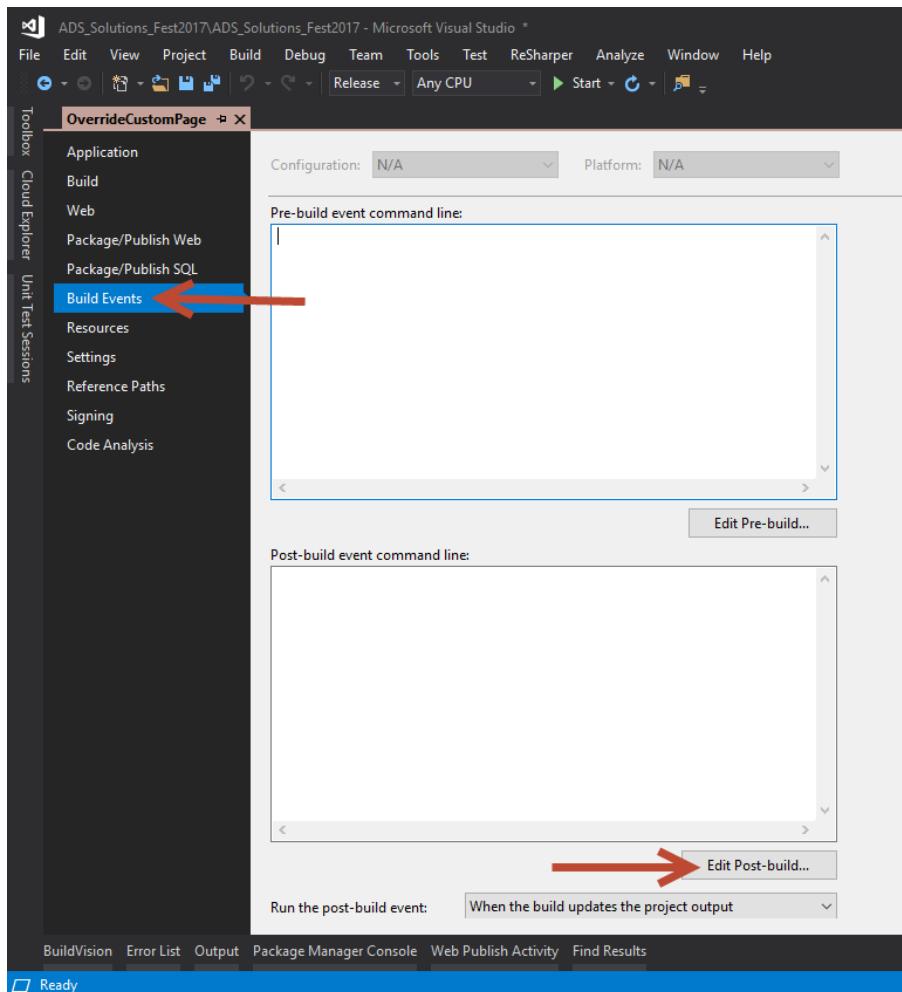
8. Copy the text for the post build publish event.



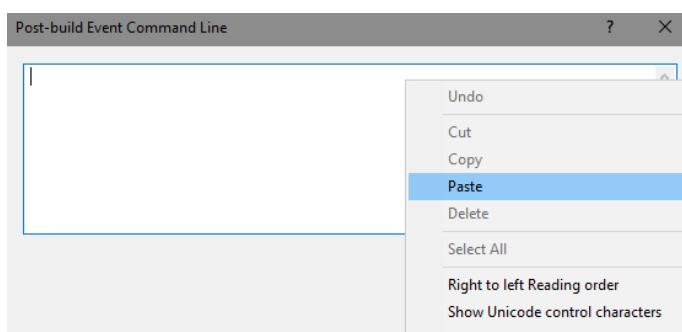
9. Right click on the **OverrideCustomPage** project and select **Properties** option.



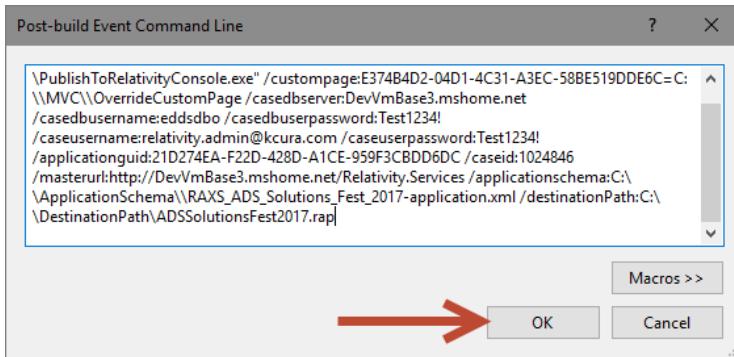
10. Select the **Build Events** option and click on the **Edit Post-build...** button.



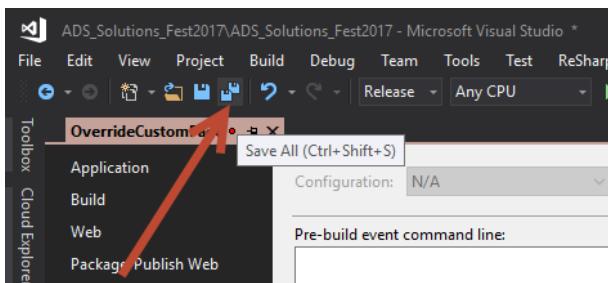
11. Now paste the copied text in the pop up window.



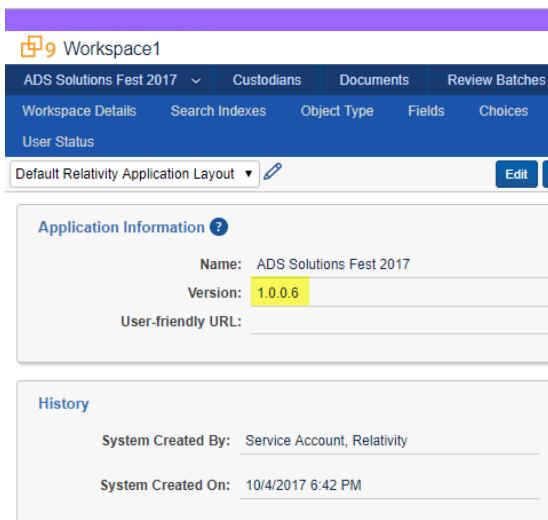
12. Click the **OK** button.



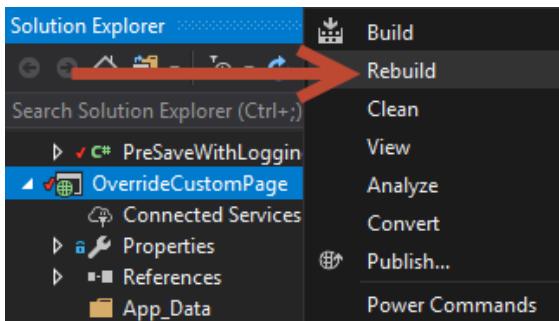
13. Next click the **Save All** button in Visual Studio.



14. Open the **ADS Solutions Fest 2017** Relativity application in the **ADS Solutions Fest 2017 workspace**. Note down the current version of the application.



15. Now right click on the **OverrideCustomPage** project in Visual Studio and select the **Rebuild** option.



16. Now again open the **ADS Solutions Fest 2017** Relativity application in the **ADS Solutions Fest 2017 workspace** or **refresh** the page if it's already opened. Verify that the application version is increased.

The screenshot shows the 'Application Information' section of the Relativity application. It includes fields for 'Name' (ADS Solutions Fest 2017), 'Version' (1.0.0.7, highlighted with a yellow box), and 'User-friendly URL'. Below this is a 'History' section with 'System Created By' (Service Account, Relativity) and 'System Created On' (10/4/2017 6:42 PM).

## 5.3 Object Rules

1. Unlock the Relativity application by clicking the **Unlock application** button.

The screenshot shows the Relativity application's workspace details screen. On the right, a sidebar titled 'RELATIVITY APPLICATION' contains several buttons: 'Manage Application', 'Upgrade Application', 'Export Application', 'Export Application Schema', 'Unlock Application' (which has a red arrow pointing to it), and 'Push to Library'. Below these are links for 'Application Information' and 'Show Application Breakdown'.

2. Next click the **Object Type** tab, search for the **override** word and click the **Override RDO** link.

The screenshot shows the 'Object Type' page in the Relativity application. A red arrow points to the 'Object Type' tab in the top navigation bar. Another red arrow points to the 'Override RDO' link in the table below, which is associated with the 'override' object type.

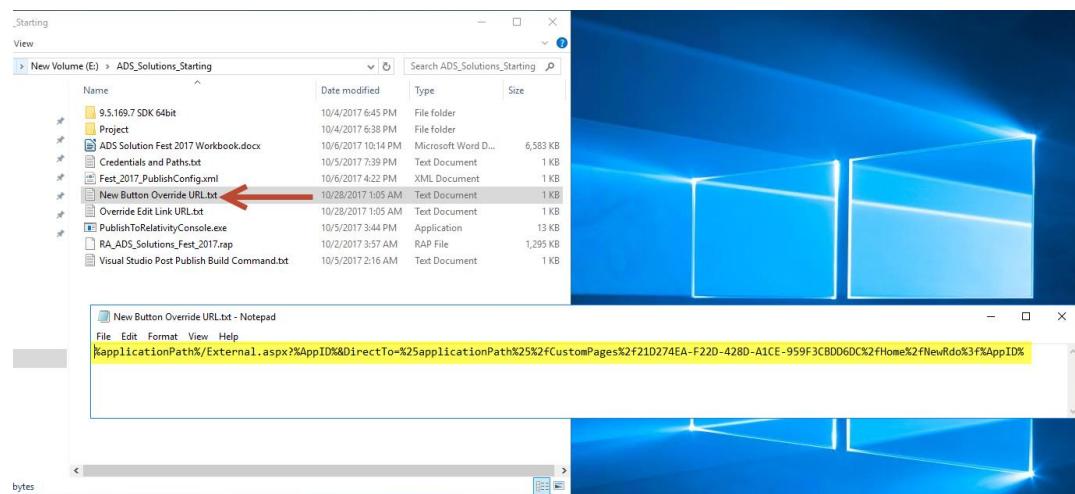
#	Name	System	Created On	Created By
1	override	(All)	(All)	Filter
1	Override RDO	No	10/01/2017 2:30 PM C...	Service Account, Relativity

3. Scroll down to the Rules section and click on the New button.

The screenshot shows the Relativity web interface with the following details:

- Header:** ADS Solutions Fest 2017, devvmbase3.mshome.net/Relativity/Case/ObjectType/View.aspx?AppID=1024846&ArtifactID=1039315&SelectedTab=null, Developer Mode Is Active.
- Main Navigation:** ADS Solutions Fest 2017, Documents, Review Batches, Reporting, Case Admin, Job Admin, Workspace Admin.
- Sub-navigation:** Workspace Details, Search Indexes, Object Type (selected), Fields, Choices, Choices List, Layouts, Views, Tabs, Relativity Applications.
- Buttons:** Edit, Delete, Back, Edit Permissions, View Audit.
- Event Handlers:** A table with columns: DLL, Class Name, Company Name, Description, Execution Type, Application Name. Filter: (All) for all columns.
- Rules:** A table with columns: Name, Type, Field, Value, Action, Artifact ID. A red arrow points to the "New" button.
- Mass Operations:** A table with columns: Name, Type, Custom Page URL, Class Name, Application Name. Filter: 0 Selected Item(s).

4. In the popup window, select **New Button Override** for Rule Type.
  - Enter **New Button Override** for the Name field.
  - For the **Link** field enter the text from the **New Button Override URL.txt** text file located in **E:\ADS\_Solutions\_Starting** folder as shown in the below screenshot.



Relativity - Google Chrome

devvmbase3.mshome.net/Relativity/Controls/ManageObjectRule.aspx?ArtifactTypeID=1000042&AppID=1024846

### Add New Rule

**Save** **Cancel**

Rule Type: New Button Override

Name: New Button Override

Link: %applicationPath%/External.aspx?

5. Click the **Save** button.
6. You can learn more about **New Button Override** at this link:  
[https://platform.relativity.com/9.5/Content/Managing\\_Relativity\\_dynamic\\_objects/RDO\\_9.5/Editing\\_Relativity\\_Objects.htm?Highlight=edit%20link%20override#NewButtonOverride](https://platform.relativity.com/9.5/Content/Managing_Relativity_dynamic_objects/RDO_9.5/Editing_Relativity_Objects.htm?Highlight=edit%20link%20override#NewButtonOverride)
7. Click the **New** button again to add a second rule.

ADS Solutions Fest 2017

Developer Mode Is Active

ADS Solutions Fest 2017

ADS Solutions Fest 2017    Documents    Review Batches    Reporting    Case Admin    Job Admin    Workspace Admin

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

Custom Pages    History    User Status

Edit    Delete    Back    Edit Permissions    View Audit

DLL	Class Name	Company Name	Description	Execution Type	Application Name
(All)	(All)	(All)	(All)	(All)	(All)

**Rules** **New** 

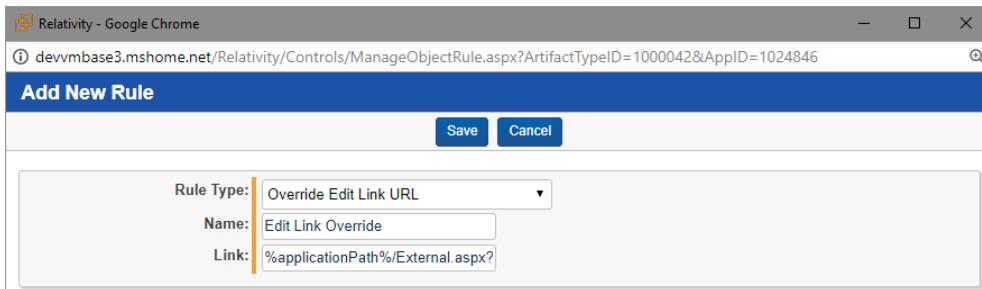
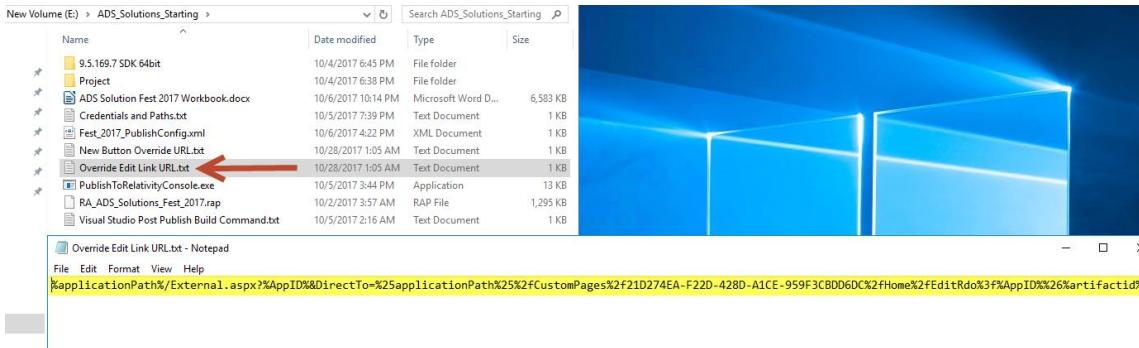
Name	Type	Field	Value	Action	Artifact ID
Edit	New Button Override			%applicationPath%/External.aspx?%AppID%&DirectTo=%25F22D-428D-A1CE-959F3CBD6DC%2fHome	1039322

**Mass Operations** **New** **Delete**

Name	Type	Custom Page URL	Class Name	Application Name
0 Selected Item(s)				Select Page Size: 10

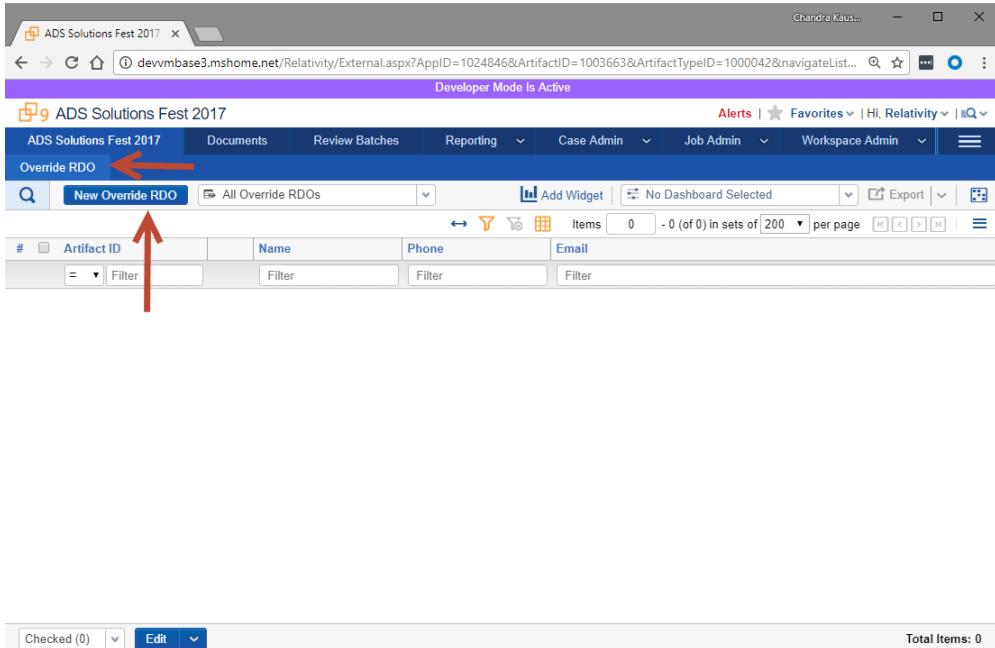
8. In the popup window, select **Override Edit Link URL** for Rule Type.
  - a. Enter **Edit Link Override** for the Name field.
  - b. For the Link field enter the text from the **Override Edit Link URL.txt** text file located in **E:\ADS\_Solutions\_Startup** folder as shown in the below screenshot.

C.



9. Next click the **Save** button.
10. You can learn more about **Override Edit Link URL** at this link:  
[Relativity](https://platform.relativity.com/9.5/Content/Managing_Relativity_dynamic_objects/RDO_9.5/Editing_Relativity_Objects.htm?Highlight=edit%20link%20override#Override>Edit Link URL</a>
</li>
</ol>
</div>
<div data-bbox=)

11. Next go to **Override RDO** tab and click the **New Override RDO** button.



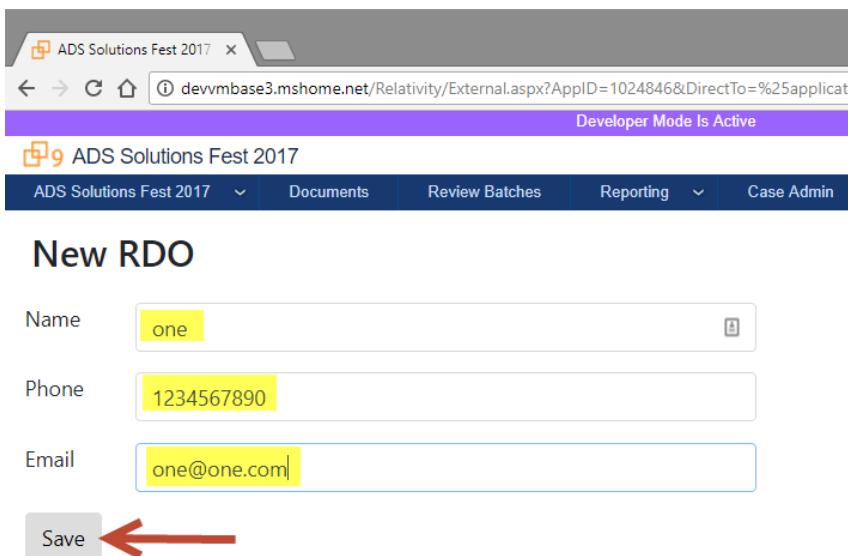
The screenshot shows a web browser window for 'ADS Solutions Fest 2017'. The URL is 'devvmbase3.mshome.net/Relativity/External.aspx?AppID=1024846&ArtifactID=1003663&ArtifactTypeID=1000042&navigateList...'. The top navigation bar includes 'Developer Mode Is Active', 'ADS Solutions Fest 2017', 'Documents', 'Review Batches', 'Reporting', 'Case Admin', 'Job Admin', 'Workspace Admin', and a menu icon. A red arrow points to the 'Override RDO' button in the top left of the main content area. Below it is a search bar with 'New Override RDO' and a dropdown for 'All Override RDOs'. The main table has columns for '#', 'Artifact ID', 'Name', 'Phone', and 'Email'. There are filter buttons for each column. At the bottom, there are buttons for 'Checked (0)', 'Edit', and 'Total Items: 0'.

12. For the form, enter the following values

Name: one

Phone: 1234567890

Email: [one@one.com](mailto:one@one.com)



The screenshot shows a 'New RDO' form. It has three input fields: 'Name' with value 'one', 'Phone' with value '1234567890', and 'Email' with value 'one@one.com'. Below the form is a 'Save' button, which is highlighted with a red arrow pointing to it.

13. Click the **Save** button.

14. Once the RDO record is saved you will be shown a confirmation page as shown in the following screenshot.

The screenshot shows a browser window titled "ADS Solutions Fest 2017". The address bar contains the URL "devvmbase3.mshome.net/Relativity/External.aspx?AppID=1024846&Dir=...". A purple header bar says "Developer Mode Is Active". Below it, the main menu includes "ADS Solutions Fest 2017", "Documents", "Review Batches", "Reporting", and "Case Admin". The current tab is "New RDO". A success message "Your new record was saved successfully. [RSAPI]" is displayed with a red arrow pointing to the "[RSAPI]" part.

15. Next go to **Override RDO** tab and click the **Edit** link for the RDO record created above.

The screenshot shows the "Override RDO" tab selected in the navigation bar. The main content area displays a table with columns: #, Artifact ID, Name, Phone, and Email. A single row is visible with values: 1, 1039324, one, 1234567890, and one@one.com. A red arrow points to the "Edit" link in the second column of this row.

16. In the form, update the value of **Name** to **one-edit** and click on the **Save** button.

The screenshot shows the "Edit RDO" form. It has three input fields: "Name" (containing "one-edit"), "Phone" (containing "1234567890"), and "Email" (containing "one@one.com"). A red arrow points to the "Save" button at the bottom left of the form.

17. Once the RDO record is updated you will be shown a confirmation page as shown in the following screenshot.

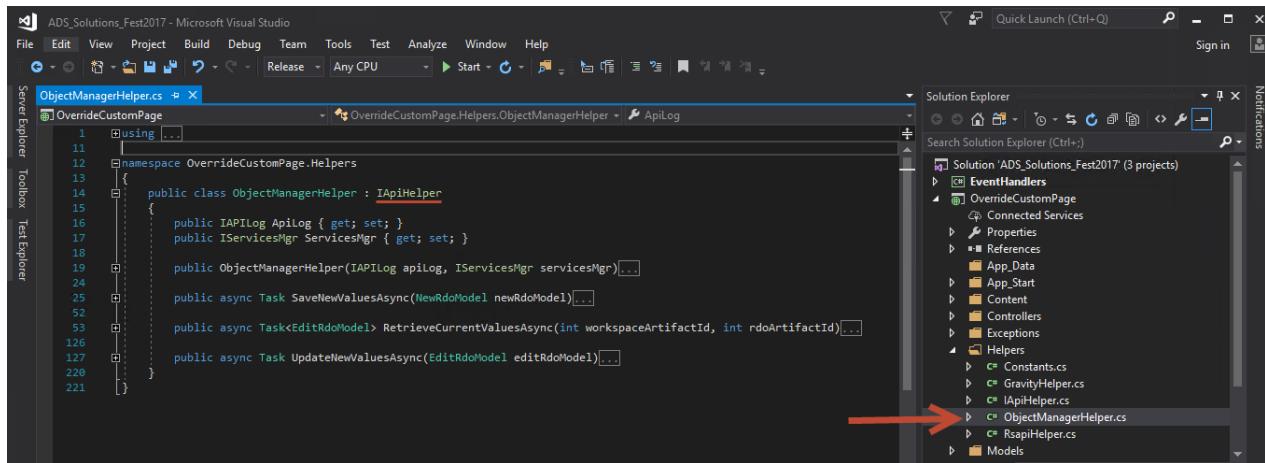
The screenshot shows a web browser window titled 'ADS Solutions Fest 2017'. The address bar contains the URL 'devvmbase3.mshome.net/Relativity/External.aspx?AppID=1024846&DirectTo=%'. A purple header bar indicates 'Developer Mode Is Active'. Below the header, there's a navigation menu with items like 'ADS Solutions Fest 2017', 'Documents', 'Review Batches', 'Reporting', and 'Case Admin'. The main content area is titled 'Edit RDO' and displays the message 'Your record was updated successfully. [RSAPI]'. A red arrow points to the '[RSAPI]' text.

18. Go to **Override RDO** tab to verify the record was successfully updated.

The screenshot shows a web browser window titled 'ADS Solutions Fest 2017'. The address bar contains the URL 'devvmbase3.mshome.net/Relativity/External.aspx?AppID=1024846&ArtifactID=1003663&Artifact'. A purple header bar indicates 'Developer Mode Is Active'. Below the header, there's a navigation menu with items like 'ADS Solutions Fest 2017', 'Documents', 'Review Batches', 'Reporting', 'Case Admin', and 'Job Admin'. The 'Override RDO' tab is selected. The main content area shows a table with columns: '#', 'Artifact ID', 'Name', 'Phone', and 'Email'. There is one row of data: #1, Artifact ID 1039324, Name 'one-edit', Phone '1234567890', and Email 'one@one.com'. The 'Name' column cell for the first row is highlighted with a yellow background.

## 5.4 ObjectManager API

1. Go to Visual Studio and open the **ObjectManagerHelper.cs** file under **Helpers** folder in **OverrideCustomPage** project. This class is the implementation for the **IApiHelper** interface using the ObjectManager API.



The screenshot shows the Microsoft Visual Studio interface. The title bar says "ADS\_Solutions\_Fest2017 - Microsoft Visual Studio". The menu bar includes File, Edit, View, Project, Build, Debug, Team, Tools, Test, Analyze, Window, Help. The toolbar has icons for Save, Undo, Redo, Cut, Copy, Paste, Find, and others. The status bar at the bottom right says "Sign in".

The code editor window is open with the file "ObjectManagerHelper.cs" selected. The code implements the **IApiHelper** interface:

```
1  using ...
11 
12  namespace OverrideCustomPage.Helpers
13  {
14      public class ObjectManagerHelper : IApiHelper
15      {
16          public IAPILog ApiLog { get; set; }
17          public IServicesMgr ServicesMgr { get; set; }
18
19          public ObjectManagerHelper(IAPILog apiLog, IServicesMgr servicesMgr)...
20
21          public async Task SaveNewValuesAsync(NewRdoModel newRdoModel)...
22
23          public async Task<EditRdoModel> RetrieveCurrentValuesAsync(int workspaceArtifactId, int rdoArtifactId)...
24
25          public async Task UpdateNewValuesAsync(EditRdoModel editRdoModel)...
26
27      }
28  }
```

The Solution Explorer window on the right shows the project structure:

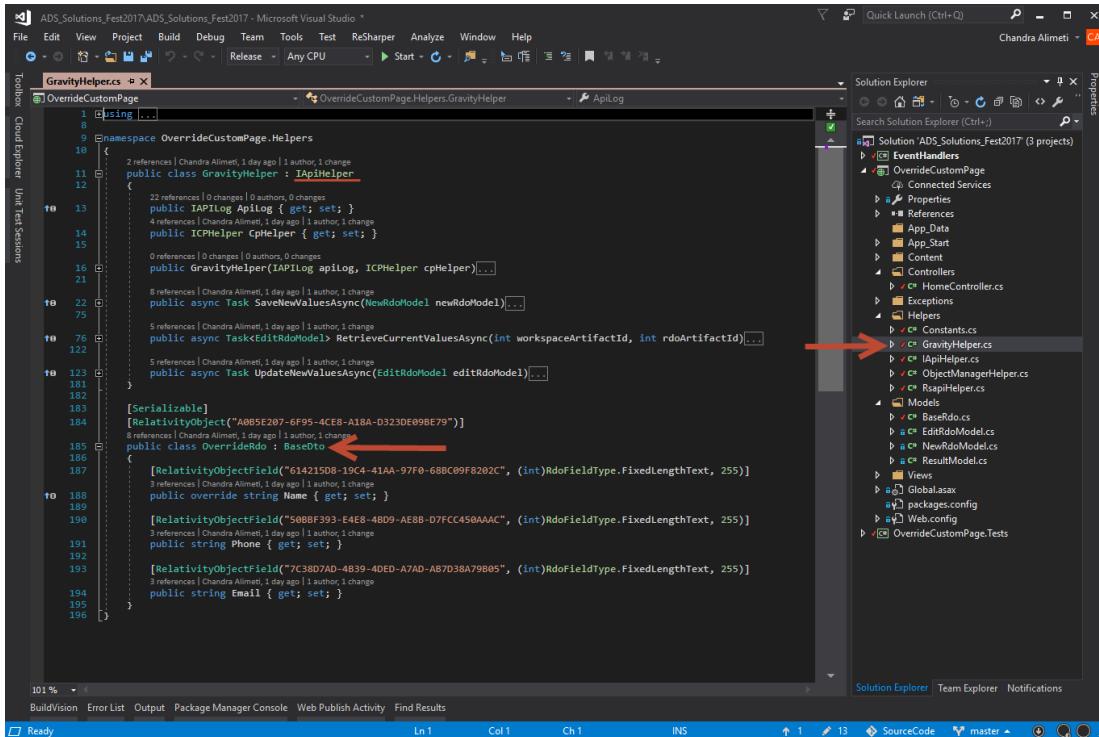
- Solution 'ADS\_Solutions\_Fest2017' (3 projects)
  - EventHandlers
  - OverrideCustomPage
    - Connected Services
    - Properties
    - References
    - App\_Data
    - App\_Start
    - Content
    - Controllers
    - Exceptions
    - Helpers
      - Constants.cs
      - GravityHelper.cs
      - IApiHelper.cs
      - ObjectManagerHelper.cs
      - RsapifHelper.cs
    - Models

A red arrow points to the "ObjectManagerHelper.cs" file in the "Helpers" folder of the "OverrideCustomPage" project.

2. Just take a look at how you write code for the ObjectManager API. We will not be doing any workshop exercise on this.

## 5.5 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/tsdServices/Gravity>
2. Go to Visual Studio and open the **GravityHelper.cs** file under **Helpers** folder in **OverrideCustomPage** project.



3. To use the **Gravity API**, first we must create a class for each RDO or Document with its fields as properties as shown in below screenshot.

```
[Serializable]
[RelativityObject("A0B5E207-6F95-4CE8-A18A-D323DE09BE79")]
public class OverrideRdo : BaseDto
{
    [RelativityObjectField("614215D8-19C4-41AA-97F0-68BC09F8202C", (int)RdoFieldType.FixedLengthText, 255)]
    public override string Name { get; set; }

    [RelativityObjectField("50BBF393-E4E8-4BD9-AE8B-D7FCC450AAC", (int)RdoFieldType.FixedLengthText, 255)]
    public string Phone { get; set; }

    [RelativityObjectField("7C38D7AD-4B39-4DED-A7AD-AB7D38A79B05", (int)RdoFieldType.FixedLengthText, 255)]
    public string Email { get; set; }
}
```

4. Next, we create an instance of **RsapiDao**.

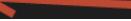
```
RsapiDao gravityRsapiDao = new RsapiDao(CpHelper, newRdoModel.WorkspaceArtifactId);
```

5. Then using the RsapiDao we can call various CRUDQ methods for a Document or RDO.

```
gravityRsapiDao.InsertRelativityObject<OverrideRdo>(newOverrideRdo);
```

6. To use Gravity API in the custom page, open the HomeController.cs file in Visual Studio. Comment the RSAPI code and replace it with the Gravity API code as shown in the below screenshot.

```
public HomeController()
{
    IAPILog apiLog = ConnectionHelper.Helper().GetLoggerFactory().GetLogger();
    //APIOptions rsapiApiOptions = new APIOptions
    //{
    //    WorkspaceID = -1
    //};
    //IRSAPIClient rsapiClient = ConnectionHelper.Helper().GetServicesManager().CreateProxy<IRSAPIClient>(ExecutionIdentity.System);
    //rsapiClient.APIOptions = rsapiApiOptions;
    //IGenericRepository<RDO> rdoRepository = rsapiClient.Repositories.RDO;
    //ApiHelper = new RsapiHelper(apiLog, rdoRepository, rsapiApiOptions);
    ApiHelper = new GravityHelper(apiLog, ConnectionHelper.Helper());
```



#### Code to comment:

```
//APIOptions rsapiApiOptions = new APIOptions
//{
//    WorkspaceID = -1
//};

//IRSAPIClient rsapiClient =
ConnectionHelper.Helper().GetServicesManager().CreateProxy<IRSAPIClient>(ExecutionIdentity.System);

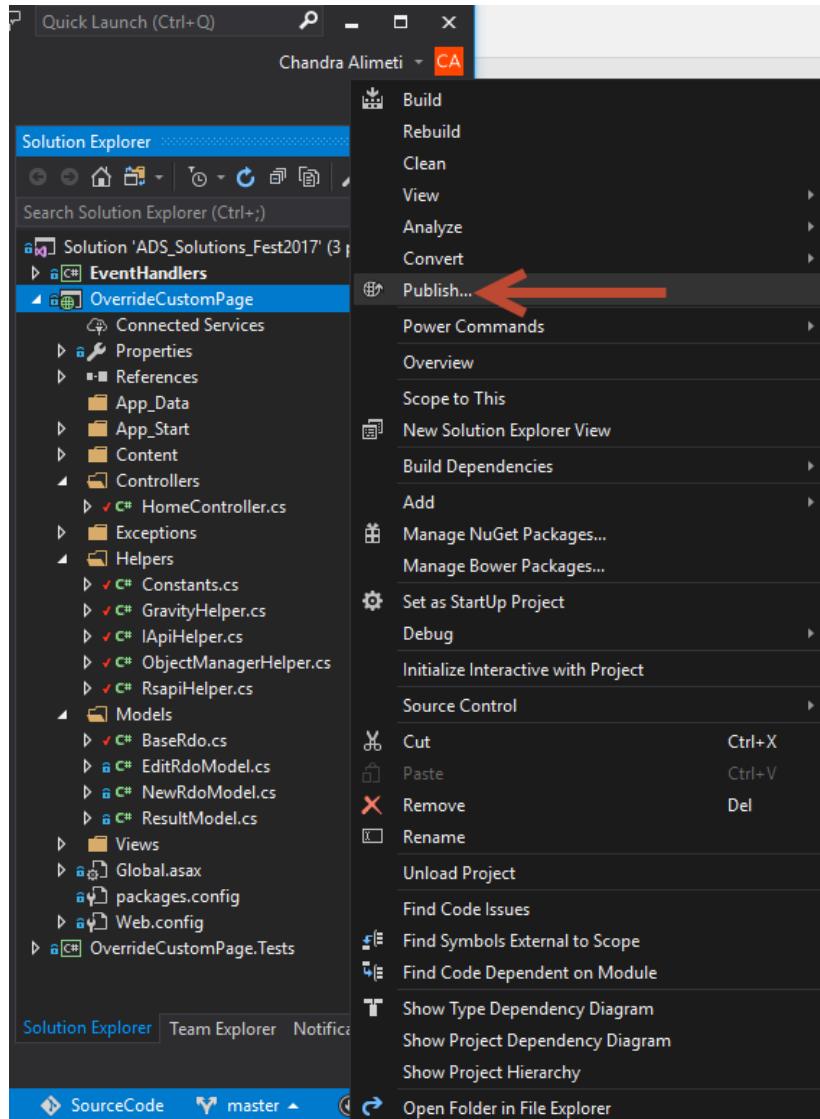
//rsapiClient.APIOptions = rsapiApiOptions;

//IGenericRepository<RDO> rdoRepository = rsapiClient.Repositories.RDO;
//ApiHelper = new RsapiHelper(apiLog, rdoRepository, rsapiApiOptions);
```

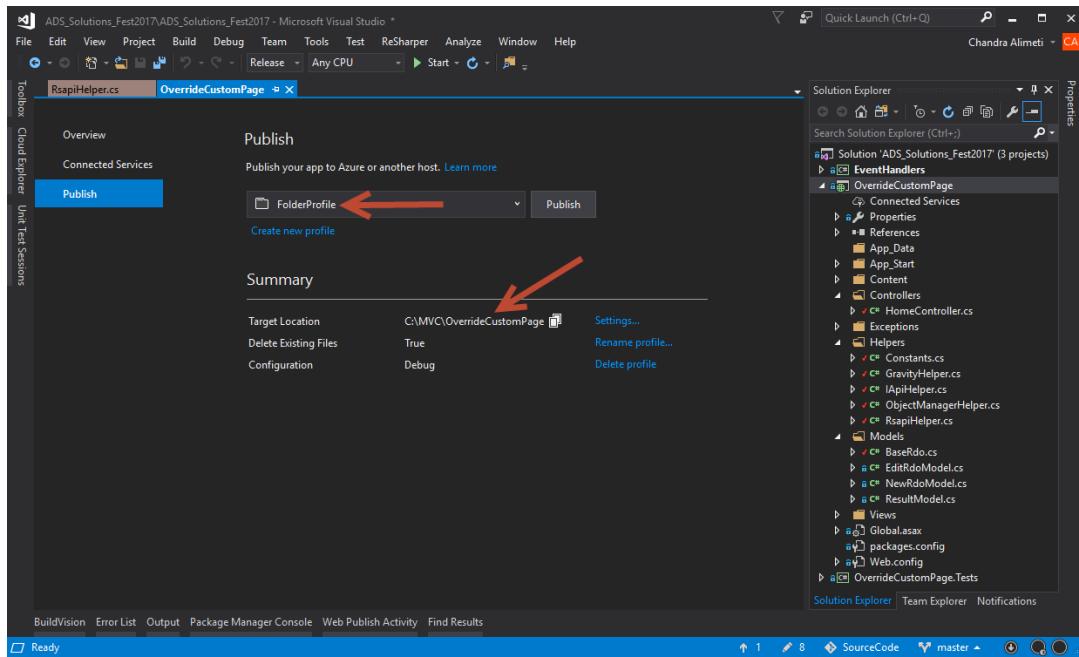
#### Code to add:

```
ApiHelper = new GravityHelper(apiLog, ConnectionHelper.Helper());
```

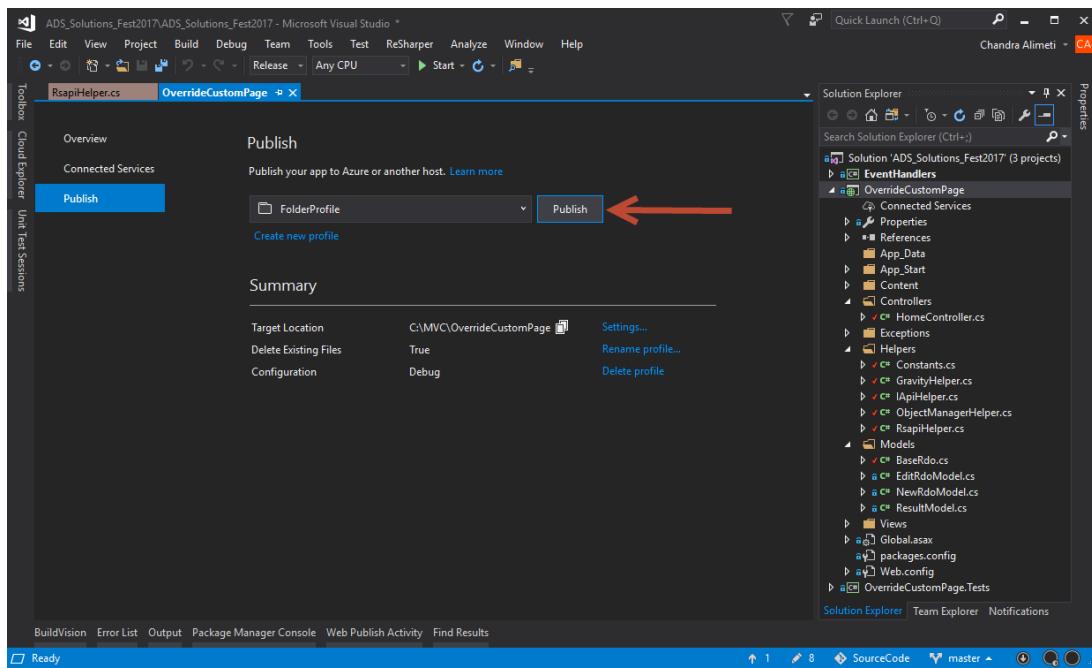
7. Right click on the **OverrideCustomPage** project and click on the **Publish** option.



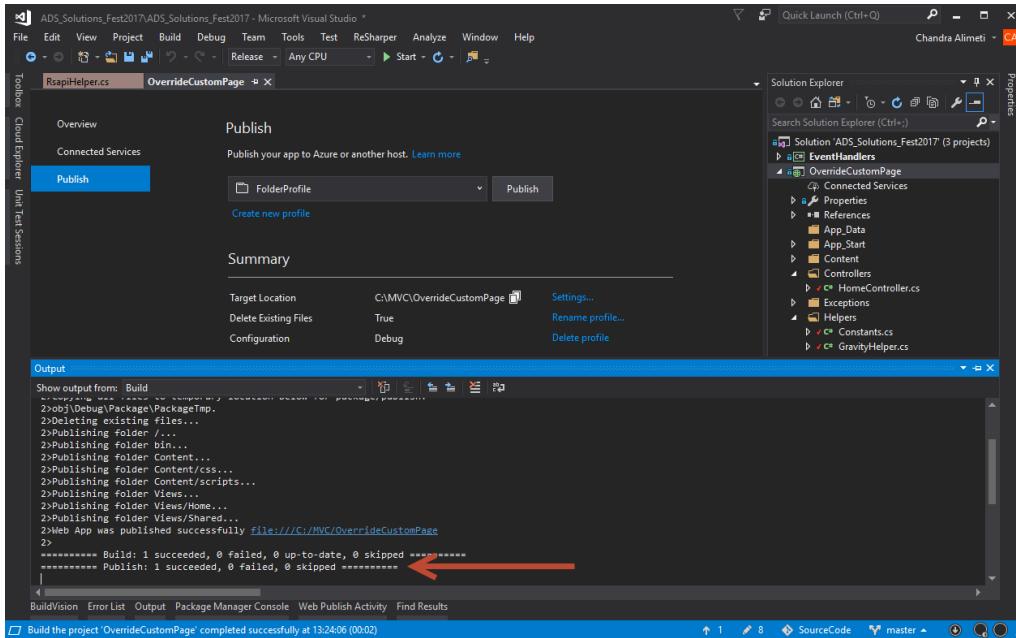
8. You should see a **Publish** settings tab opened. It already contains a **FolderProfile** which is set to publish the custom page to **C:\MVC\OverrideCustomPage** folder.



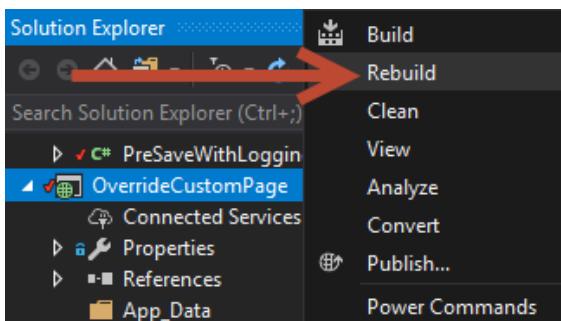
9. Click on the **Publish** button.



10. You will see a confirmation in the **Output** windows saying the custom page was successfully published.



11. Now right click on the **OverrideCustomPage** project in Visual Studio and select the **Rebuild** option.



12. Next go to **Override RDO** tab in Relativity and click the **New Override RDO** button.

The screenshot shows the Relativity web application. At the top, there's a navigation bar with links like 'ADS Solutions Fest 2017', 'Documents', 'Review Batches', 'Reporting', 'Case Admin', and 'Job Admin'. Below the navigation, a blue header bar says 'Override RDO' and has a red arrow pointing to it. Underneath is a search bar with 'New Override RDO' and another red arrow pointing to it. The main content area is a table titled 'Artifact ID' with columns for Name, Phone, and Email. There is one row of data: Artifact ID 1039324, Name 'one-edit', Phone '1234567890', and Email 'one@one.com'.

13. For the form, enter the following values

Name: two  
Phone: 1234567890  
Email: [two@two.com](mailto:two@two.com)

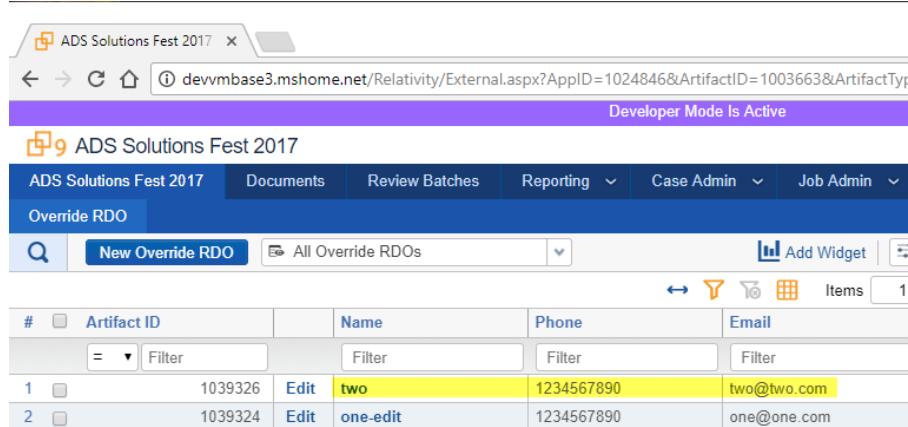
The screenshot shows a web browser window for 'ADS Solutions Fest 2017'. The URL is [devvmbase3.mshome.net/Relativity/External.aspx?AppID=1024846&DirectTo=%25application](http://devvmbase3.mshome.net/Relativity/External.aspx?AppID=1024846&DirectTo=%25application). The page title is 'ADS Solutions Fest 2017'. The main content is a 'New RDO' form with three fields: 'Name' (two), 'Phone' (1234567890), and 'Email' (two@two.com). A red arrow points to the 'Save' button at the bottom left of the form.

14. Click the **Save** button.

15. Once the RDO record is saved you will be shown a confirmation page as shown in the following screenshot.

The screenshot shows a web browser window for 'ADS Solutions Fest 2017'. The URL is [devvmbase3.mshome.net/Relativity/External.aspx?AppID=1024846&DirectTo=%25application](http://devvmbase3.mshome.net/Relativity/External.aspx?AppID=1024846&DirectTo=%25application). The page title is 'ADS Solutions Fest 2017'. The main content is a confirmation message: 'Your new record was saved successfully. [Gravity API]'. A red arrow points to the message text.

16. Go to **Override RDO** tab to verify the record was successfully saved.

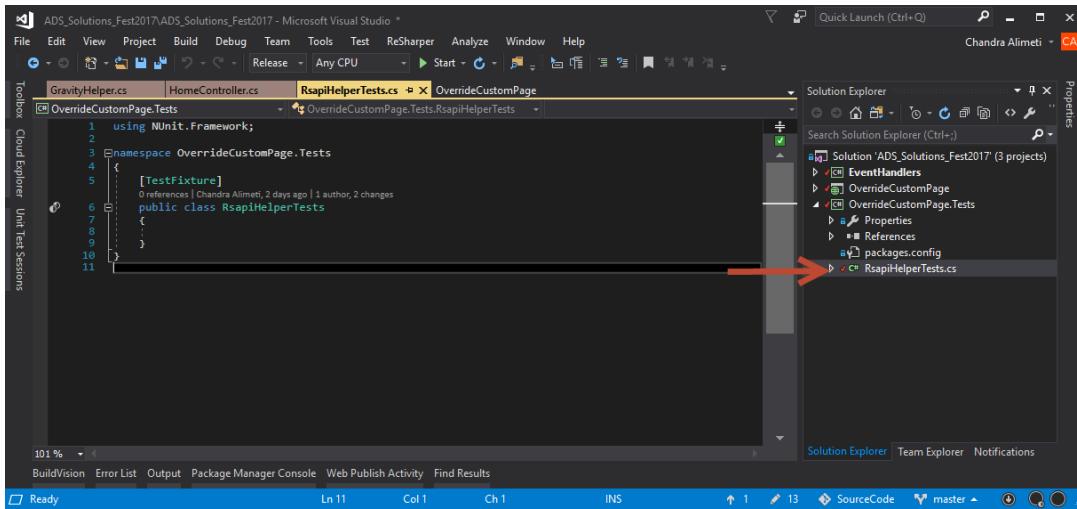


The screenshot shows a web browser window for 'ADS Solutions Fest 2017' at the URL [devvmbase3.mshome.net/Relativity/External.aspx?AppID=1024846&ArtifactID=1003663&ArtifactType=1](http://devvmbase3.mshome.net/Relativity/External.aspx?AppID=1024846&ArtifactID=1003663&ArtifactType=1). A purple bar at the top says 'Developer Mode Is Active'. The main menu includes 'Documents', 'Review Batches', 'Reporting', 'Case Admin', and 'Job Admin'. Below the menu is a search bar with 'New Override RDO' and a dropdown for 'All Override RDOs'. A toolbar has icons for 'Add Widget' and 'Items 1'. A table lists two override records:

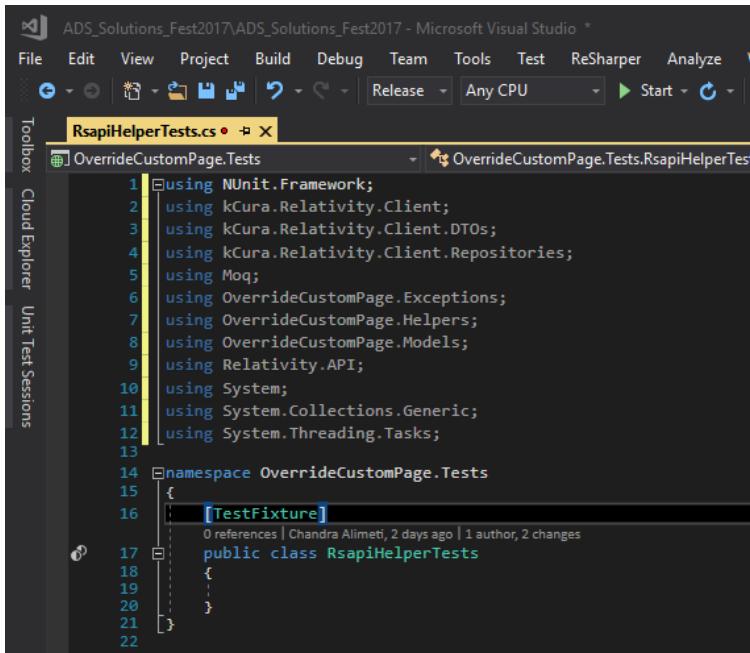
#	Artifact ID	Name	Phone	Email
1	1039326	two	1234567890	two@two.com
2	1039324	one-edit	1234567890	one@one.com

## 6 Unit Tests

1. In this section, we will write unit tests for the **RsapiHelper** class which contains all the RSAPI calls for the custom page.
2. Go to Visual Studio, expand the **OverrideCustomPage.Tests** project.
3. Double click on the **RsapiHelperTests.cs** file. This is an empty class where we will be writing unit tests.



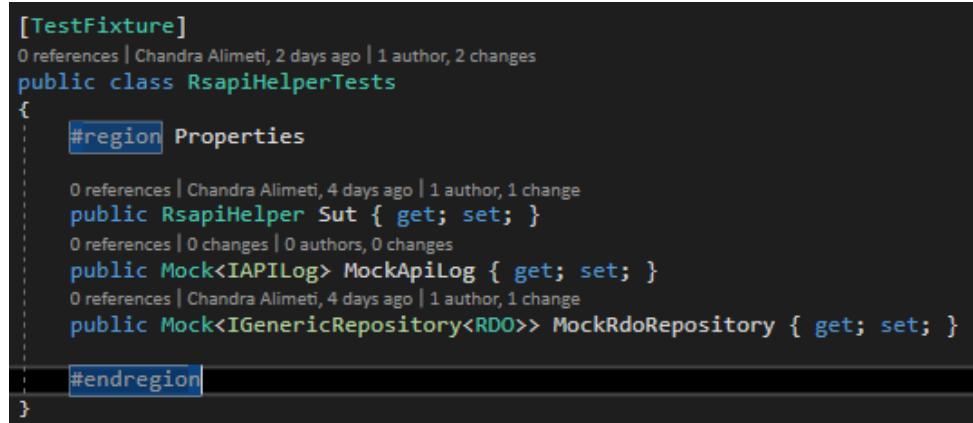
4. Add the following **using** statements to the class if they are not already present.



**Code:**

```
using kCura.Reliability.Client;
using kCura.Reliability.Client.DTOs;
using kCura.Reliability.Client.Repositories;
using Moq;
using OverrideCustomPage.Exceptions;
using OverrideCustomPage.Helpers;
using OverrideCustomPage.Models;
using Reliability.API;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
```

5. Next add the following **properties** to the class.



```
[TestFixture]
0 references | Chandra Alimeti, 2 days ago | 1 author, 2 changes
public class RsapiHelperTests
{
    #region Properties

    0 references | Chandra Alimeti, 4 days ago | 1 author, 1 change
    public RsapiHelper Sut { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public Mock<IAPILog> MockApiLog { get; set; }
    0 references | Chandra Alimeti, 4 days ago | 1 author, 1 change
    public Mock<IGenericRepository<RDO>> MockRdoRepository { get; set; }

    #endregion
}
```

**Code:**

```
#region Properties
    public RsapiHelper Sut { get; set; }
    public Mock<IAPILog> MockApiLog { get; set; }
    public Mock<IGenericRepository<RDO>> MockRdoRepository { get; set; }

#endregion
```

6. Next add the following **SetUp** and **TearDown** methods to the class.

```
#region Setup and TearDown
[SetUp] ←
0 references | Chandra Alimeti, 4 days ago | 1 author, 1 change
public void SetUp()
{
    MockApiLog = new Mock<IAPILog>();
    MockRdoRepository = new Mock<IGenericRepository<RDO>>();
    APIOptions rsapiApiOptions = new APIOptions
    {
        WorkspaceID = -1
    };
    Sut = new RsapiHelper(MockApiLog.Object, MockRdoRepository.Object, rsapiApiOptions);
}

[TearDown] ←
0 references | Chandra Alimeti, 4 days ago | 1 author, 1 change
public void TearDown()
{
    MockRdoRepository = null;
    Sut = null;
}

#endregion
```

**Code:**

```
#region Setup and TearDown

[SetUp]
public void SetUp()
{
    MockApiLog = new Mock<IAPILog>();
    MockRdoRepository = new Mock<IGenericRepository<RDO>>();
    APIOptions rsapiApiOptions = new APIOptions
    {
        WorkspaceID = -1
    };
    Sut = new RsapiHelper(MockApiLog.Object, MockRdoRepository.Object, rsapiApiOptions);
}

[TearDown]
public void TearDown()
{
    MockRdoRepository = null;
    Sut = null;
}

#endregion
```

7. Next add the following **Test constants** to the class. We will be using them in our tests.

```
#region TestConstants

private const int TestWorkspaceArtifactId = 123;
private const int TestRdoArtifactId = 456;
private const string TestName = "name";
private const string TestPhone = "1234567890";
private const string TestEmail = "email@email.com";

#endregion
```

**Code:**

```
#region TestConstants

private const int TestWorkspaceArtifactId = 123;
private const int TestRdoArtifactId = 456;
private const string TestName = "name";
private const string TestPhone = "1234567890";
private const string TestEmail = "email@email.com";

#endregion
```

8. Next, add the following method which mocks the **.LogError** method on the **IApiLog** interface.

```
private void Mock_MockApiLog_LogError_Works()
{
    MockApiLog
        .Setup(x => x.LogError(It.IsAny<string>(), It.IsAny<Exception>()));
}
```

**Code:**

```
private void Mock_MockApiLog_LogError_Works()
{
    MockApiLog
        .Setup(x => x.LogError(It.IsAny<string>(), It.IsAny<Exception>()));
}
```

9. Next, add the following method which verifies that the **.LogError** method on the **IApiLog** interface was called. This will be part of the Assert section of the unit test.

```
private void Verify_Mock_MockApiLog_LogError_Works_Was_Called(int timesCalled)
{
    MockApiLog
        .Verify(x => x.LogError(It.IsAny<string>(), It.IsAny<Exception>())
            , Times.Exactly(timesCalled));
}
```

**Code:**

```
private void Verify_Mock_MockApiLog_LogError_Works_Was_Called(int timesCalled)
{
    MockApiLog
        .Verify(x => x.LogError(It.IsAny<string>(), It.IsAny<Exception>())
            , Times.Exactly(timesCalled));
}
```

10. Next add the following method which mocks the **CreateSingle** method on the **IRSAPIClient** interface.

```
private void Mock_RdoRepository_CreateSingle_Works()
{
    MockRdoRepository
        .Setup(x => x.CreateSingle(It.IsAny<RDO>()));
}
```

**Code:**

```
private void Mock_RdoRepository_CreateSingle_Works()
{
    MockRdoRepository
        .Setup(x => x.CreateSingle(It.IsAny<RDO>()));
}
```

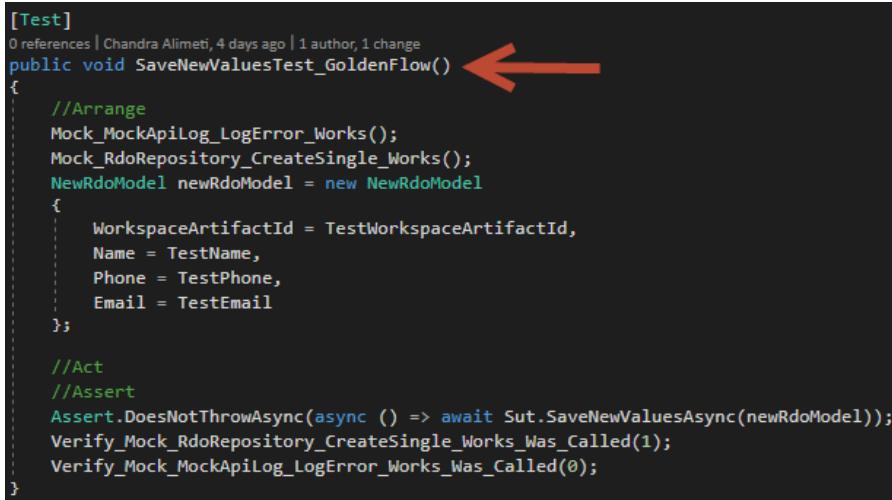
11. Next add the following method which verifies that the the **CreateSingle** method on the **IRSAPIClient** interface was called. This will be part of the Assert section of the unit test.

```
private void Verify_Mock_RdoRepository_CreateSingle_Works_Was_Called(int timesCalled)
{
    MockRdoRepository
        .Verify(x => x.CreateSingle(It.IsAny<RDO>())
            , Times.Exactly(timesCalled));
}
```

**Code:**

```
private void Verify_Mock_RdoRepository_CreateSingle_Works_Was_Called(int timesCalled)
{
    MockRdoRepository
        .Verify(x => x.CreateSingle(It.IsAny<RDO>())
            , Times.Exactly(timesCalled));
}
```

12. Next, add the following unit test for the **SaveNewValues()** method in the **RsapiHelper** class. This unit test tests the scenario where the RSAPI CreateSingle API call works without any errors.



```
[Test]
public void SaveNewValuesTest_GoldenFlow()
{
    //Arrange
    Mock_MockApiLog_LogError_Works();
    Mock_RdoRepository_CreateSingle_Works();
    NewRdoModel newRdoModel = new NewRdoModel
    {
        WorkspaceArtifactId = TestWorkspaceArtifactId,
        Name = TestName,
        Phone = TestPhone,
        Email = TestEmail
    };

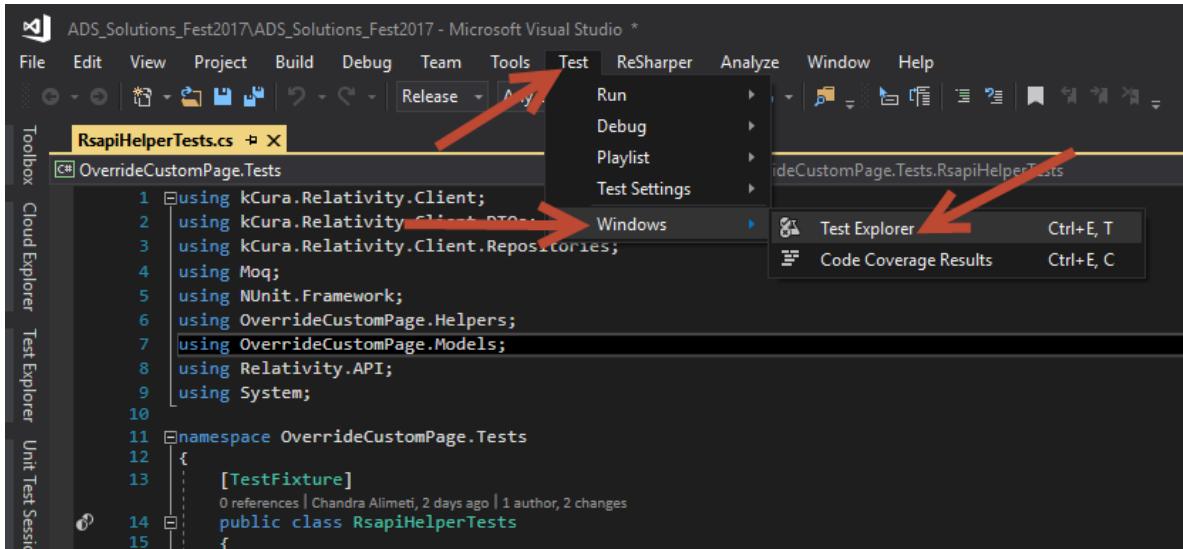
    //Act
    //Assert
    Assert.DoesNotThrowAsync(async () => await Sut.SaveNewValuesAsync(newRdoModel));
    Verify_Mock_RdoRepository_CreateSingle_Works_Was_Called(1);
    Verify_Mock_MockApiLog_LogError_Works_Was_Called(0);
}
```

**Code:**

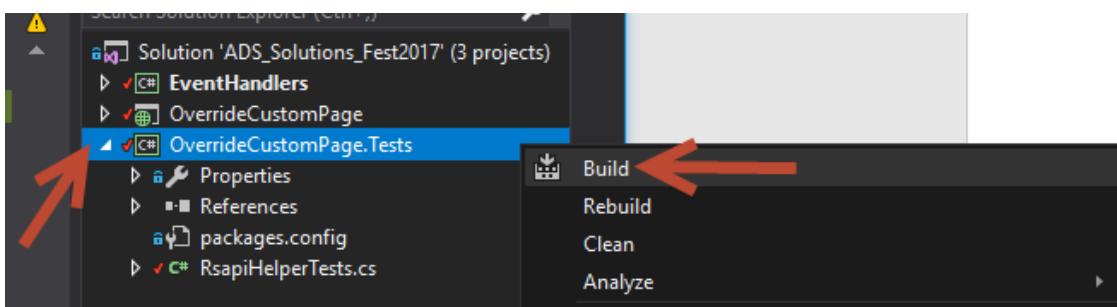
```
[Test]
public void SaveNewValuesTest_GoldenFlow()
{
    //Arrange
    Mock_MockApiLog_LogError_Works();
    Mock_RdoRepository_CreateSingle_Works();
    NewRdoModel newRdoModel = new NewRdoModel
    {
        WorkspaceArtifactId = TestWorkspaceArtifactId,
        Name = TestName,
        Phone = TestPhone,
        Email = TestEmail
    };

    //Act
    //Assert
    Assert.DoesNotThrowAsync(async () => await Sut.SaveNewValuesAsync(newRdoModel));
    Verify_Mock_RdoRepository_CreateSingle_Works_Was_Called(1);
    Verify_Mock_MockApiLog_LogError_Works_Was_Called(0);
}
```

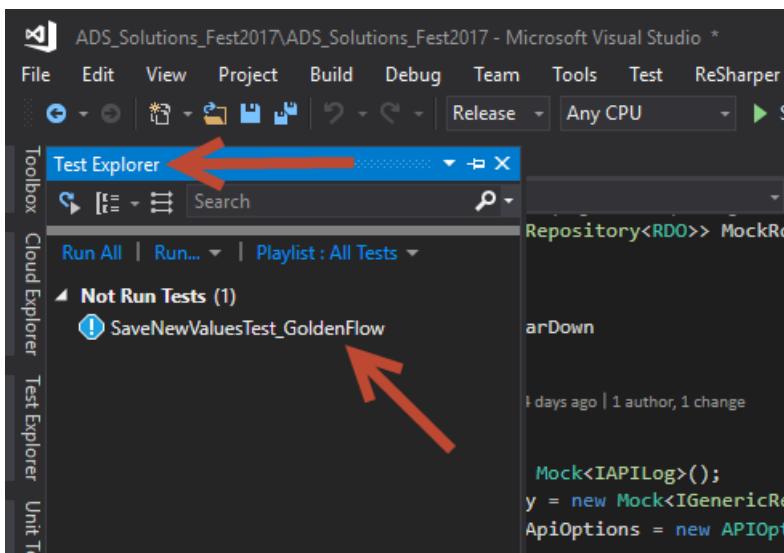
13. Go to Visual Studio, click on **Test** menu option, select **Windows** option, then click on the **Test Explorer** option.



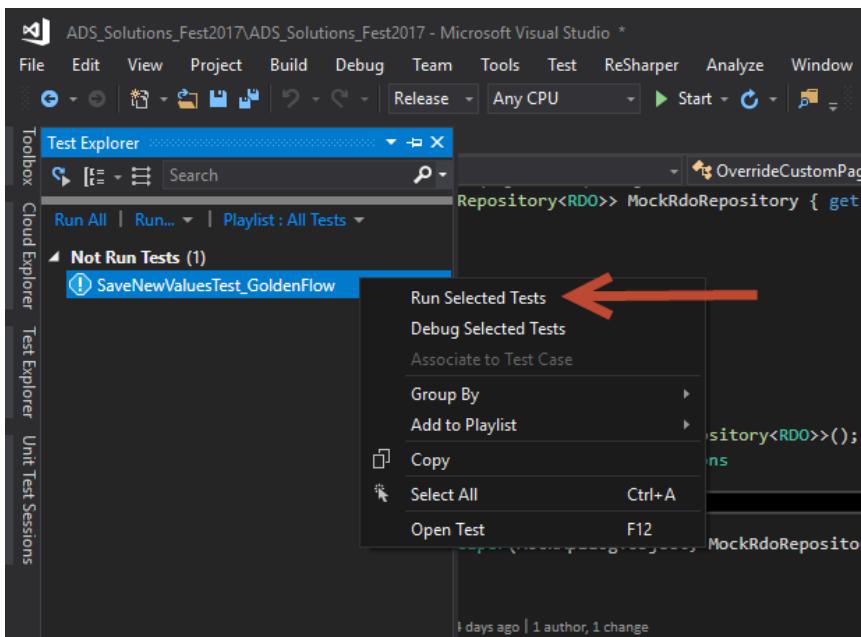
14. Build the project to see the new test in the **Test Explorer** window. Right click on the **OverrideCustomPage.Tests** project and select the **Build** option.



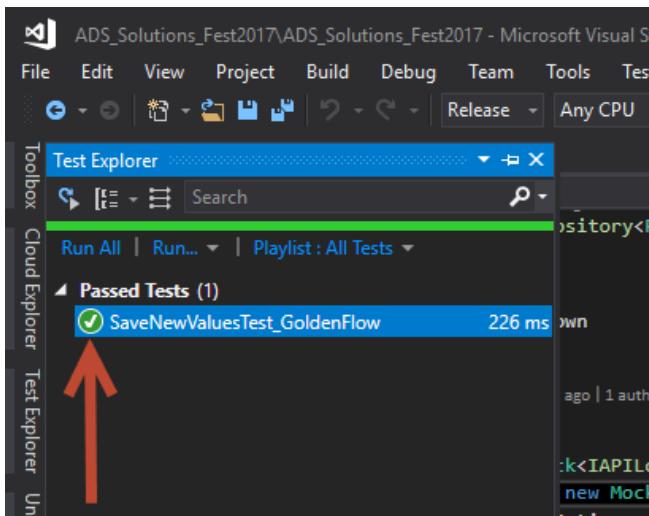
15. In Test Explorer, you should see the **SaveNewValuesTest\_GoldenFlow** test we have just written.



16. Right click on the **SaveNewValuesTest\_GodenFlow** test and select **Run Selected Tests** option.



17. Once the test is finished running, you should see a green check mark next to the test indicating the test has passed.



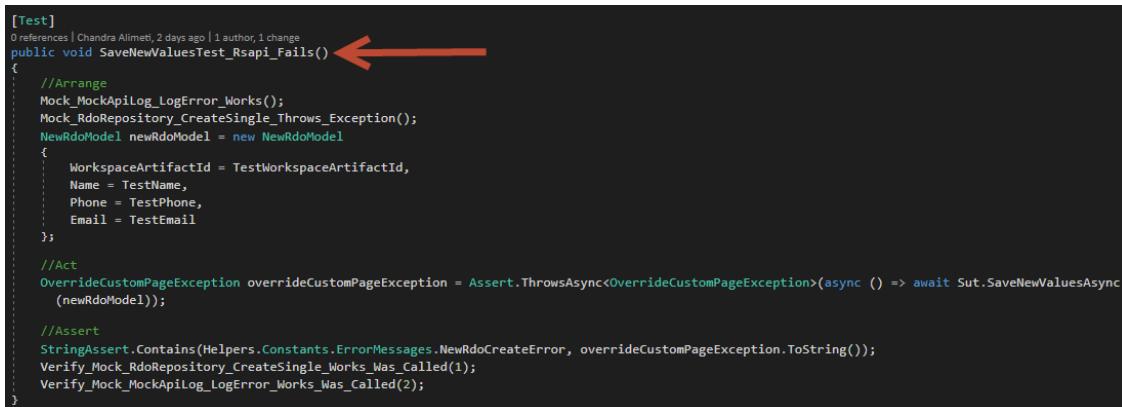
18. Next add the following method which mocks the **CreateSingle** method on the **IRSAPIClient** interface to throw an Exception.

```
private void Mock_RdoRepository_CreateSingle_Throws_Exception()
{
    MockRdoRepository
        .Setup(x => x.CreateSingle(It.IsAny<RDO>()))
        .Throws<Exception>();
```

**Code:**

```
private void Mock_RdoRepository_CreateSingle_Throws_Exception()
{
    MockRdoRepository
        .Setup(x => x.CreateSingle(It.IsAny<RDO>()))
        .Throws<Exception>();
}
```

19. Next add the following unit test for the **SaveNewValues()** method in the **RsapiHelper** class. This unit test tests the scenario where the RSAPI CreateSingle API call throws an exception.



```
[Test]
public void SaveNewValuesTest_Rsapi_Fails() ←
{
    //Arrange
    Mock_MockApiLog_LogError_Works();
    Mock_RdoRepository_CreateSingle_Throws_Exception();
    NewRdoModel newRdoModel = new NewRdoModel
    {
        WorkspaceArtifactId = TestWorkspaceArtifactId,
        Name = TestName,
        Phone = TestPhone,
        Email = TestEmail
    };

    //Act
    OverrideCustomPageException overrideCustomPageException = Assert.ThrowsAsync<OverrideCustomPageException>(async () => await Sut.SaveNewValuesAsync
        (newRdoModel));

    //Assert
    StringAssert.Contains(Helpers.Constants.ErrorMessages.NewRdoCreateError, overrideCustomPageException.ToString());
    Verify_Mock_RdoRepository_CreateSingle_Works_Was_Called(1);
    Verify_Mock_MockApiLog_LogError_Works_Was_Called(2);
}
```

**Code:**

```
[Test]
public void SaveNewValuesTest_Rsapi_Fails()
{
    //Arrange
    Mock_MockApiLog_LogError_Works();
    Mock_RdoRepository_CreateSingle_Throws_Exception();
    NewRdoModel newRdoModel = new NewRdoModel
    {
        WorkspaceArtifactId = TestWorkspaceArtifactId,
        Name = TestName,
        Phone = TestPhone,
        Email = TestEmail
    };

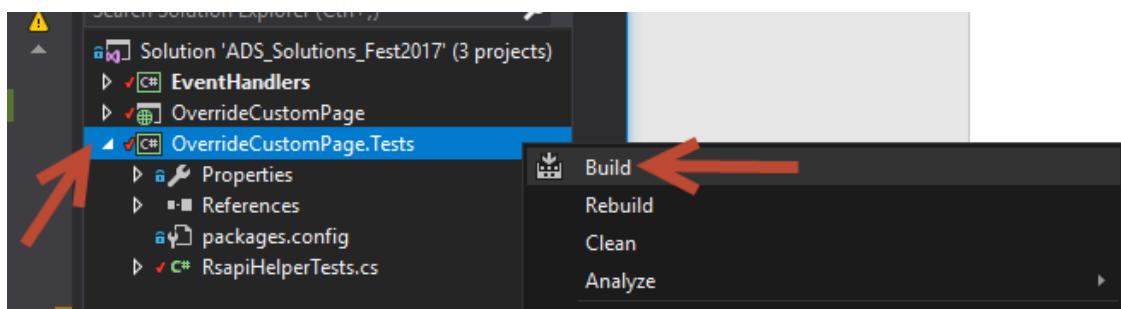
    //Act
    OverrideCustomPageException overrideCustomPageException =
    Assert.ThrowsAsync<OverrideCustomPageException>(async () => await
        Sut.SaveNewValuesAsync(newRdoModel));
```

```

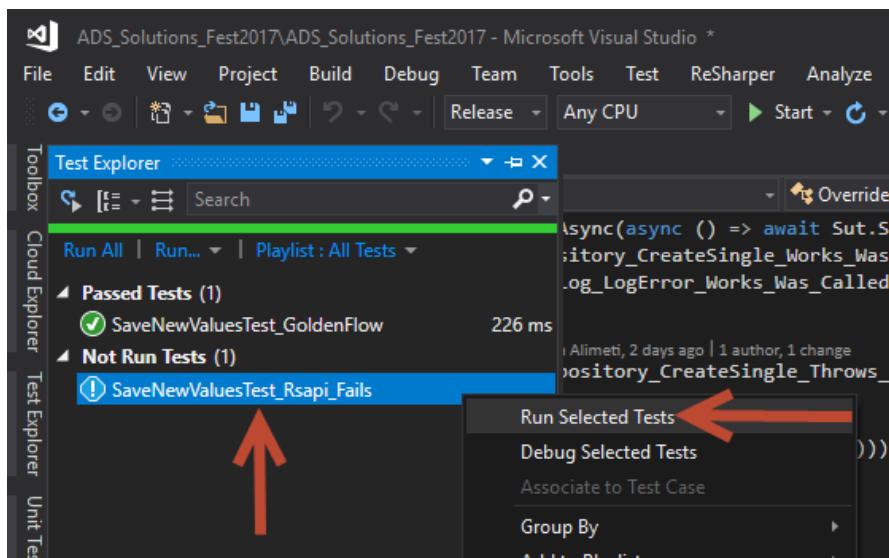
//Assert
StringAssert.Contains(Helper.Constants.ErrorMessages.NewRdoCreateError,
overrideCustomPageException.ToString());
Verify_Mock_RdoRepository_CreateSingle_Works_Was_Called(1);
Verify_Mock_MockApiLog_LogError_Works_Was_Called(2);
}

```

20. Build the project to see the new test in the **Test Explorer** window. Right click on the **OverrideCustomPage.Tests** project and select the **Build** option.



21. Right click on the **SaveNewValuesTest\_Rsapi\_Fails** test and select **Run Selected Tests** option.



22. Once the test is finished running, you should see a green check mark next to the test indicating the test has passed.

