ENABLING NEXT GENERATION AGILE, ADAPTIVE AND PROCESS-MANAGED ENTERPRISE

Getting Started Tutorial For ASP.NET Developers

Table of Contents

Disclaimer

AgilePoint Inc. makes no representations or warranties, either expressed or implied, by or with respect to anything in this document, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose or for any indirect, special or consequential damages.

Introduction

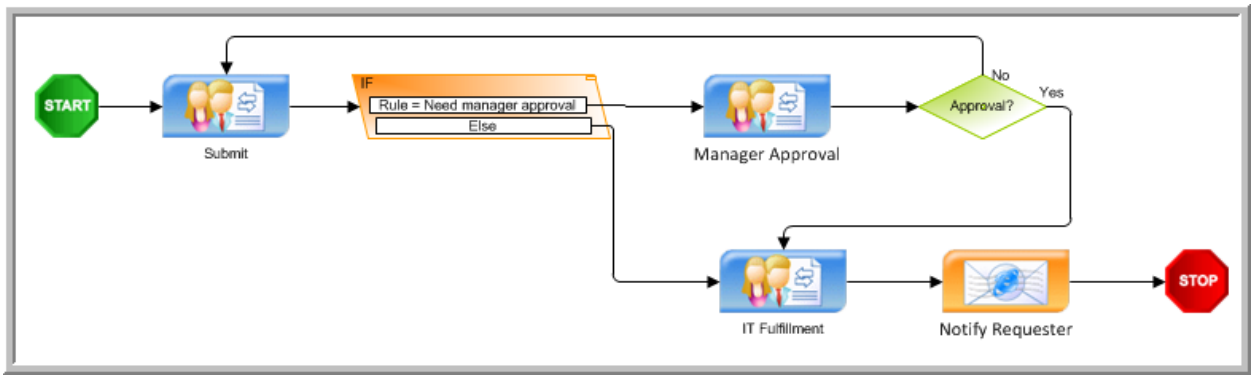
ASP.NET is a Web application framework developed and marketed by Microsoft to allow programmers to build dynamic Web sites, Web applications and w services. It was first released in January 2002 with version 1.0 of the .NET Framework, and is the successor to Microsoft's Active Server Pages (ASP) technology. ASP.NET is built on the Common Language Runtime (CLR), allowing programmers to write ASP.NET code using any supported .NET language.

ASP.NET is one of the out of the box supported form technologies. AgilePoint leverages ASP.NET framework to deliver enterprise level BPM solution in Windows Azure. AgilePoint provides its own web site template, which makes necessary infrastructure available to use in association with ASP.NET.

This document explains step by step process to create simple ASP.Net web site hosted in Windows Azure which uses AgilePoint process model.

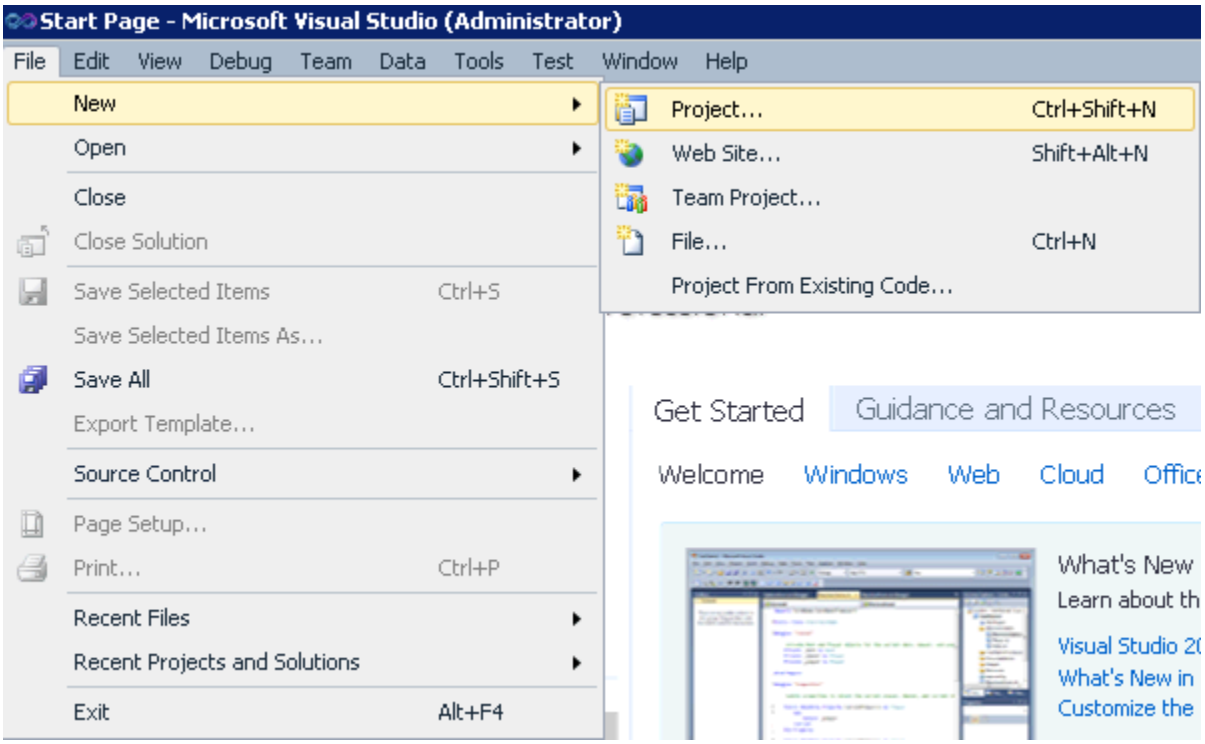
Create ASP.NET Web Site

This section explains the steps required to create an AgilePoint enabled ASP.Net web site for Windows Azure based on the process model designed in "Process Designer" guide. The first step submits the IT equipment request, and the second manual step deals with Manger Approval. Once the request is approved by manager, it is sent to IT group to fulfill the request. Let's create an ASP.Net web site using AgilePoint Developer component.



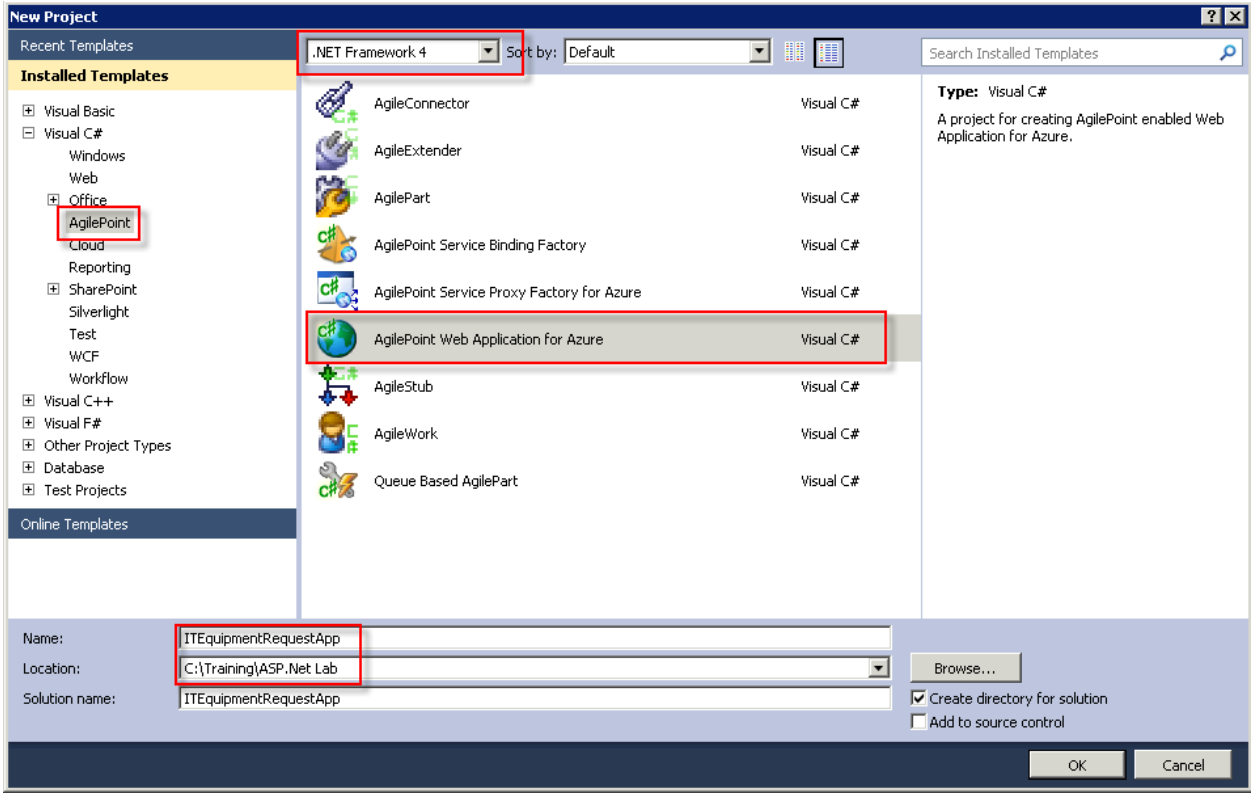
Picture: Process Design for IT Equipment

1. Launch Visual Studio.
2. Click on **File->New->Project** from the menu bar.



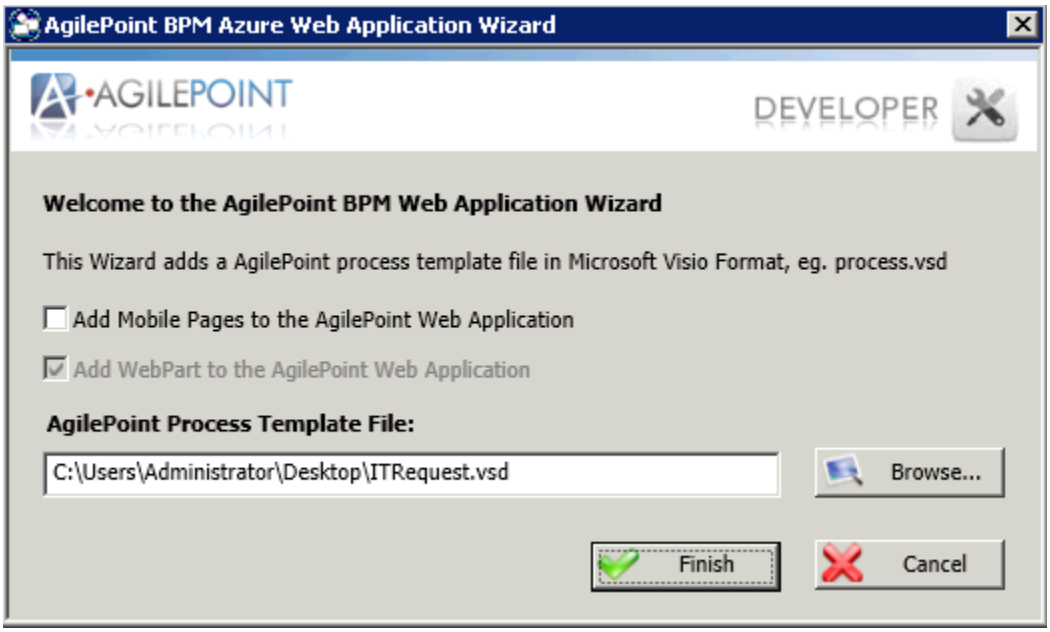
Picture: Creating ASP.Net Web Site

3. Select “AgilePoint” under Visual C# templates in the left hand panel. Please make sure you have selected .Net Framework 4.0 as shown in screenshot below. Then select “**AgilePoint Web Application for Azure**” project template. Enter name of the web application as “**ITEquipmentRequestApp**”, and location as “**C:\Training\ASP.Net Lab**”. Click “OK” button.



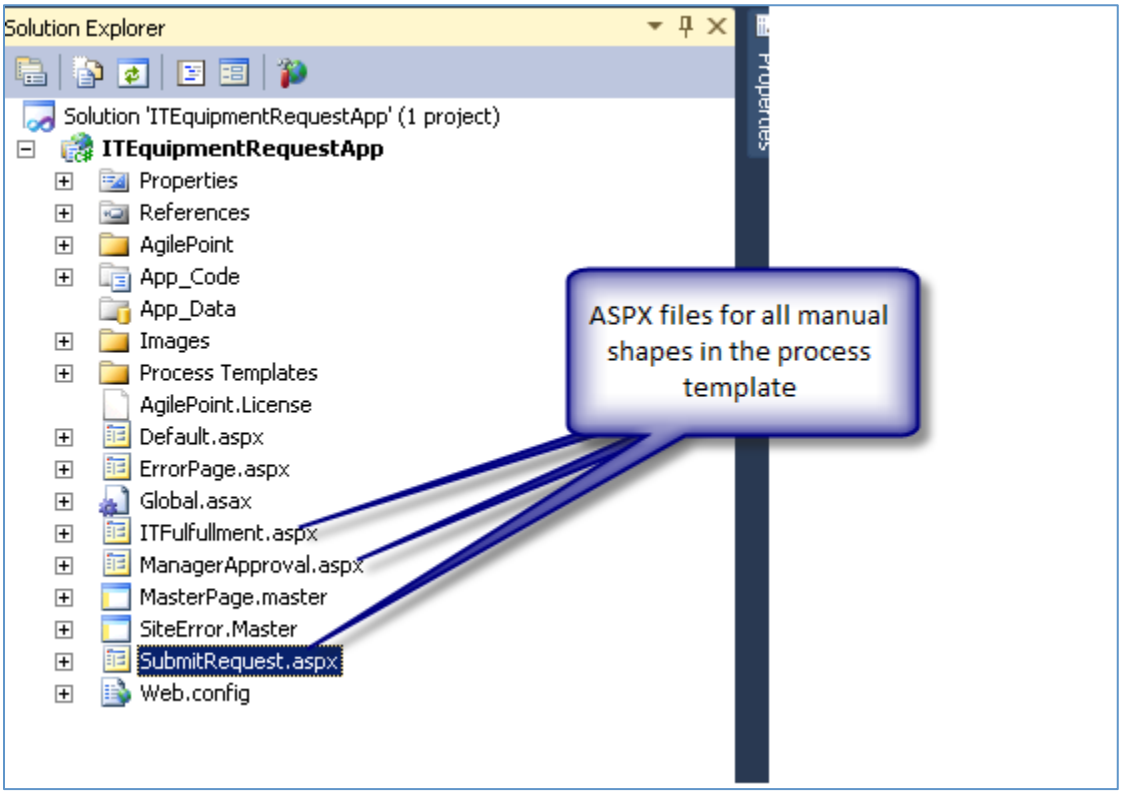
Picture: Creation of AgilePoint web application using AgilePoint Developer

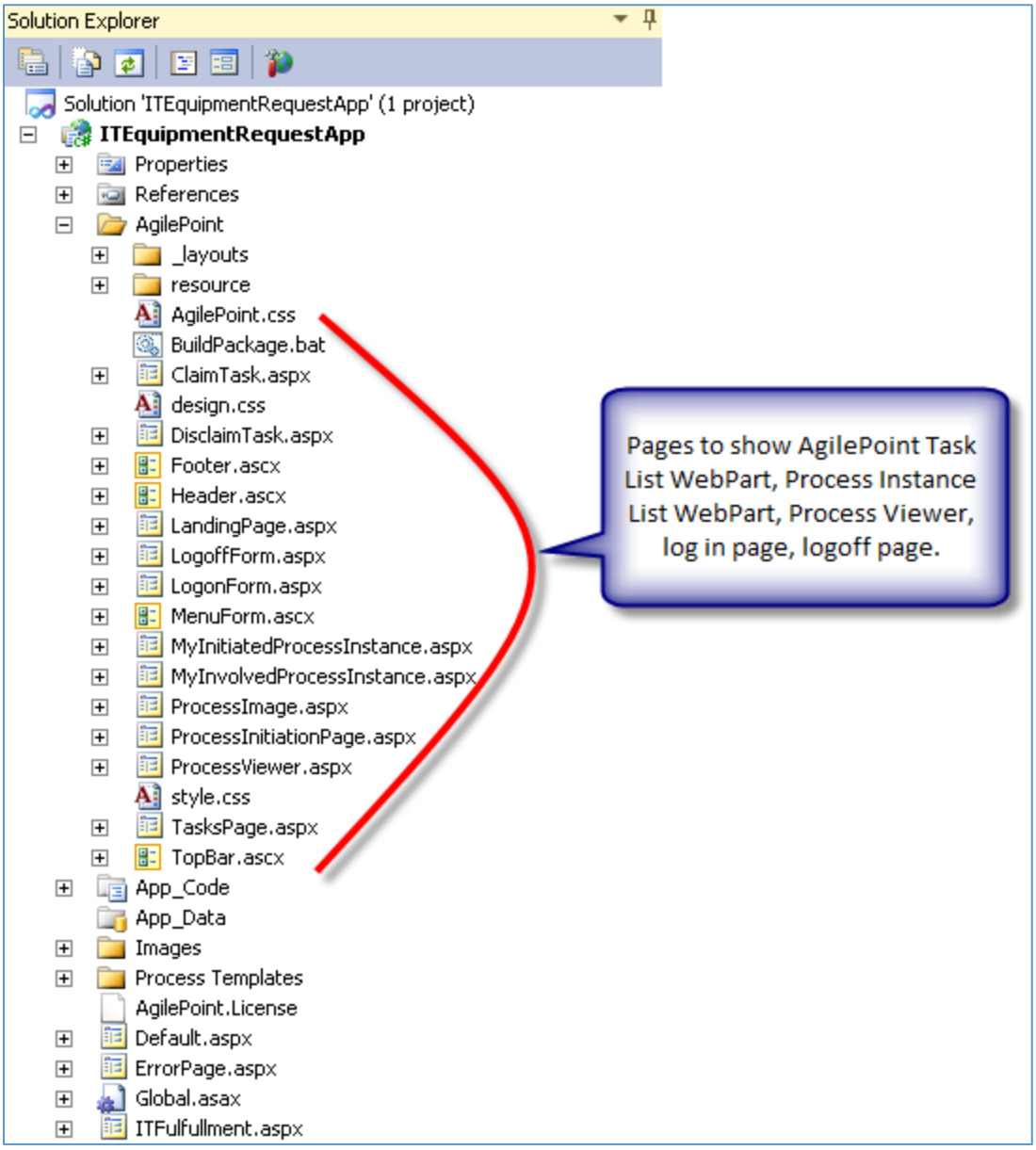
4. AgilePoint BPM Web Application Wizard starts and asks for Process Template file, which is nothing but the Visio process file. Browse to "**ITRequest.vsd**" file created following the "**Process Designer**" guide by clicking on "Browse" button. The check box "Add WebPart to the AgilePoint Web Application" is already selected and disabled since for Azure enabled applications AgilePoint generates the newer Task List WebPart version instead of making the older grid based version available as an option. Do not select other check box to "Add Mobile Pages to the AgilePoint Web Application" as we are not covering that part in this guide. Click "Finish" button. It takes a minute (depending on how complex the process is) to generate the web site solution.

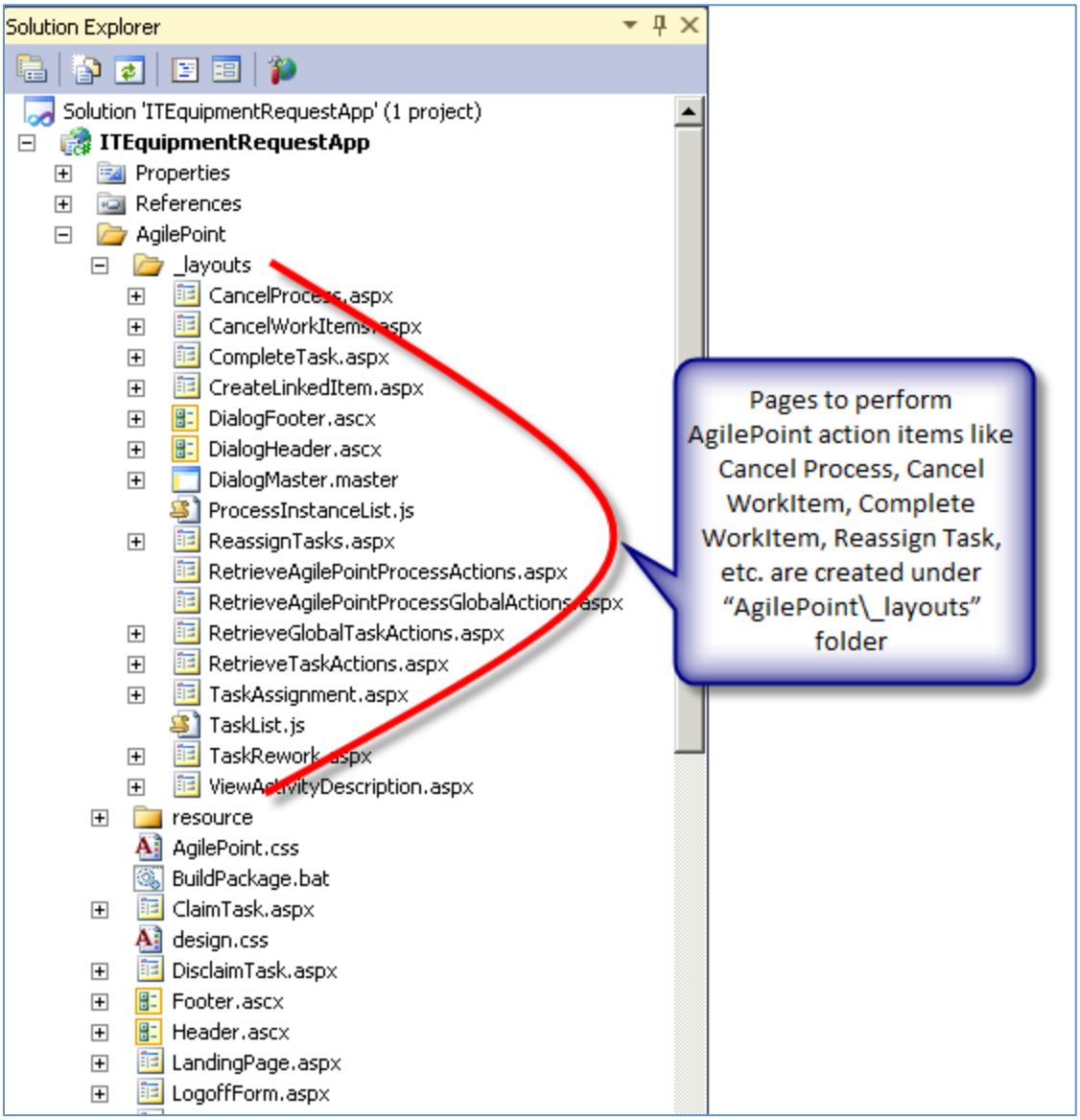


Picture: AgilePoint BPM Web Application Wizard

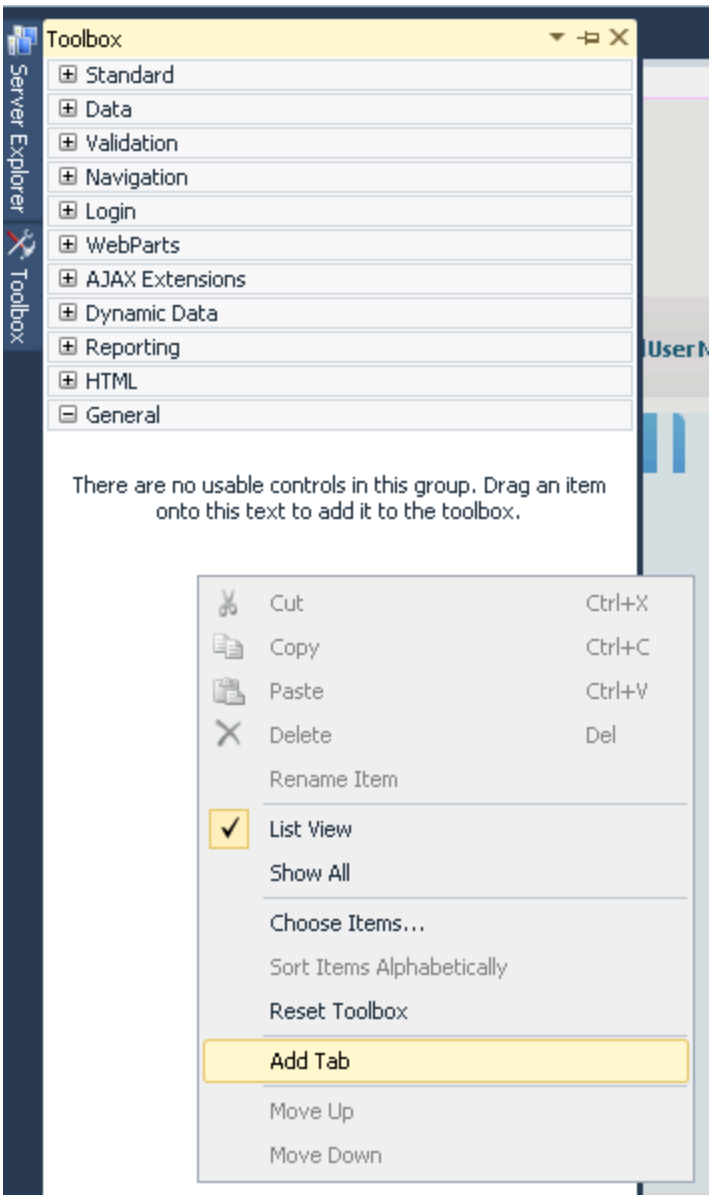
5. Let's have a closer look at Solution explorer and see what things are created automatically:
   1. ASPX files for all manual shapes in the process template. The name of the ASPX file is the value of "WorkToPerform" of the manual shape. E.g. SubmitRequest.aspx, ManagerApproval.aspx, and ITFulfillment.aspx. These would act as a UI for performing the manual tasks in the workflow. These are the pages where you would add your process specific code. The rest of the pages are general process administration pages and already have the required code.
   2. Pages to show AgilePoint Task List WebPart, Process Instance List WebPart, Process Viewer, log on page, log off page, etc. are created under "AgilePoint" folder.
   3. Pages to perform AgilePoint action items like Cancel Process, Cancel WorkItem, Complete WorkItem, Reassign Task, etc. are created under "AgilePoint_layouts" folder.
   4. Required resource files for multiple languages and image files.



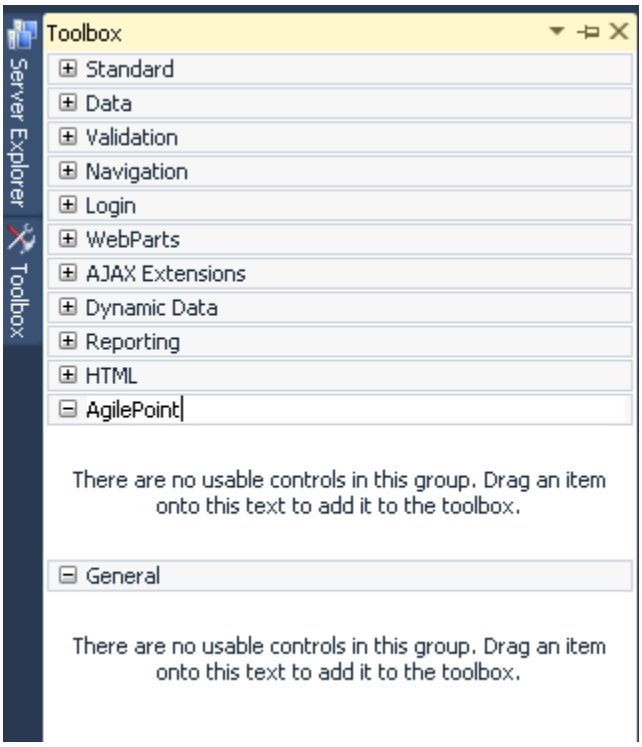




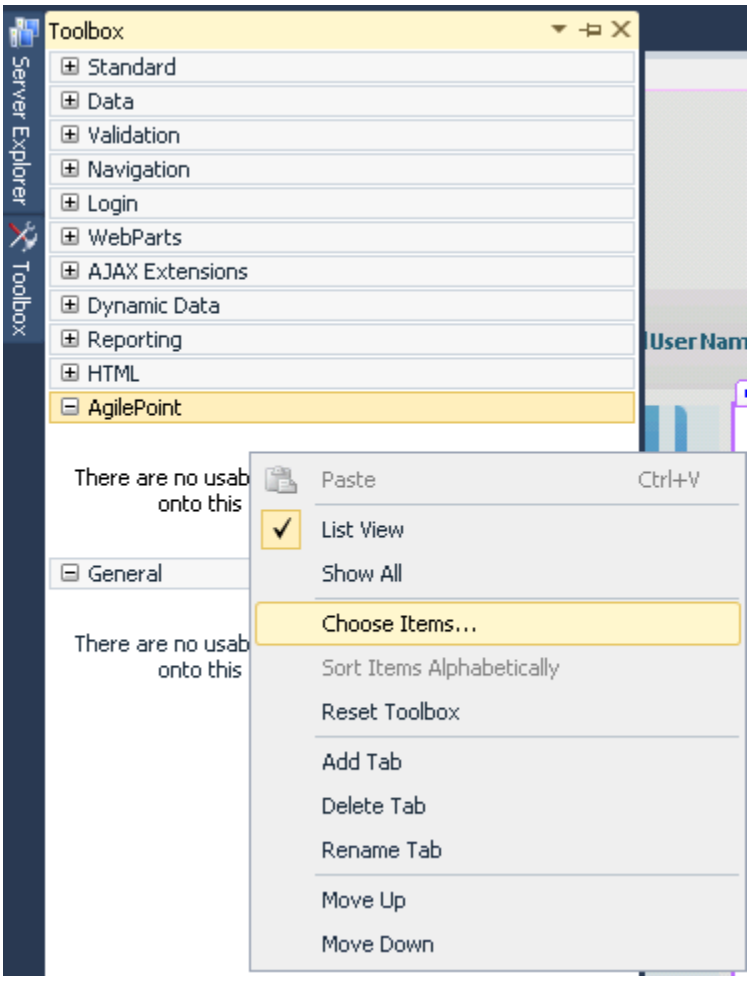
6. Since this is the first time you are using AgilePoint Website template in Visual studio, let's add AgilePoint controls on the Visual Studio toolbox. AgilePoint provides some very useful ASP.Net controls with data binding feature which can help you design your ASP.Net applications quickly. However you can also decide to use any third-party controls like any other regular ASP.Net application. However for this exercise we would use AgilePoint controls. Open Toolbox, right click and select "**Add Tab**" option.



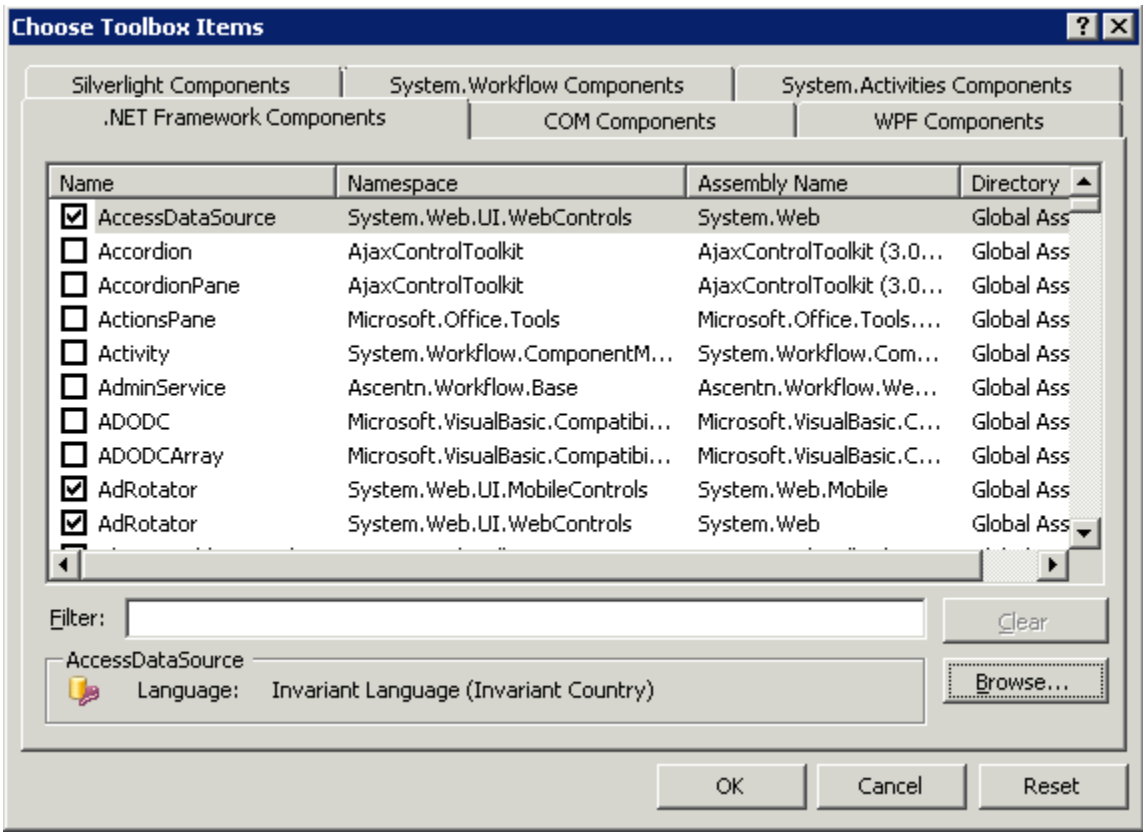
7. Name the tab as "**AgilePoint**".



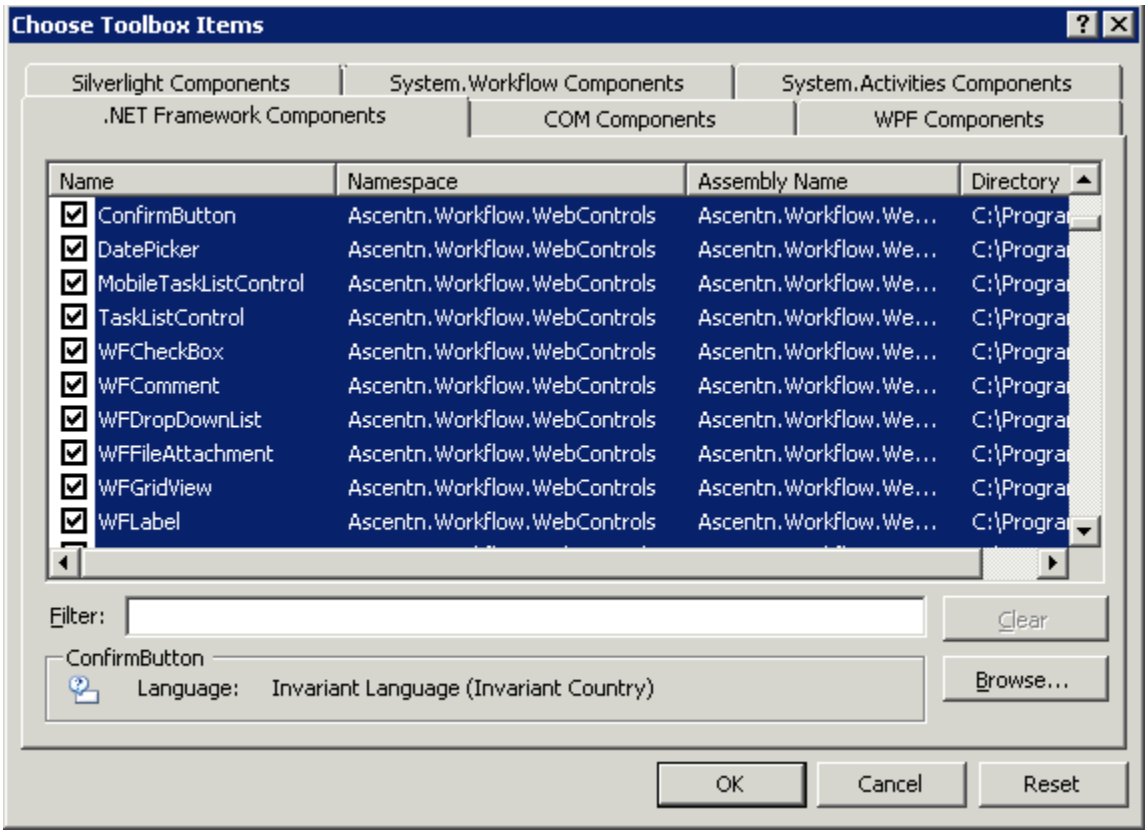
8. Right-click inside “AgilePoint” toolbox blank area and select “Choose Items…”

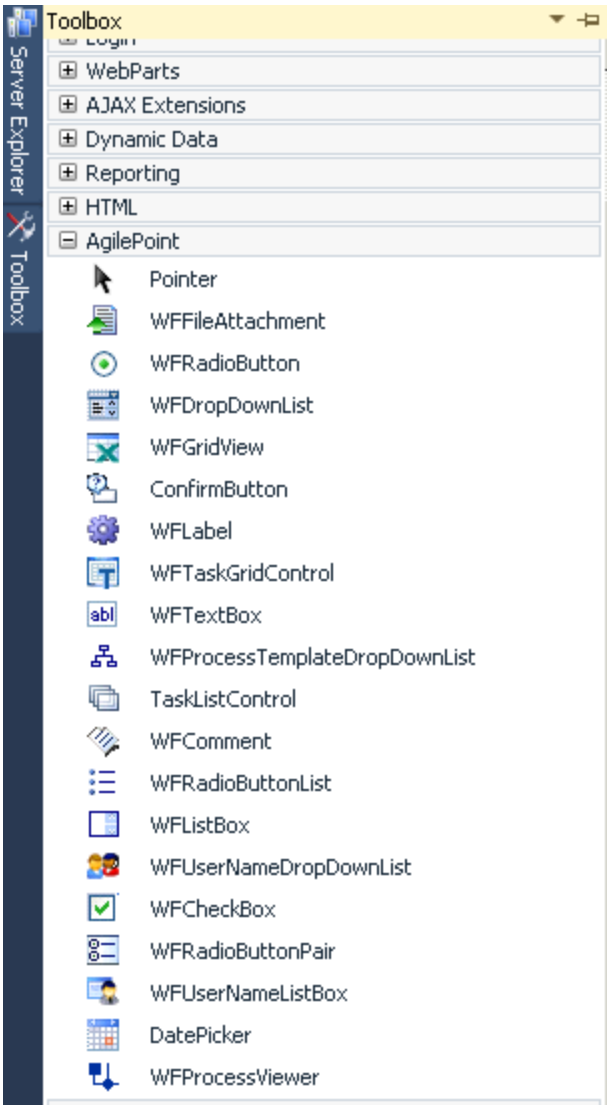


9. Click the Browse button, go to folder "**C:\Program Files (x86)\AgilePoint\AgilePoint Developer\Assembly**" and select the dll "**Ascentn.Workflow.WebControls.dll**", and click OK.



10. Select the check box of all selected items, and click OK to add those controls in AgilePoint toolbox. You can also sort the controls by namespace "**Ascentn.Workflow.WebControls"** and make sure all controls in that namespace are checked before you click OK.



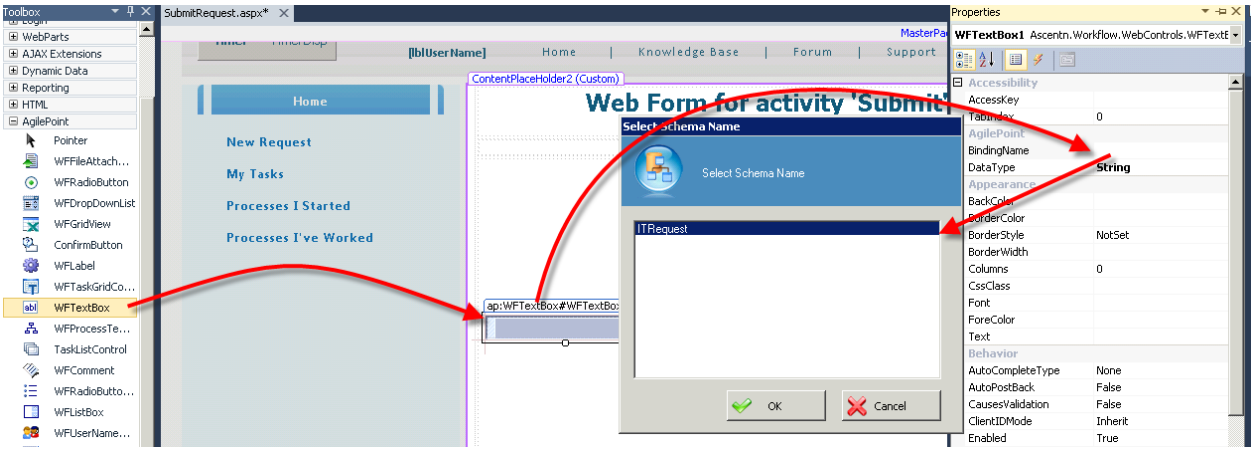


11. Now you are ready to design your task specific pages.

12. Right-click **Default.aspx** and click on "**Set as Start Page**".

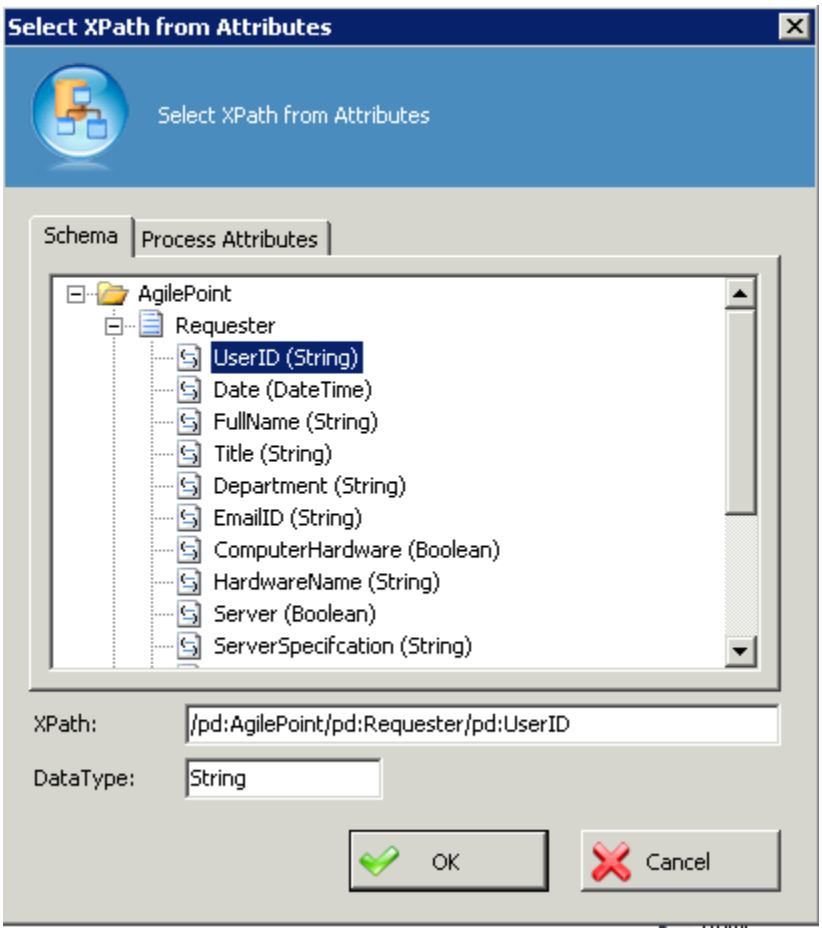
13. Open **SubmitRequest.aspx** page, and click on "**Design**" view at left bottom corner of the aspx page to open the design view of the page.

14. Drag and drop **WFTextbox** control from the AgilePoint toolbox on the page design area. Go to properties of the control and select **BindingName** property. Select the process template, "**ITRequest**" in your case when prompted to select the schema.



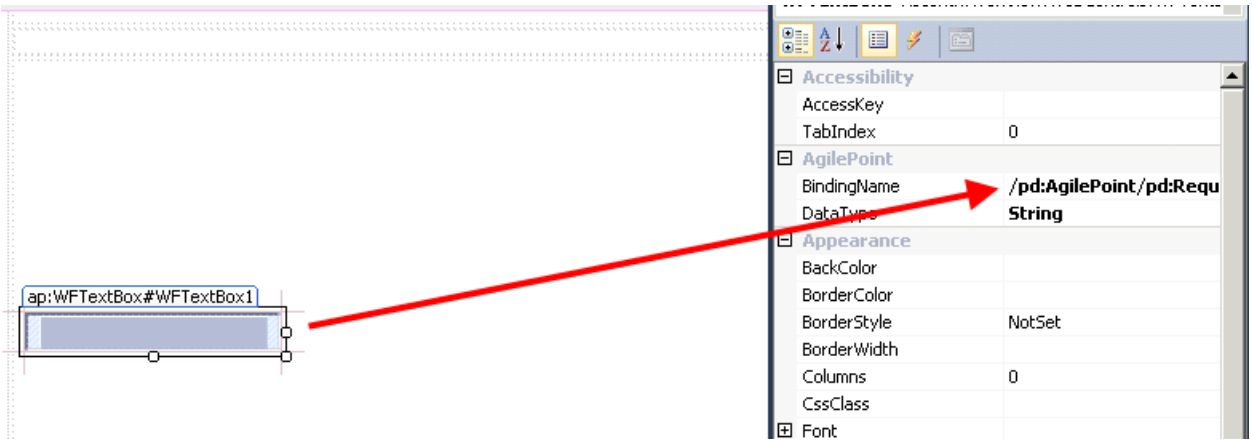
Picture: UserID textbox binding with process model

15. Select the **UserID** element from the XPath wizard.



16. This maps newly added text box to UserID element of process model schema and any value you enter in the textbox would automatically get persisted in the workflow database. BindingName is the only property different than the regular ASP.Net Textbox control and that property makes integration/binding seamless without writing any code.

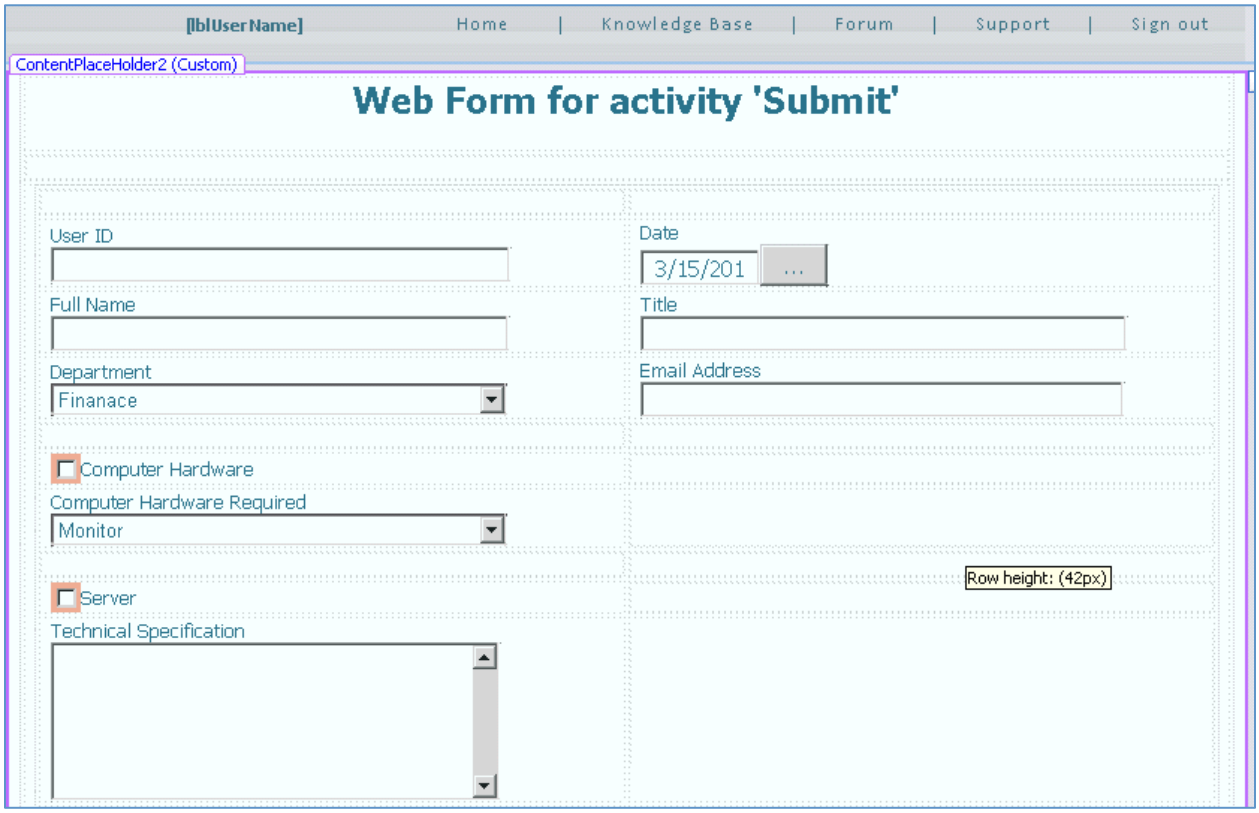
Note: Leverage HTML design experience to design the page as per the requirement.

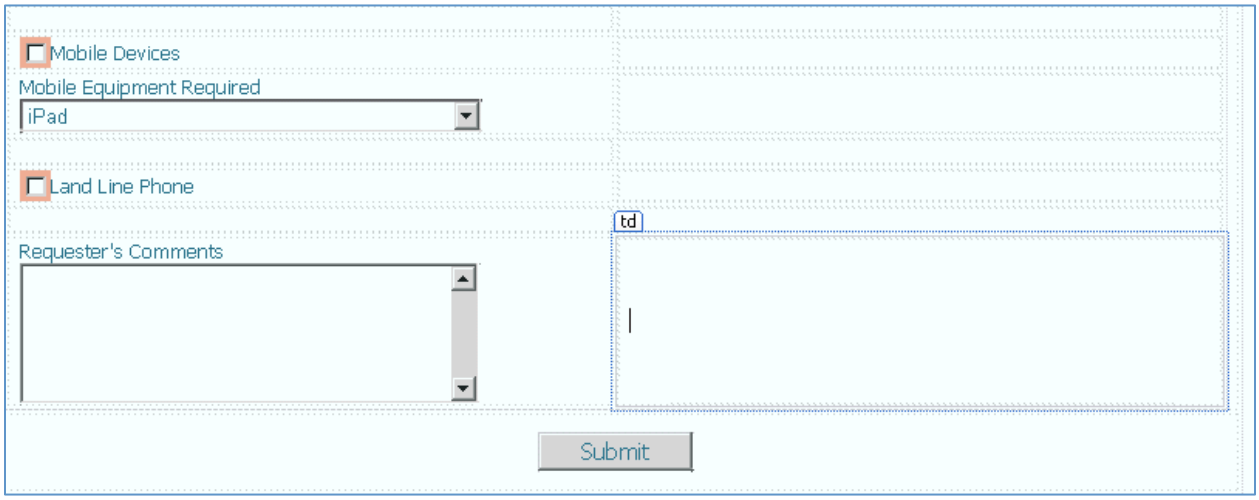


17. The following table shows the details of what other control you need to use and what it the corresponding BindingName property value. The extra properties required to configure the control are shown in Extra Properties column. Design the page by dragging and dropping these controls on design surface and set the corresponding properties as shown below. Optionally you can create a table structure in the SubmitRequest.aspx file and drag and drop the control and set the BindingName and other required properties for better layout of the screen.

Sr. No.	Label	Control	Binding Name	Extra Property
1	User ID	WFTextBox	/pd:AgilePoint/pd:Requester/pd:UserID	
2	Date	DatePicker	/pd:AgilePoint/pd:Requester/pd:Date	
3	Full Name	WFTextBox	/pd:AgilePoint/pd:Requester/pd:FullName	
4	Title	WFTextBox	/pd:AgilePoint/pd:Requester/pd:Title	
5	Department	WFDropDownList	/pd:AgilePoint/pd:Requester/pd:Department	Items: Finance, HR, Marketing and Sales
6	Email Address	WFTextBox	/pd:AgilePoint/pd:Requester/pd:EmailID	
7	Computer Hardware	WFCheckBox	/pd:AgilePoint/pd:Requester/pd:ComputerHardware	
8	Computer Hardware Required	WFDropDownList	/pd:AgilePoint/pd:Requester/pd:HardwareName	Items: Monitor, Laptop, Desktop and Projector
9	Server	WFCheckBox	/pd:AgilePoint/pd:Requester/pd:Server	
10	Technical Specification	WFTextBox	/pd:AgilePoint/pd:Requester/pd:ServerSpecifcation	TextMode : MultiLine
11	Mobile Device	WFCheckBox	/pd:AgilePoint/pd:Requester/pd:MobileDevice	
12	Mobile Equipment Required	WFDropDownList	/pd:AgilePoint/pd:Requester/pd:DeviceName	Items: iPad, Cell Phone
13	Land Line Phone	WFCheckBox	/pd:AgilePoint/pd:Requester/pd:LandlinePhone	
14	Requester's Comments	WFTextBox	/pd:AgilePoint/pd:Requester/pd:RequesterComment	TextMode : MultiLine

18. After you configure the page, it will look like as shown below. Optionally you can add images, title and icons. These are regular ASP.Net features, and are not covered here.





19. Once the design part is done, turn to code behind file. The Page_Load event method has some autogenerated code provided by AgilePoint component to save coding effort. Since SubmitRequest is the first page, process initialization needs to be done from the code of this page when user clicks Submit button. Change the code of SubmitRequest class as shown below. As you can see from code below, you just have to add a couple of extra lines of code and any UI

logic like validation or conditionally enabling/disabling controls but rest of the data persistence logic is automatically handled for you using the data binding feature available in AgilePoint's ASP.Net framework.

C# Code

```
public partial class SubmitRequest : Ascentn.Workflow.WebControls.WFWorkSheetPage
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!string.IsNullOrEmpty(Request.QueryString["WID"]))
              {
                     WFManualWorkItem workItem;
                     try
                     {
                            workItem =
GetAPI(Session).GetWorkItem(Request.QueryString["WID"]);
                     }
                     catch (Exception excp)
                     {
                            throw excp;
                     }
                     if (workItem != null &&
!workItem.UserID.Equals(HttpContext.Current.User.Identity.Name,
StringComparison.InvariantCultureIgnoreCase))
                     {
                            Response.Write("<script language='javascript'>alert('This task
is not assigned to you. Click OK to redirect to Tasks page.');
window.location.href='AgilePoint/TasksPage.aspx';</script>");
                     }
              }
       }

       protected void btnSubmit_Click(object sender, System.EventArgs e)
       {
        if (string.IsNullOrEmpty(Request.QueryString["WID"]))
        {
            string processModel =
System.Configuration.ConfigurationManager.AppSettings["ProcessModel"];
            base.CreateProcInst(processModel, processModel + "-" + UUID.GetID());
            Response.Redirect("AgilePoint/MyInitiatedProcessInstance.aspx", true);
        }
        else
        {
            base.CompleteWorkItem();
            Response.Redirect("AgilePoint/TasksPage.aspx", true);
        }
       }
}
```

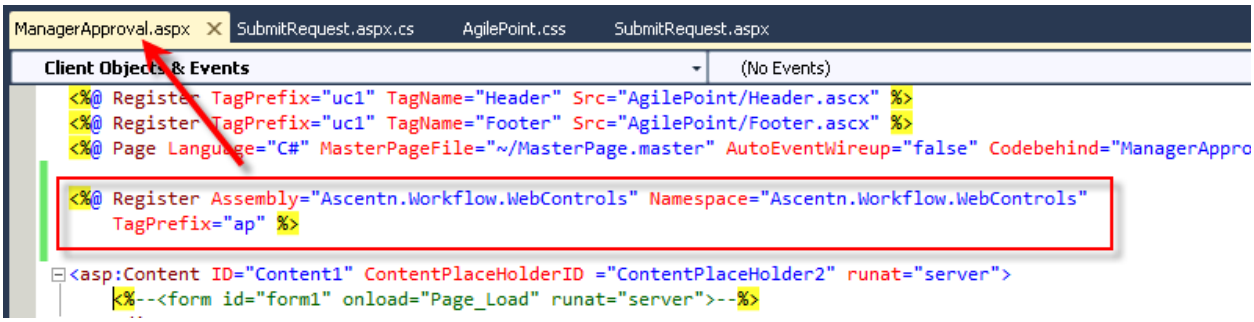
Code: Page_Load and btnSubmit_Click event methods on SubmitRequest web page

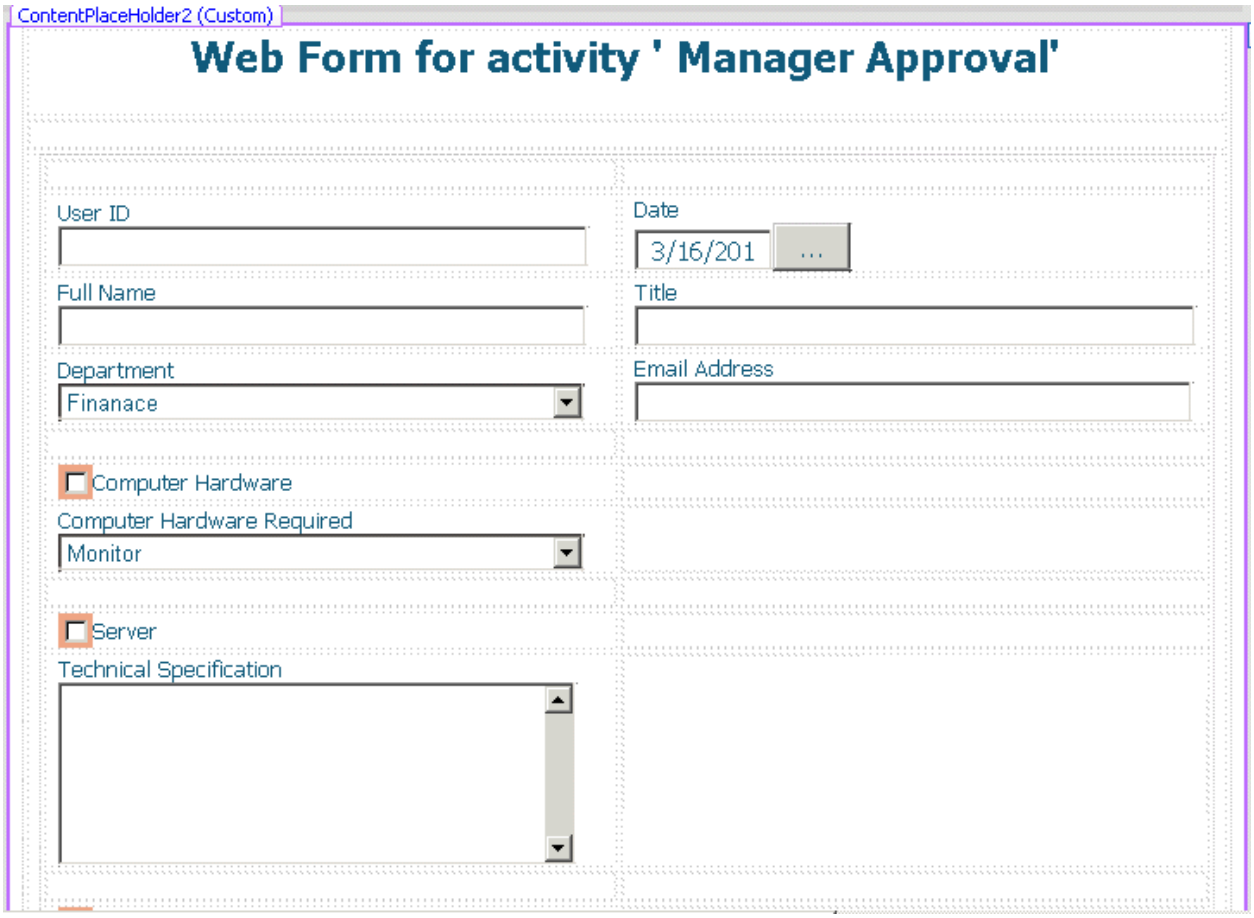
20. Next Go to ManagerApproval.aspx page. We can reuse the design of SubmitRequest.aspx page to design ManagerApproval.aspx page. Copy the contents from SubmitRequest.aspx and paste it in ManagerApproval.aspx as show in images below.

21. Switch to the Design view of ManagerApproval.aspx page the page will look like this:

22. This is because missing Assembly directive for AgilePoint controls in ManagerApproval.aspx page. Add this line at the top in ManagerApproval.aspx page:

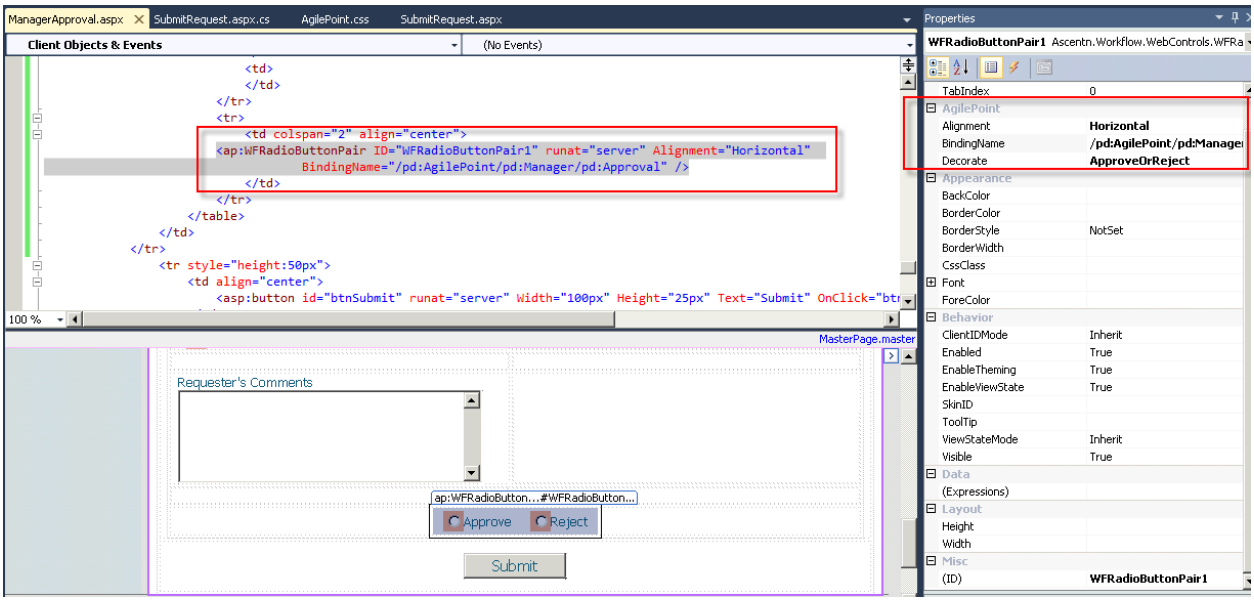
```
<%@ Register Assembly="Ascentn.Workflow.WebControls"
Namespace="Ascentn.Workflow.WebControls" TagPrefix="ap" %>
```





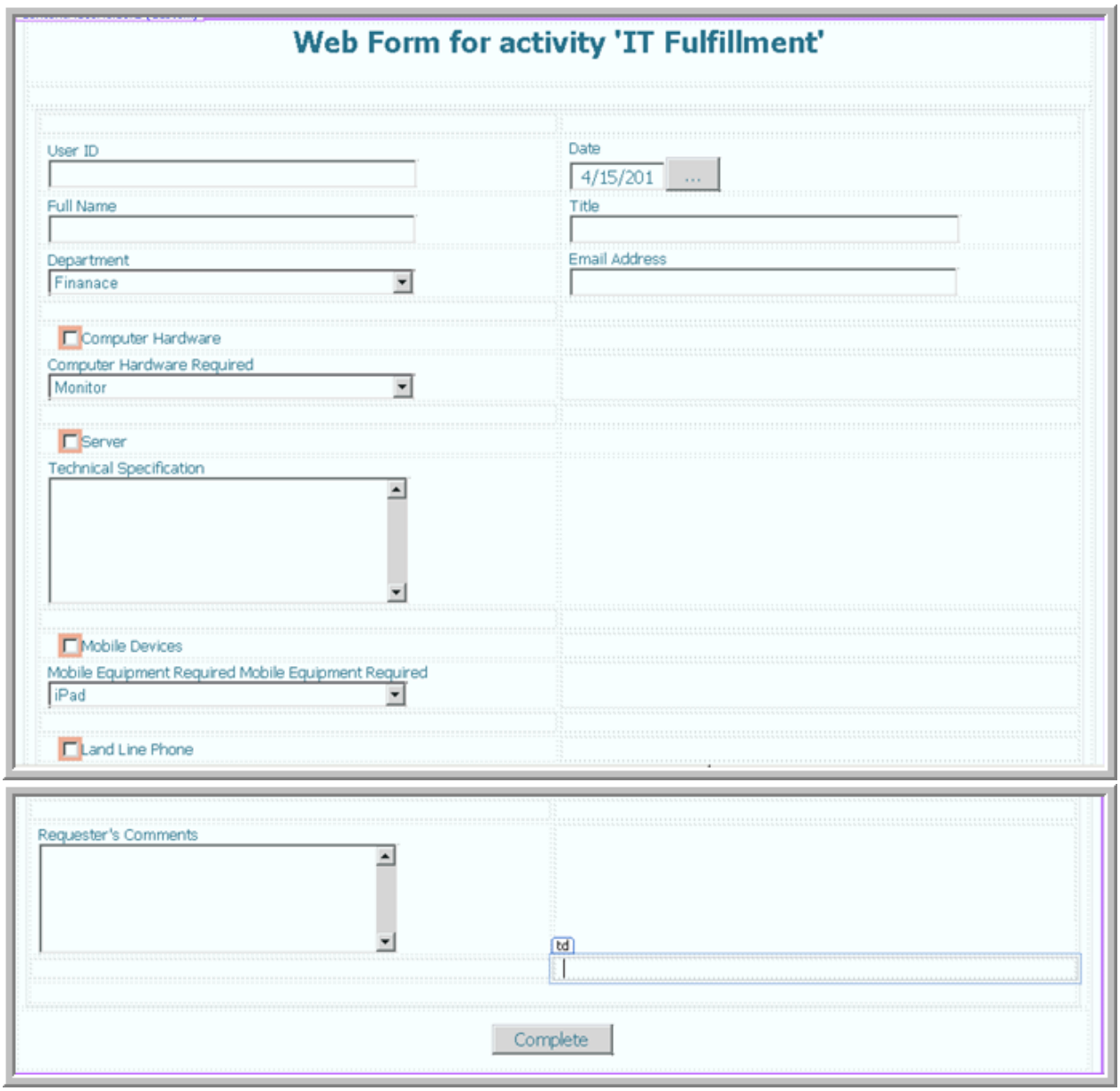
23. Make all the controls read only by setting either the ReadOnly property or Enabled property to False by right clicking on individual controls and going to their property panel.

24. Next we will add new controls required to perform work on manager approval screen. Add WFRadioButtonPair control and bind it to "/pd:AgilePoint/pd:Manager/pd:Approval" element from the schema. The page should look very similar to SubmitRequest page except additional approve/reject radio button.

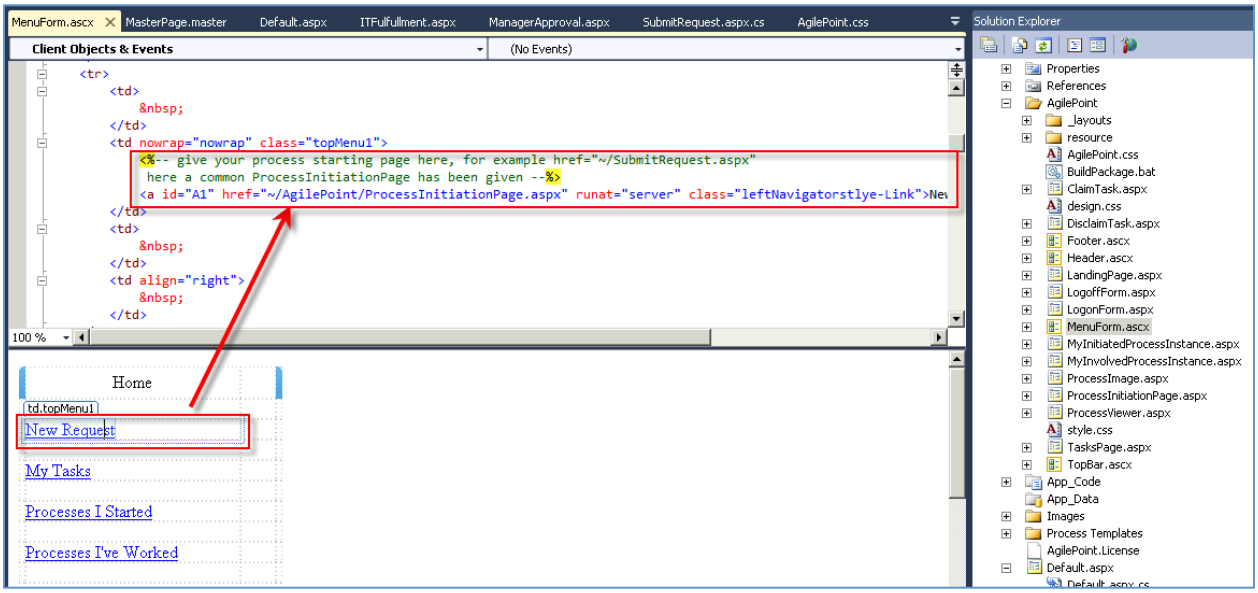


Note: There is no need to add or change any code in the code behind file since required code is already generated for you, unless you wanted to add any UI logic for formatting. As you can see the AgilePoint framework automatically handles the entire logic of loading the controls with values entered on the submit screen and also persists back any changes made on this page, back to the workflow database. This is possible because of the data binding feature. This data can be optionally captured in your custom database for reporting purpose without writing any code using AgilePoint data services feature. However in this simple exercise we will not use that feature.

25. Repeat the same steps to design ITFulfillment.aspx page and change the text on Submit button to “Complete”.



26. Open AgilePoint\MenuForm.ascx. In a "Split View" of the page, select the link "New Request" and modify the value of the href attribute as per the comment above the control to **"~/SubmitRequest.aspx**" which is the page to create new request. Also change the text to "**New IT Request**".

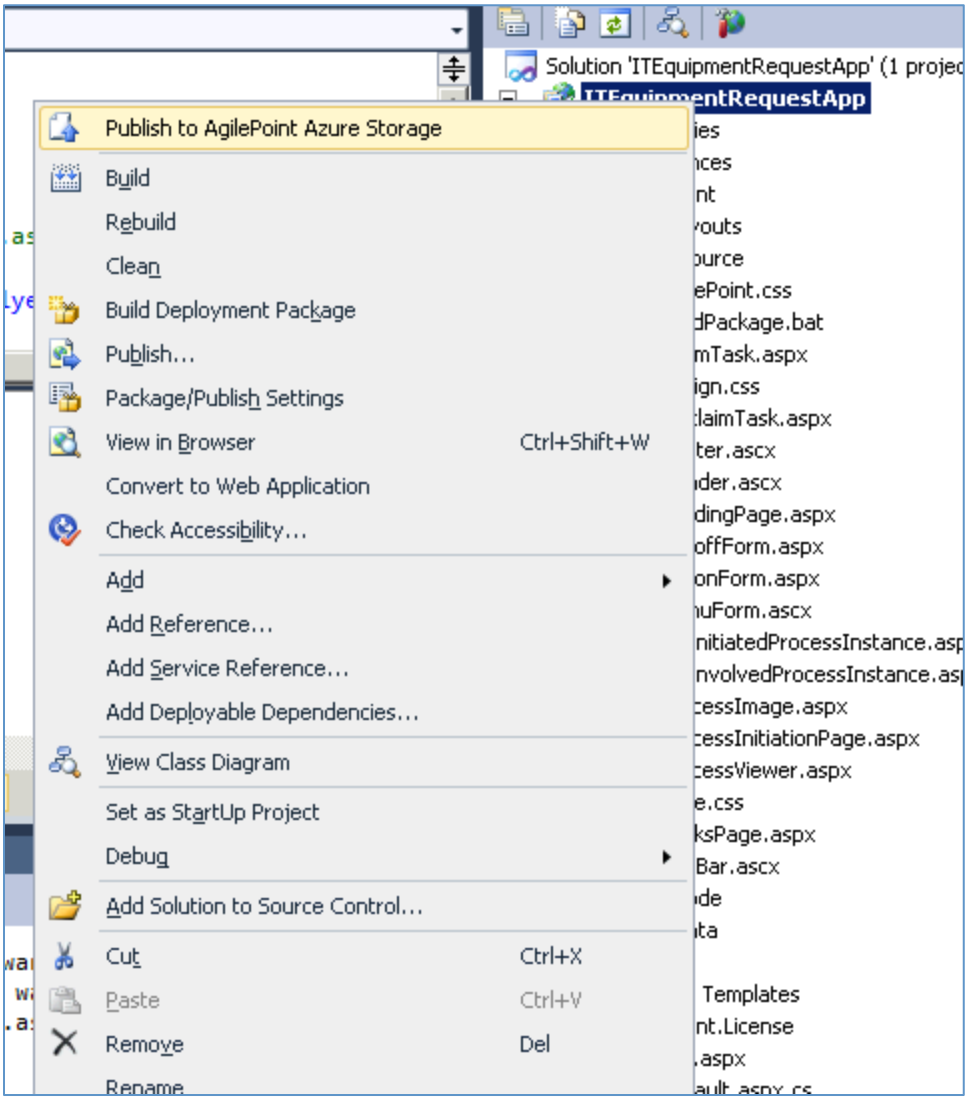


27. Build the web application.

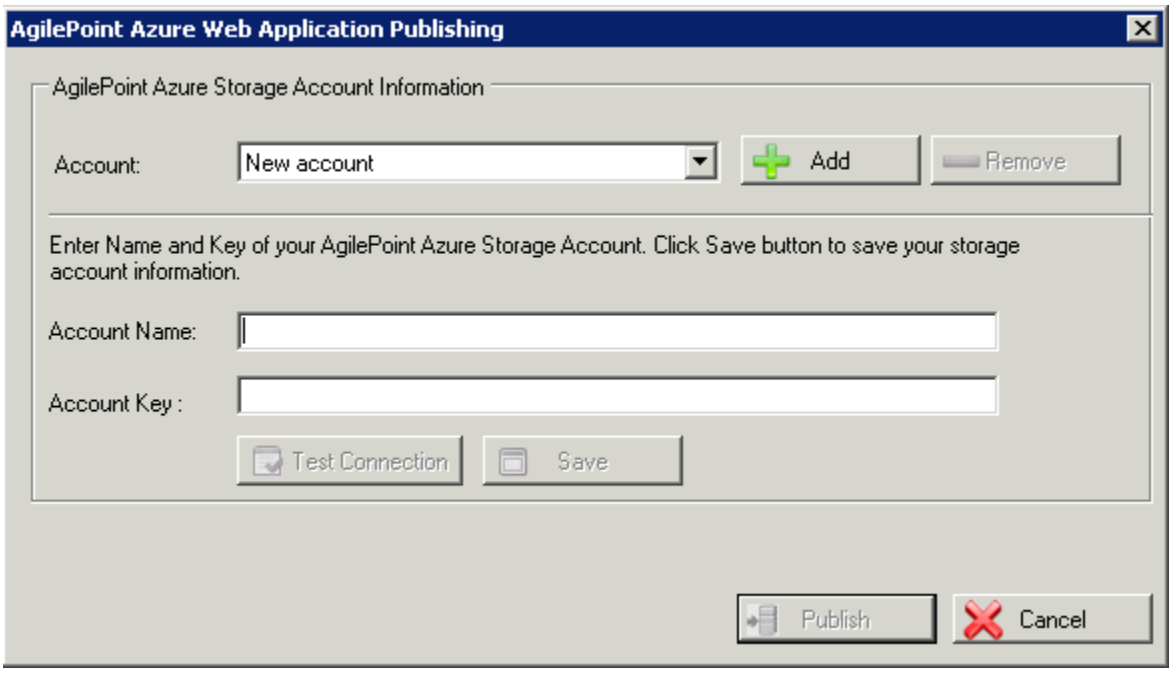
Publishing the Web Application to Azure Storage

If build operation is successful, you are ready to publish the web application to your Azure environment. Following are the steps to publish the web application.

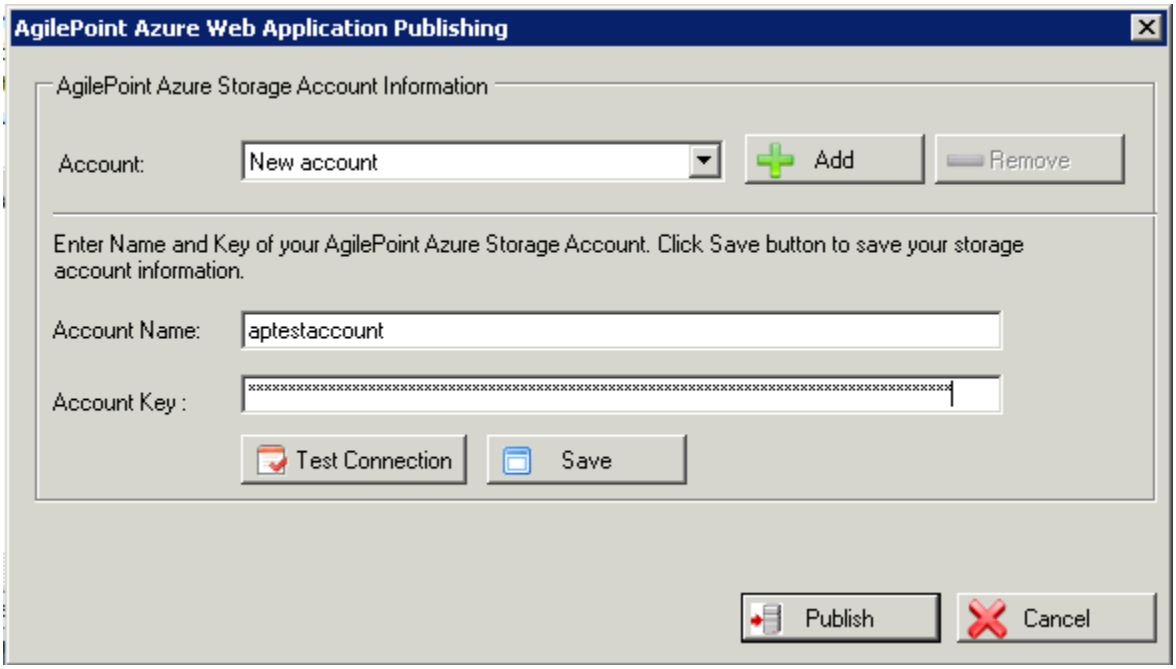
1. Right click on project in solution explorer and select “Publish to AgilePoint Azure Storage”



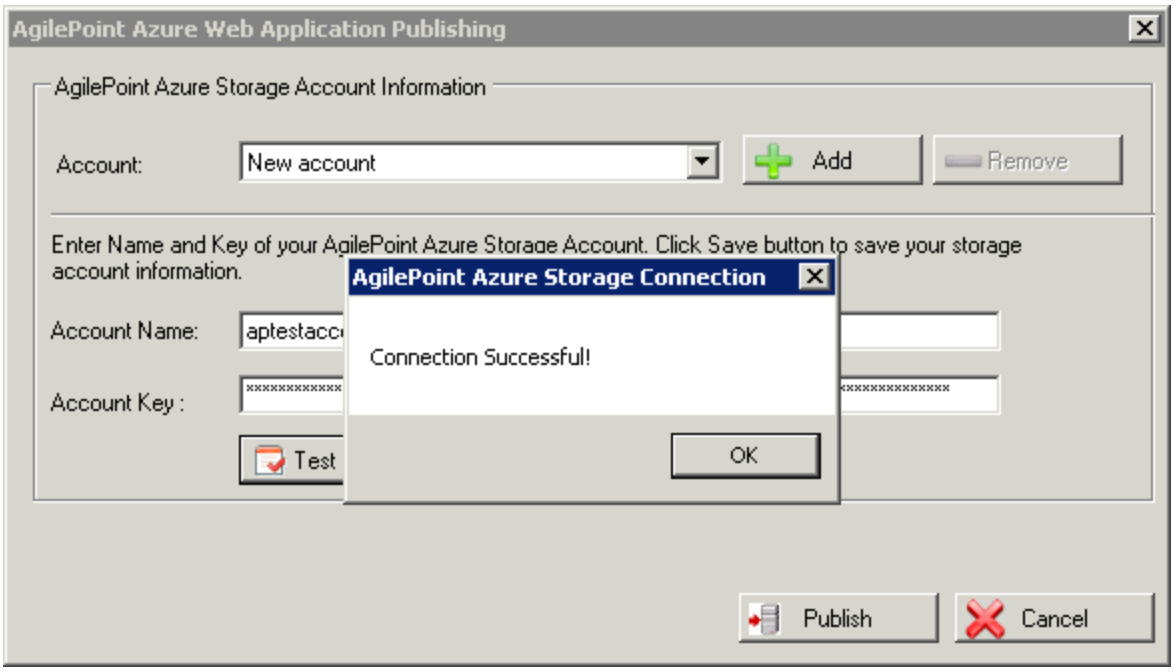
2. It will open up Web Application Publishing dialog.



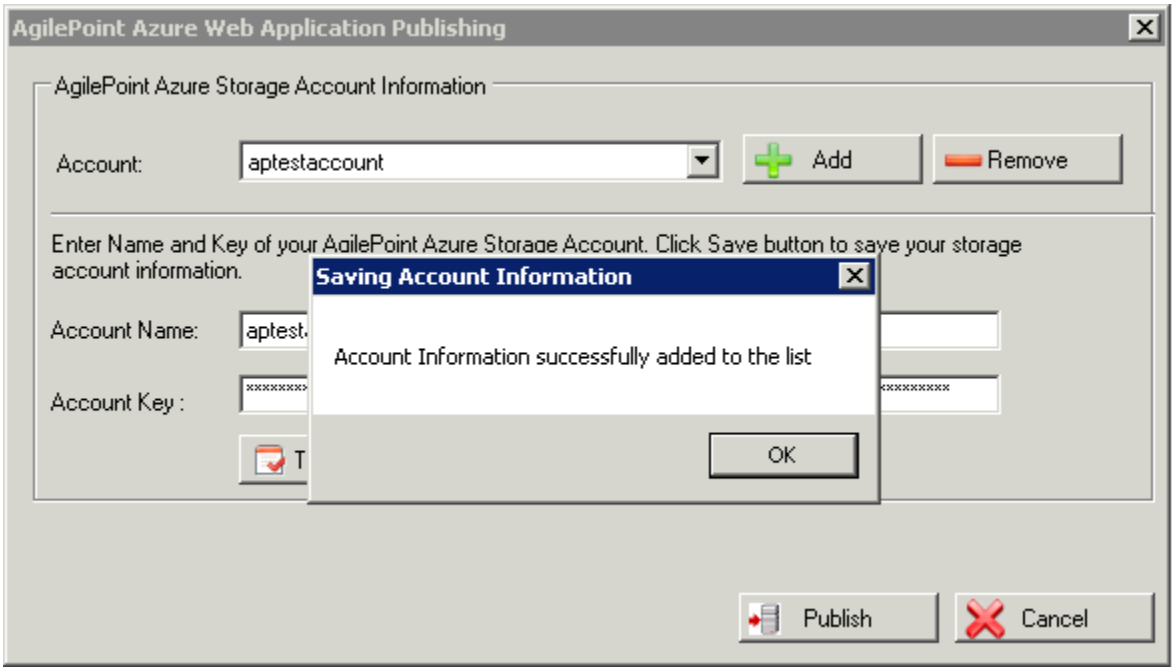
3. Enter your Azure Storage account's Name and Primary Key in the text boxes. This is an Azure Storage account which was used while provisioning your AgilePoint Server in Azure. Once Name and Key are entered "**Test Connection**" and "**Save**" buttons get enabled.



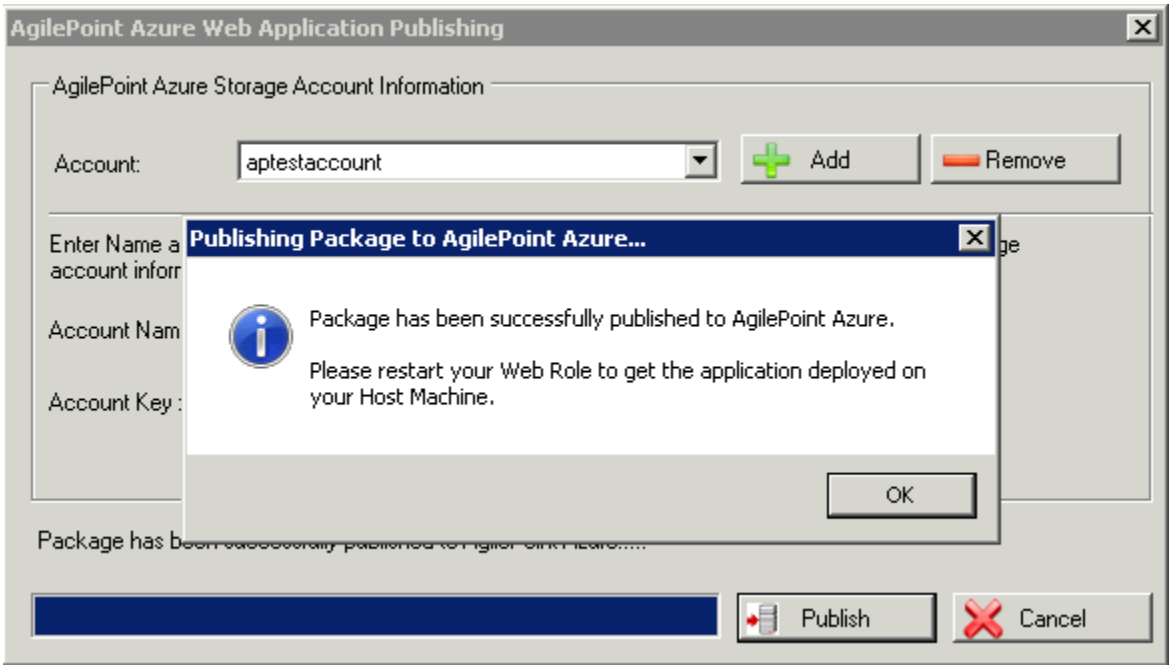
4. Click on Test Connection button to verify Storage account is accessible from this machine.



5. Save the Storage Account information for future use by clicking on save button. Next time when you want to publish your application then you can just select the Storage account from the "Account" drop down box.

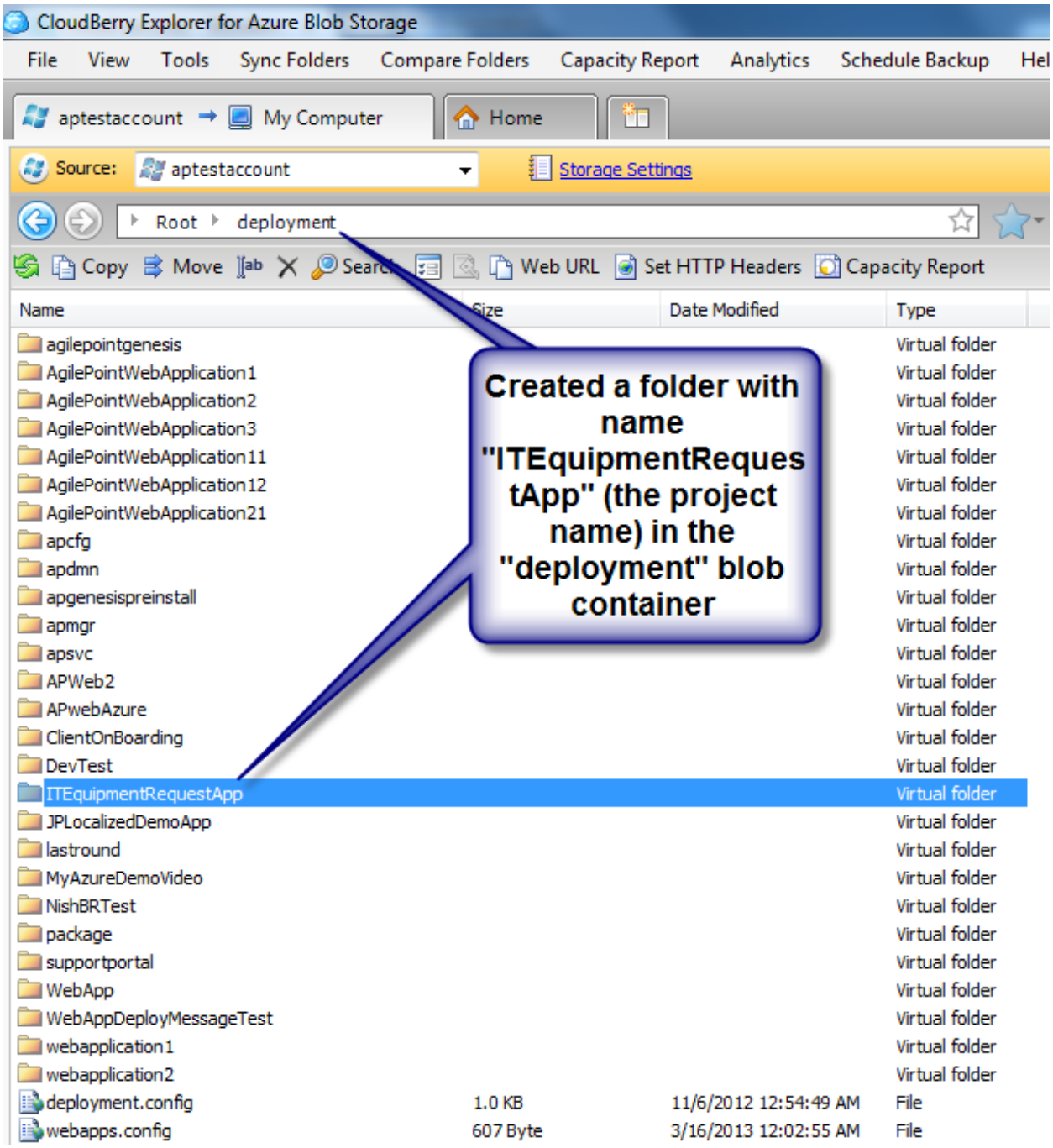


6. Click the Publish button. After the successful deployment of the application to Azure Storage you just need to restart the WebRole in your AgilePoint Server instance on Azure to deploy the application automatically to the cloud service in Azure environment where the WebRole is hosted.

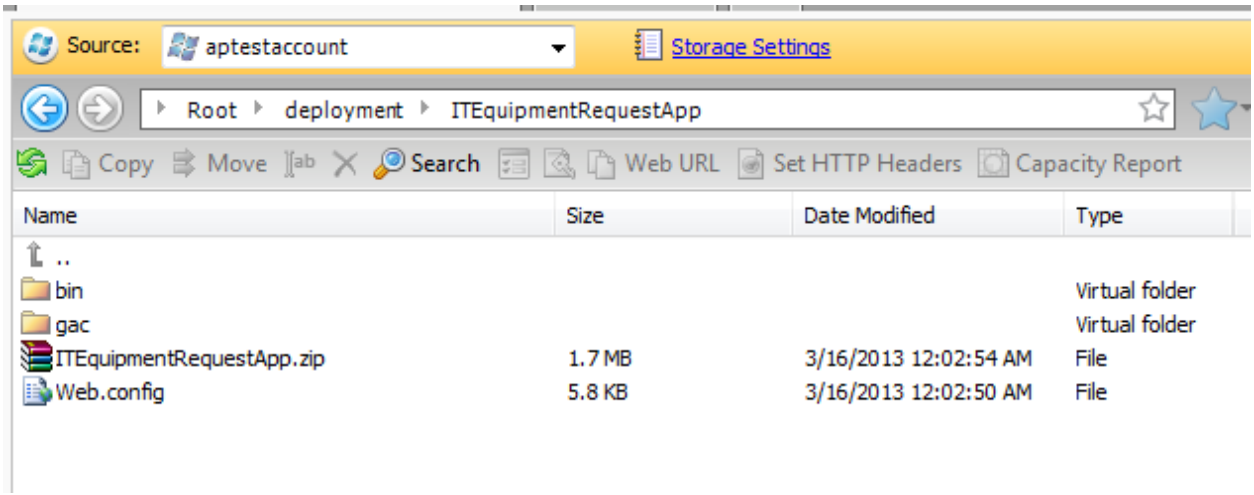


7. Before you restart the WebRole, if you want to know what this deployment has deployed in your Azure Storage then open your Azure Storage in a Storage explorer tool (You can use CloudBerry storage explorer or any explorer of your choice).

   Inside the blob container named "**deployment**" you will find that a folder with name "**ITEquipmentRequestApp**" (the project name) is created.



8. Inside **ITEquipmentRequestApp** folder there are two folders “**bin**” and “**gac**”. Bin folder has the application DLL and GAC folder empty as we don’t need to put any DLL in the GAC for this application. An ITEquipmentRequestApp.zip file which contains all static files for the web application (the aspx files, image files, resource file). And a web.config file.

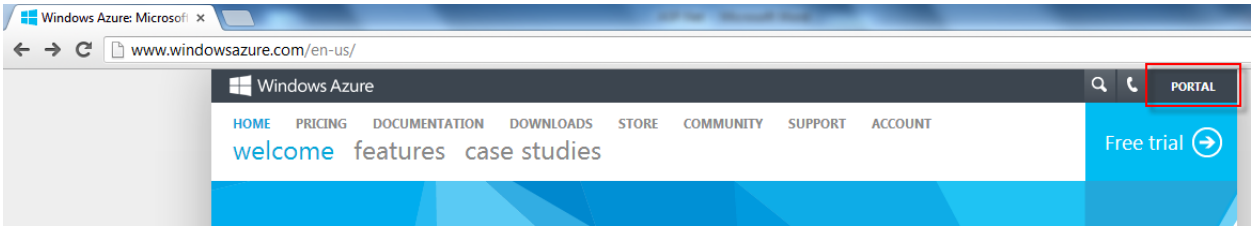


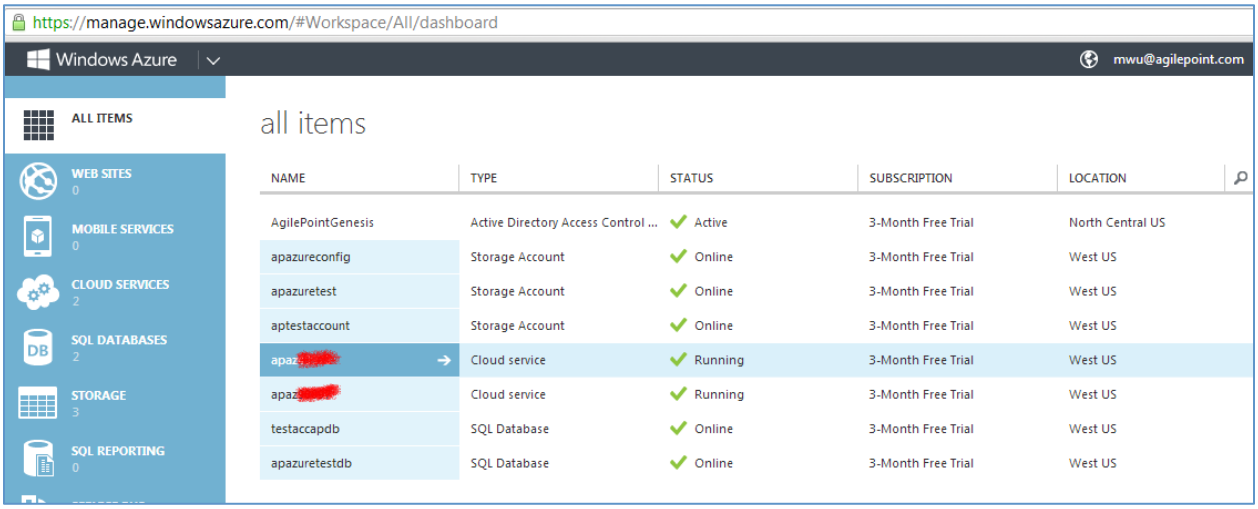
9. Also if you open a file called webapps.config which is at root level in "**deployment**" container, you will see an appsetting entry is created for the web application we just deployed. Here the key will be the Virtual Directory name and the value is the folder name of the application in the storage.

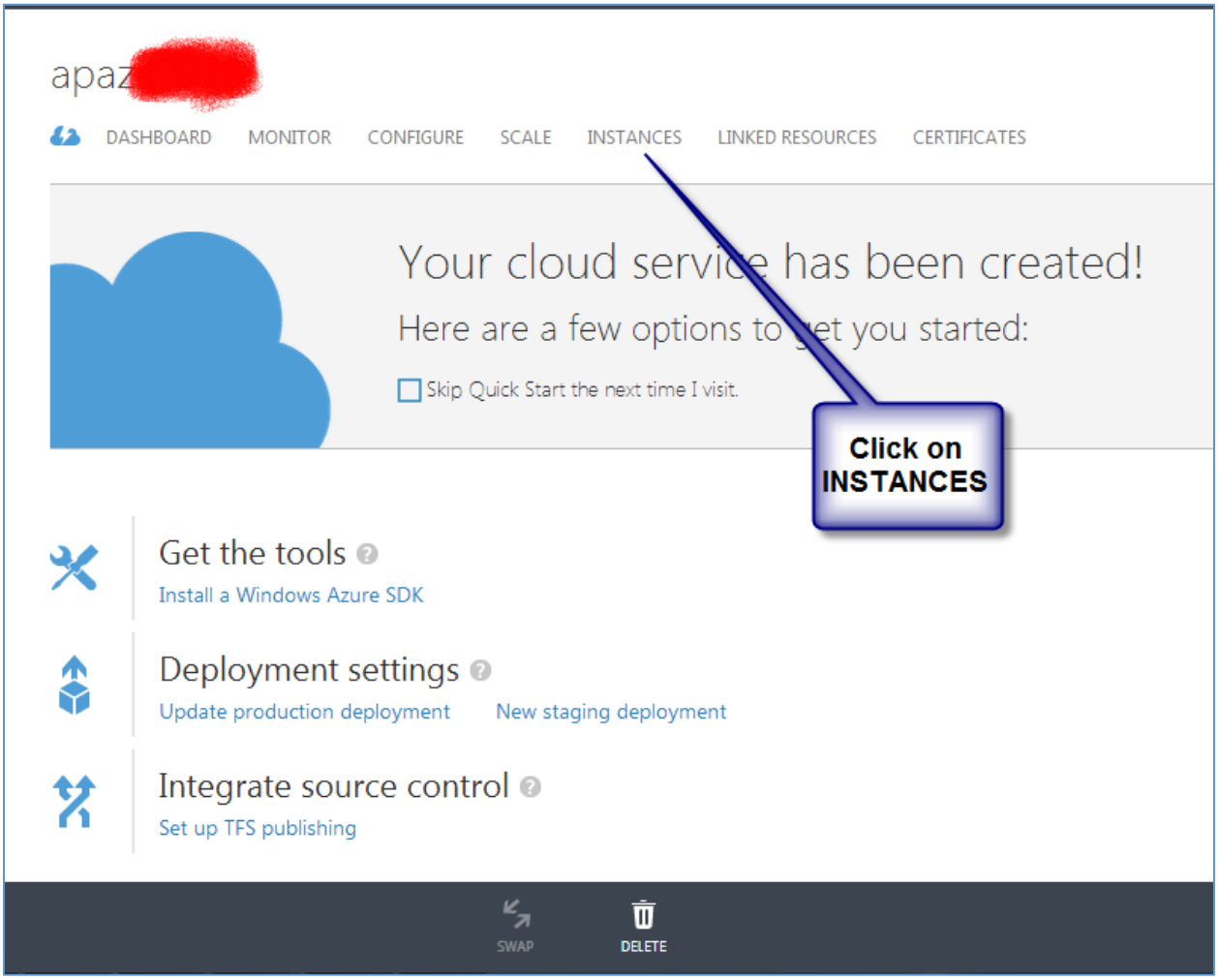
```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <appSettings>
    <add key="ITEquipmentRequestApp" value="ITEquipmentRequestApp" /></appSettings>
  </configuration>
```

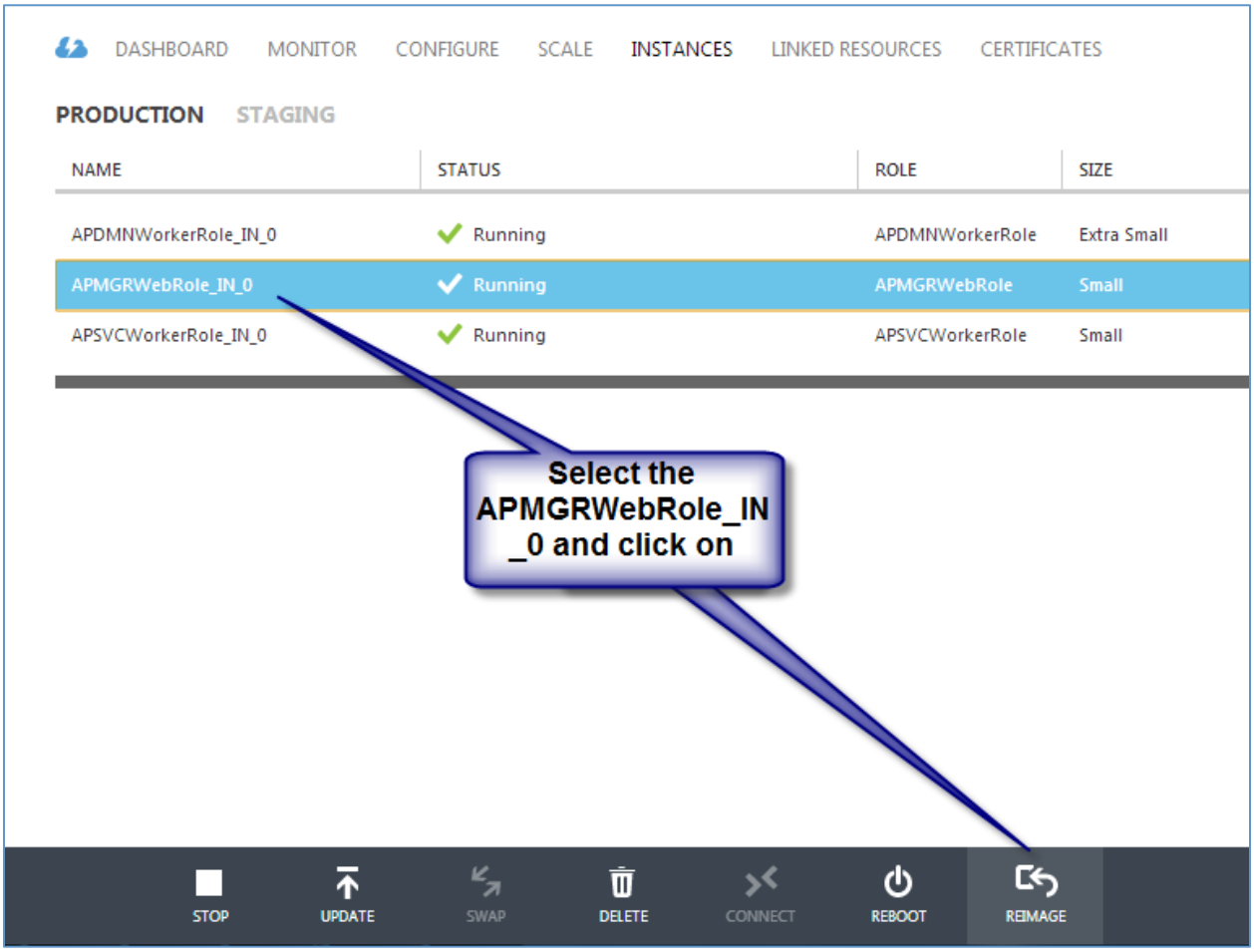
So after deployment your application's URL will be:
Http://<your dns name>.cloudapp.net/ITEquipmentRequestApp/Default.aspx
e.g. http://apaz12345.cloudapp.net/ITEquipmentRequestApp/Default.aspx

10. Next, restart the WebRole. Login to Windows Azure Portal where AgilePoint Server Hosted Service is running.







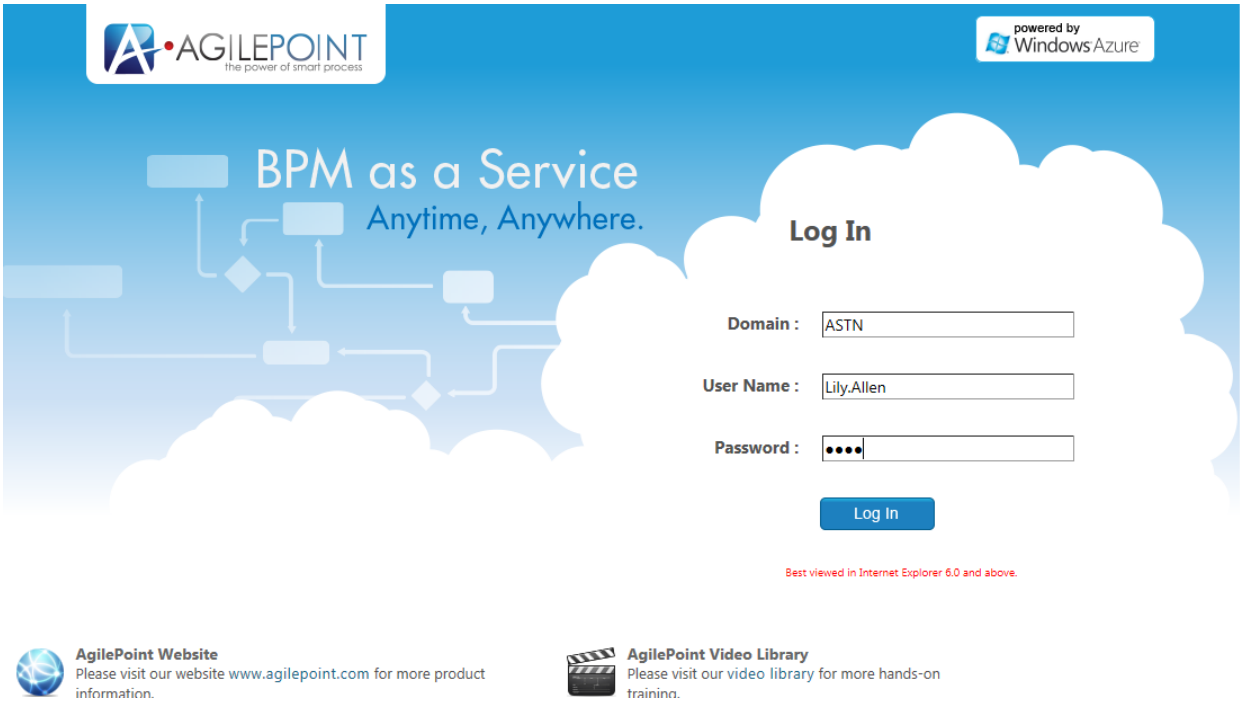


Running a deployed application

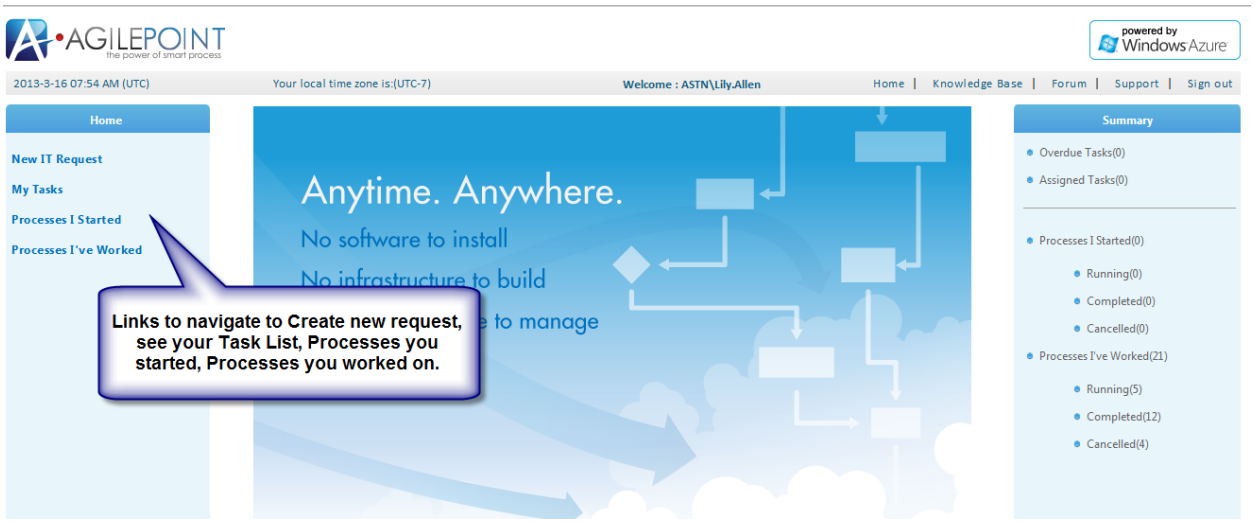
1. Once the Web Role is restarted you can browse your application and start using it right away.

   The application's URL will be:
   Http://<your dns name>.cloudapp.net/ITEquipmentRequestApp/Default.aspx
   e.g. http://apaz12345.cloudapp.net/ITEquipmentRequestApp/Default.aspx

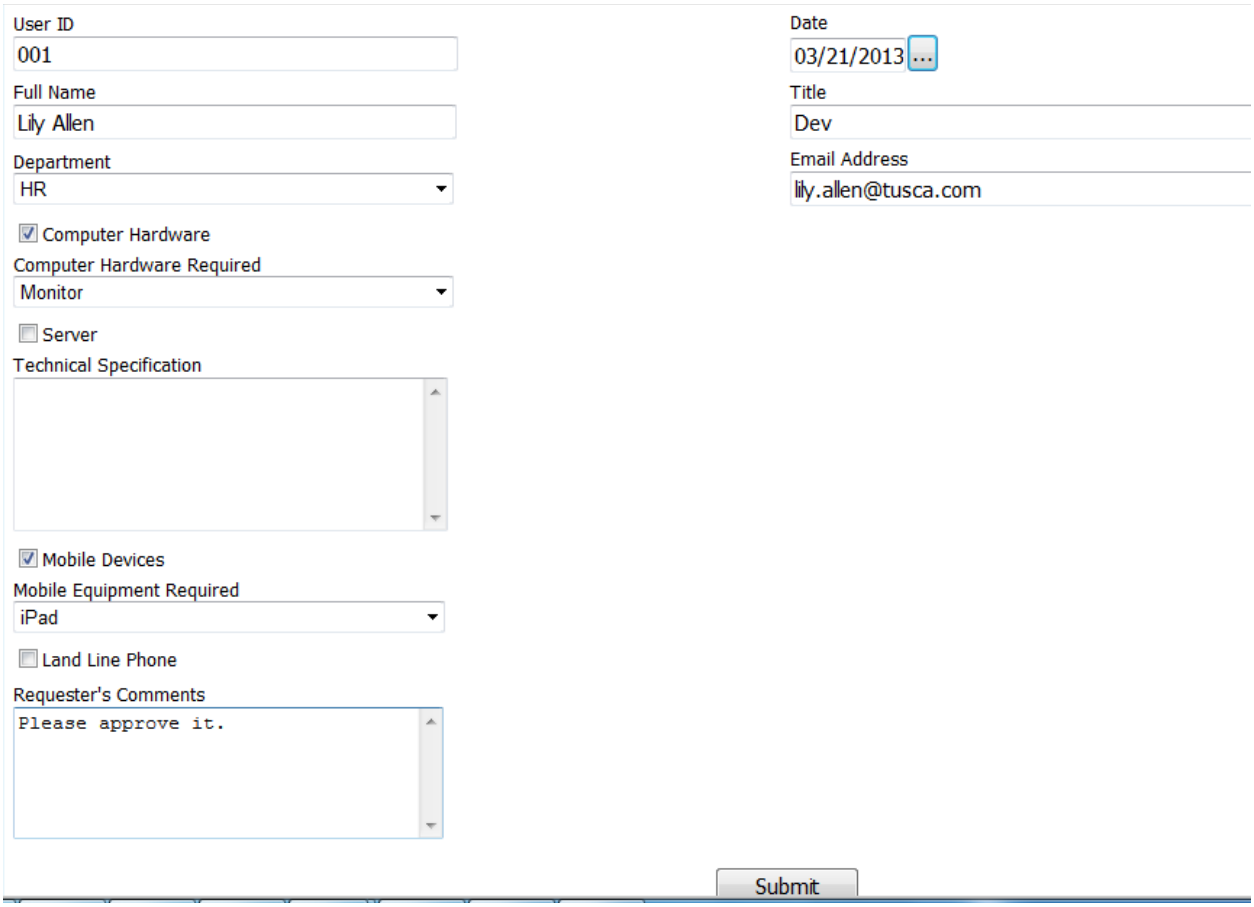
2. The evaluation version of AgilePoint Server on Azure is working against a temporary Domain called ASTN and dummy users are already created in the AgilePoint Server automatically when AgilePoint Server Hosted Service is created. This can be changed for actual production application to client specific domains and active directory connection. Following are the users and their passwords, you can log in your web application using any of these users:
   1. ASTN\Lily.Allen - pass
   2. ASTN\Bob.Hope - pass
   3. ASTN\Bryan.Ferry - pass
   4. ASTN\Michael.Jones - pass
   5. ASTN\Grace.Kelly - pass



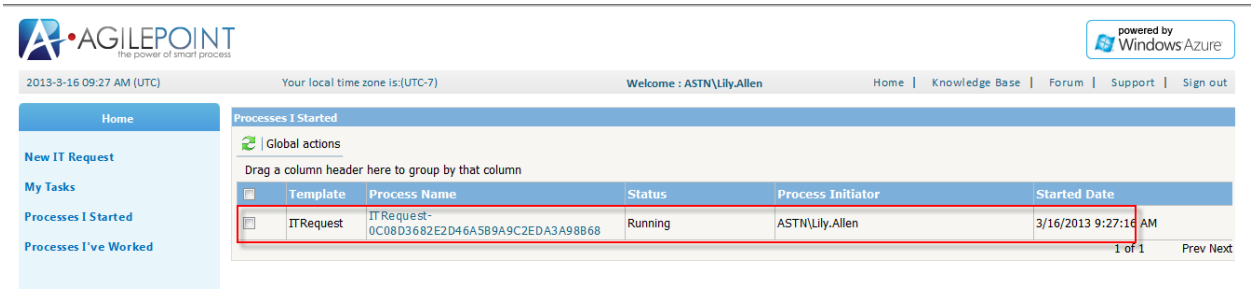
3. After Login, click on "**New IT Request**" link to create new request.



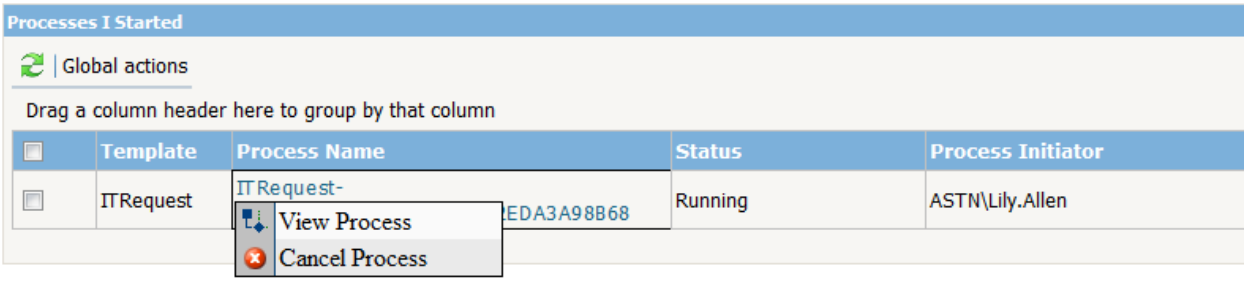
4. You will see the page we designed earlier:



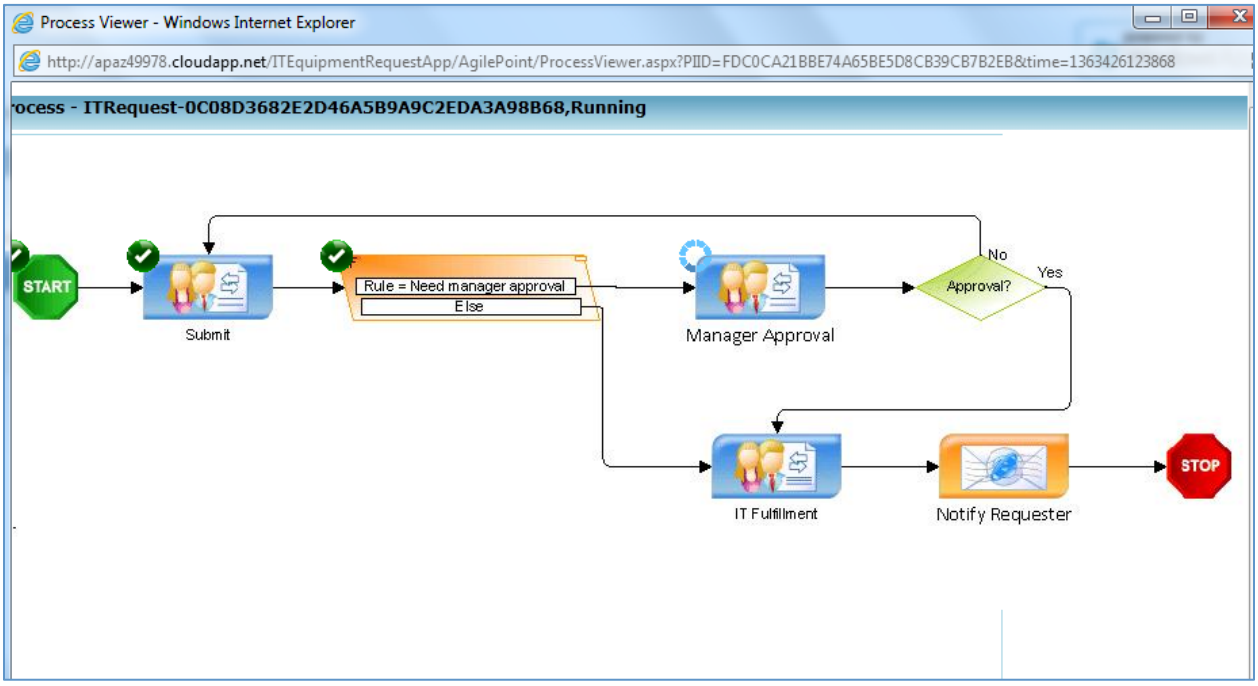
5. Fill in the information and click on Submit. It will start the process and you will be redirected to page showing "**Processes I Started**"



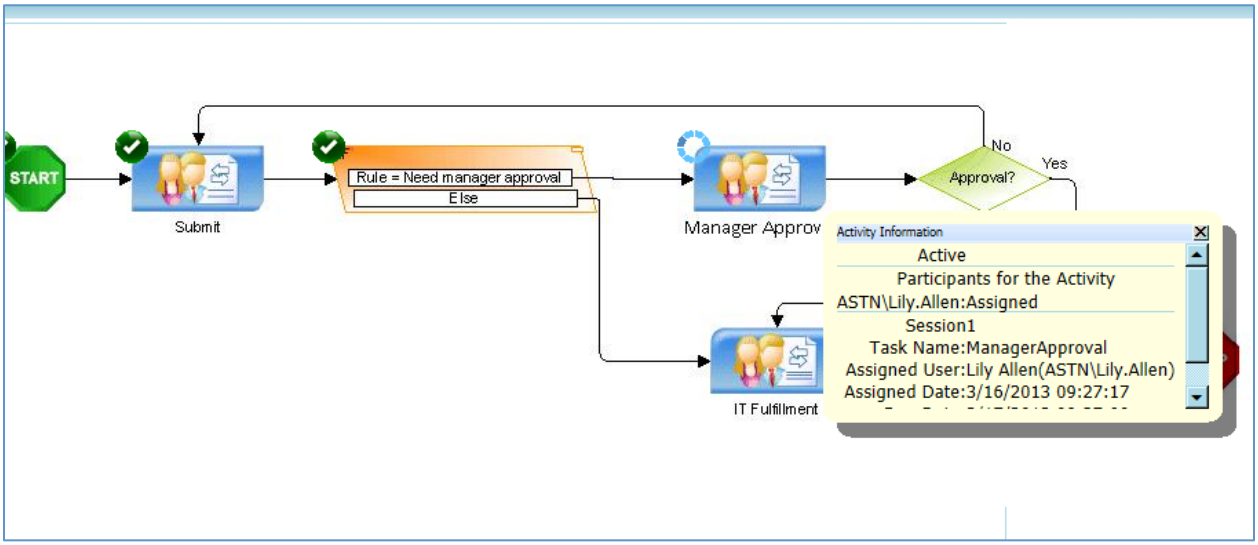
6. Click on the link to open up the options available.



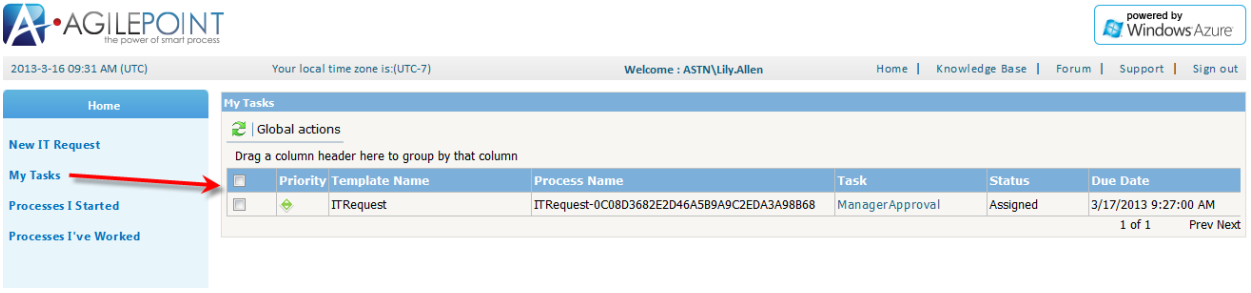
7. Click on "**View Process**" to see the state of running process instance.



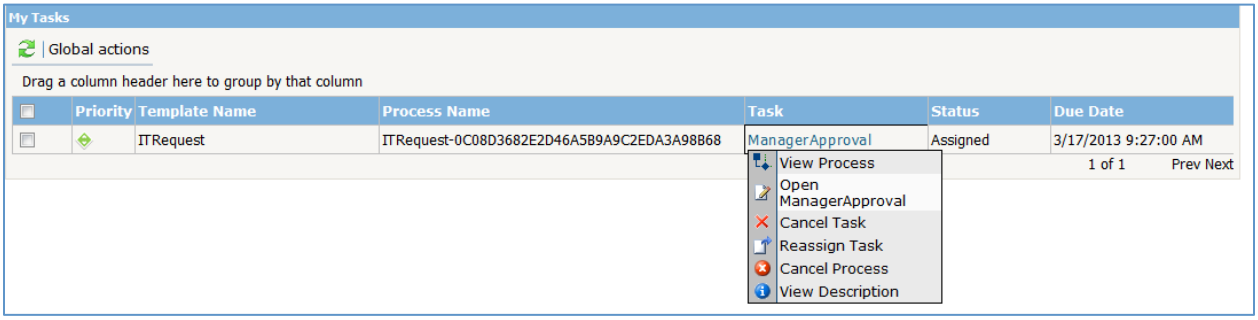
8. Click on the shape to see more information about the shape.



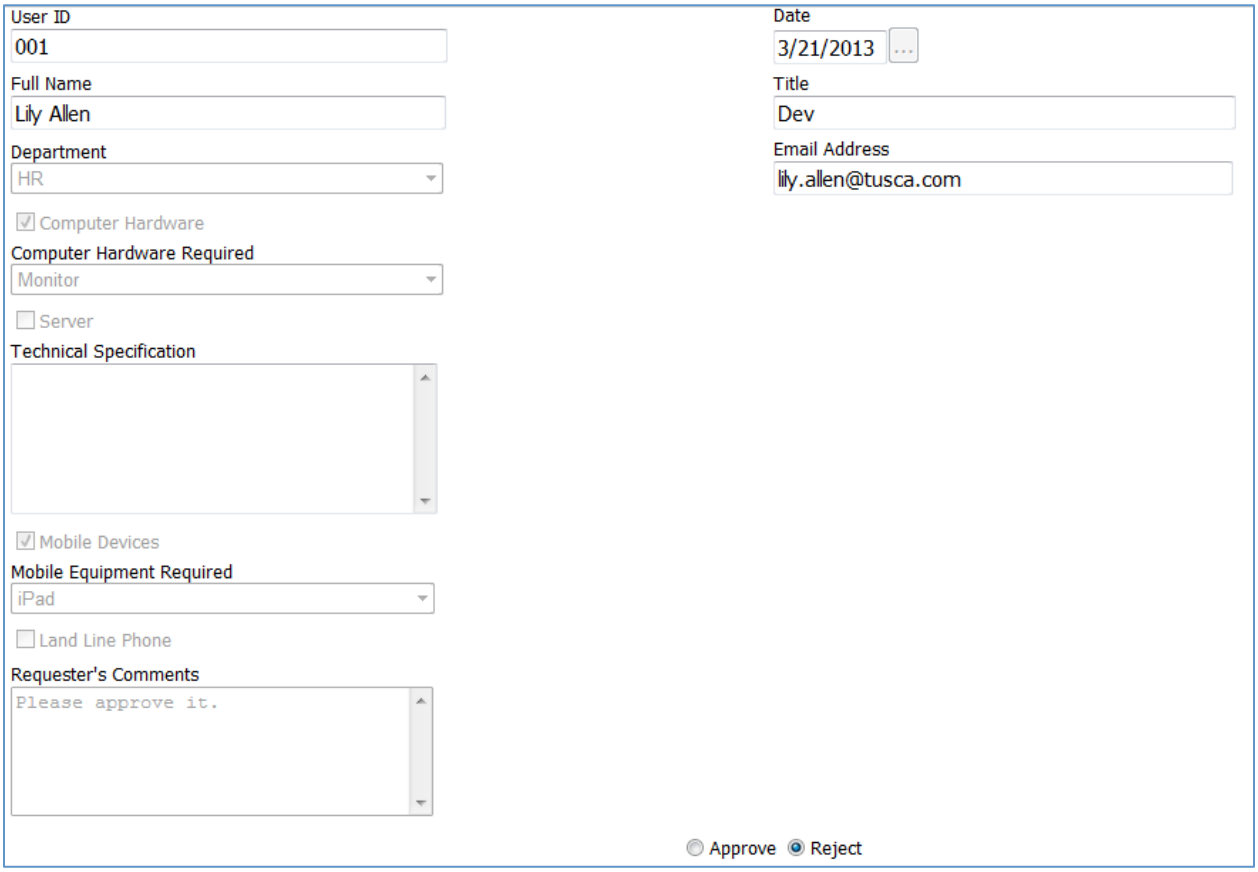
9. Next we will complete the manager approval task. Since in the initial process design we selected same user to be the participant of all 3 manual steps you do not need to log off. This can be changed to unique users or roles after you execute simple demo successfully. Click the "**My Tasks**" link on left hand side panel. That will show up task list web part.



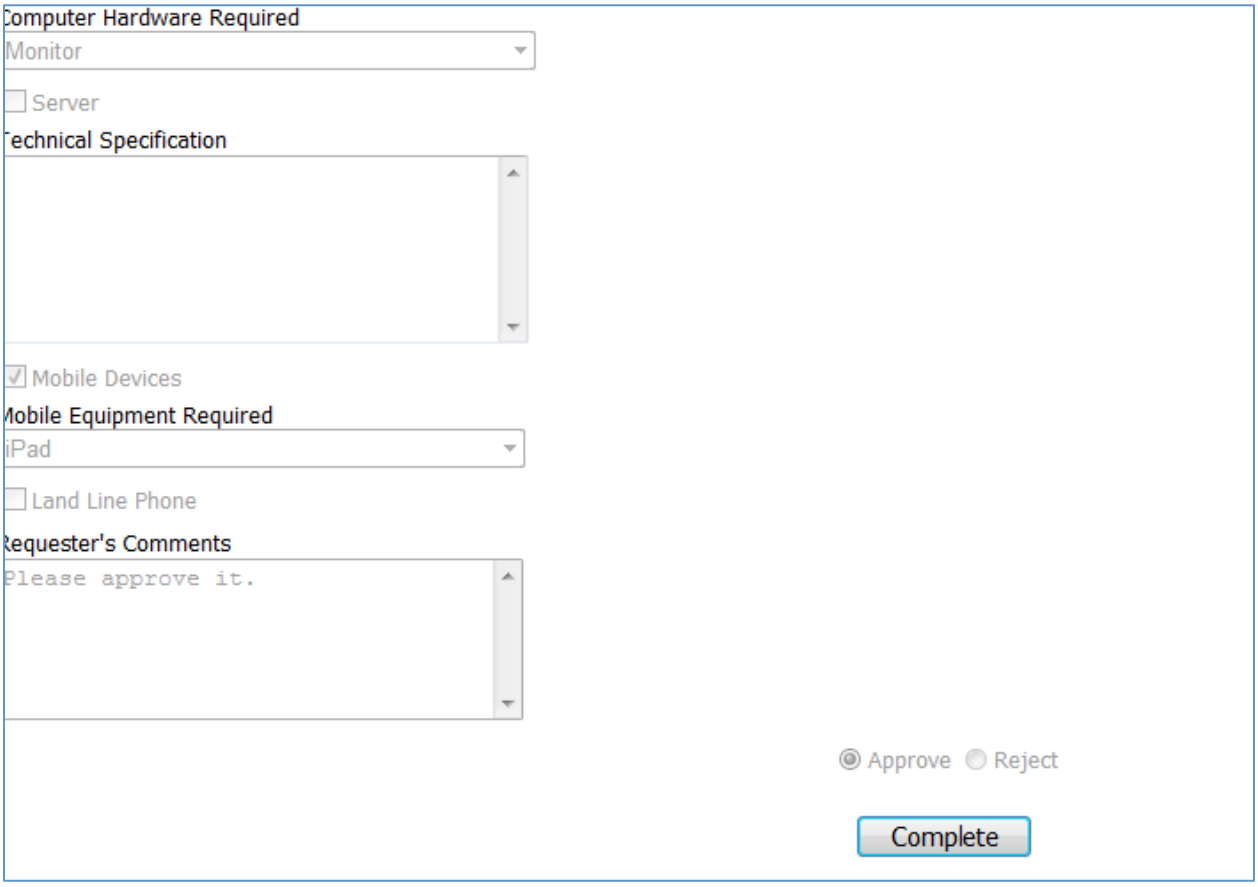
10. Click on the task link to see available operation you can perform on this task. These options depend on the permissions this user has in AgilePoint. In an evaluation environment, these are already configured for you.



11. Click "**Open ManagerApproval**" to open the task. You can see all information entered while submitting a new request is automatically available in approval screen. This is because of the Binding Name property on each control that ties the data between process and UI.



12. Select the Approve radio button, and click Submit.
13. Navigate back to the task list and open the ITFulfullment task and click on complete to finish the process.

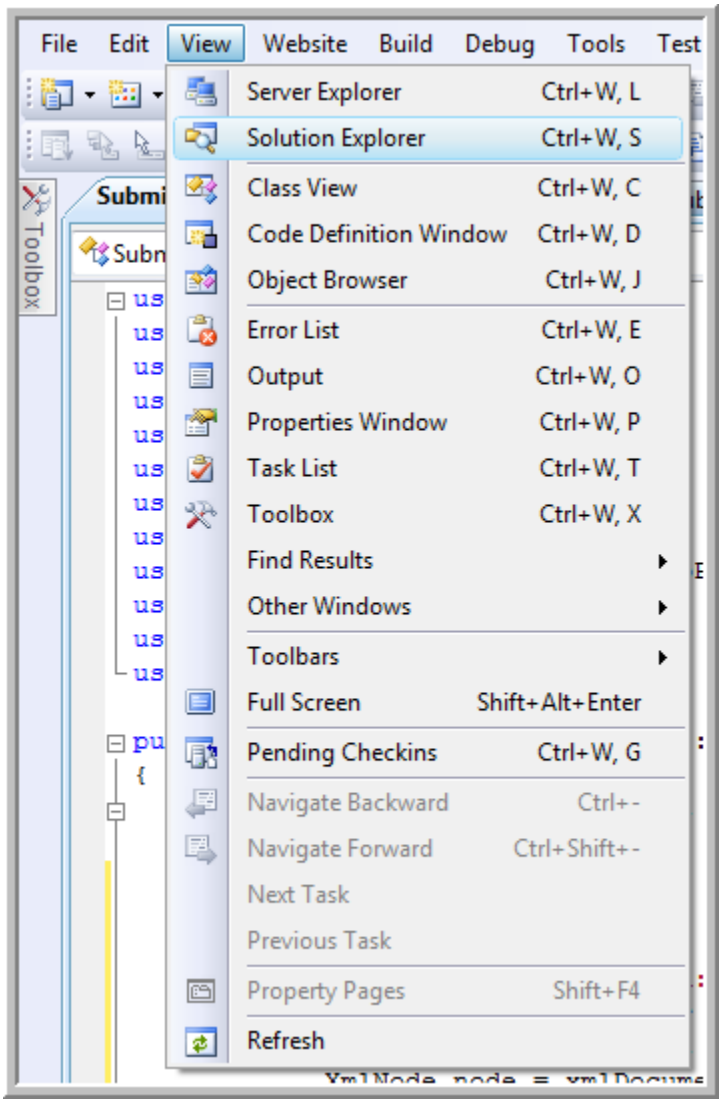


Advanced Scenarios

Include a repeating section using the WFGridView Control

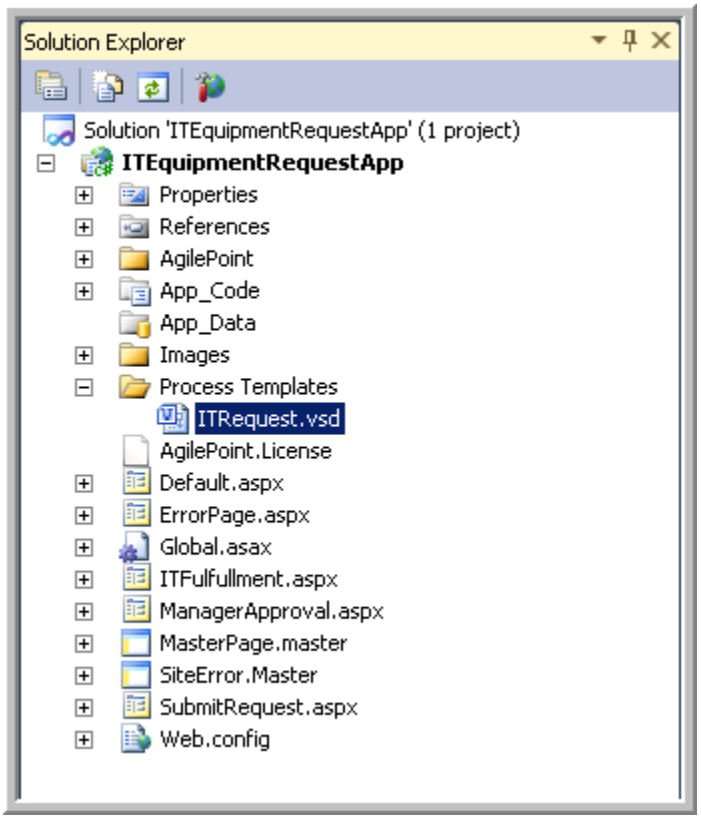
Now that the simple ASP.Net web application exercise covered in previous section is successfully completed and executed, we will try to accomplish some advance scenario within that where we will introduce a repeating node in the schema and bind it to WFGridView control.

1. Click on **View->Solution Explorer**.

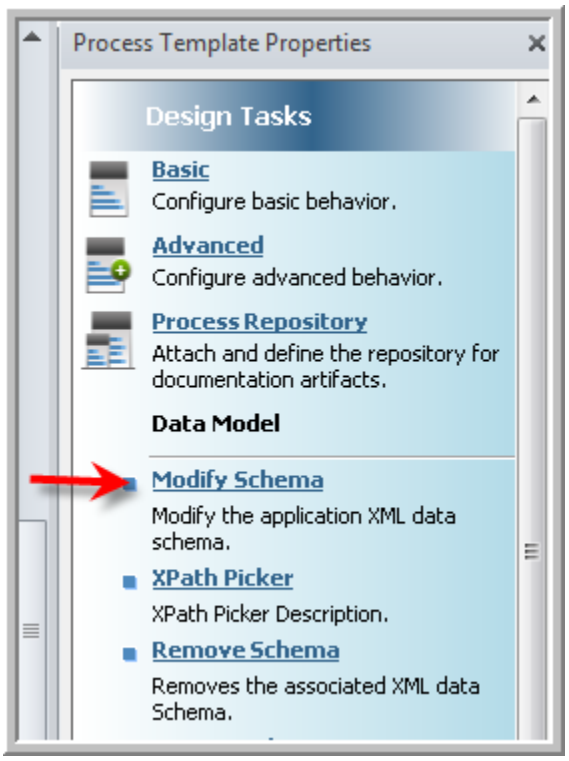


Picture: Open Solution explorer.

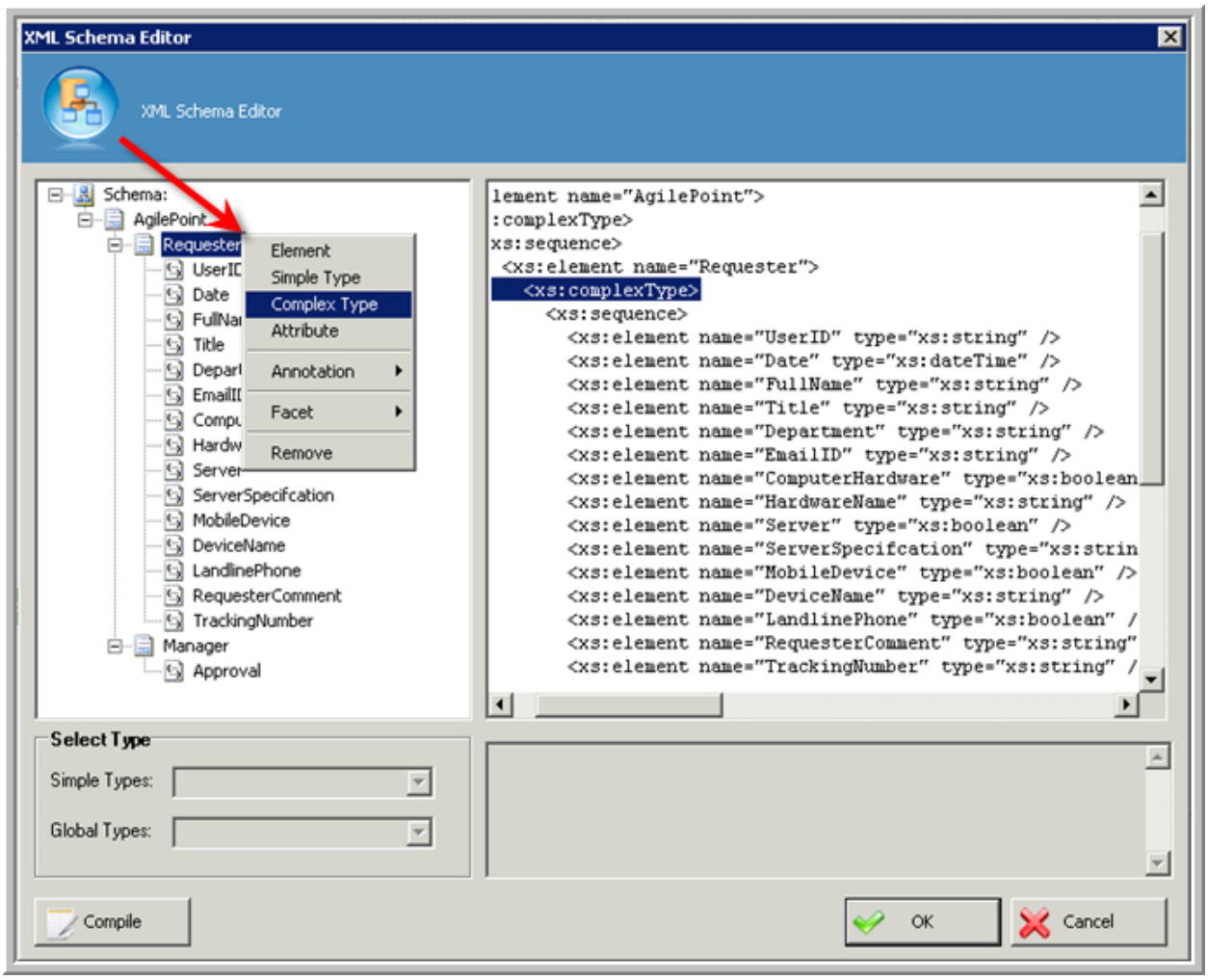
2. Double-click "**ITRequest.vsd**" from Solution Explorer. It will take you to AgilePoint Envision where you can modify your process design and schema.



3. Click on "**Modify Schema**" link in the property panel on right hand side.

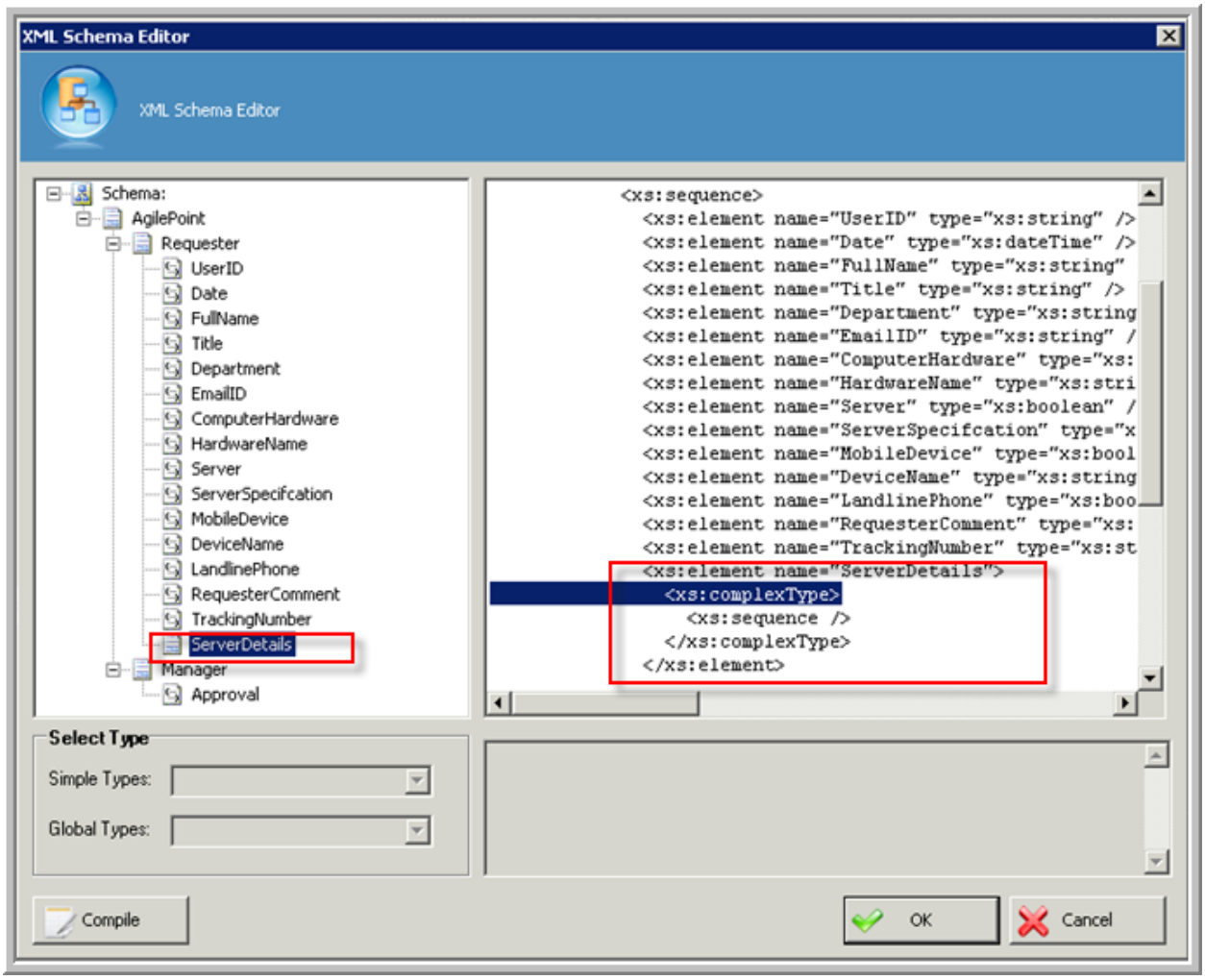


4. Select "**Requester**" node by left click (do not select any node by right click). Then right click on selected node and click "Complex Type" option to add a category for ServerDetails information where user can add multiple server requirements. Name the newly created complex type as "ServerDetails".



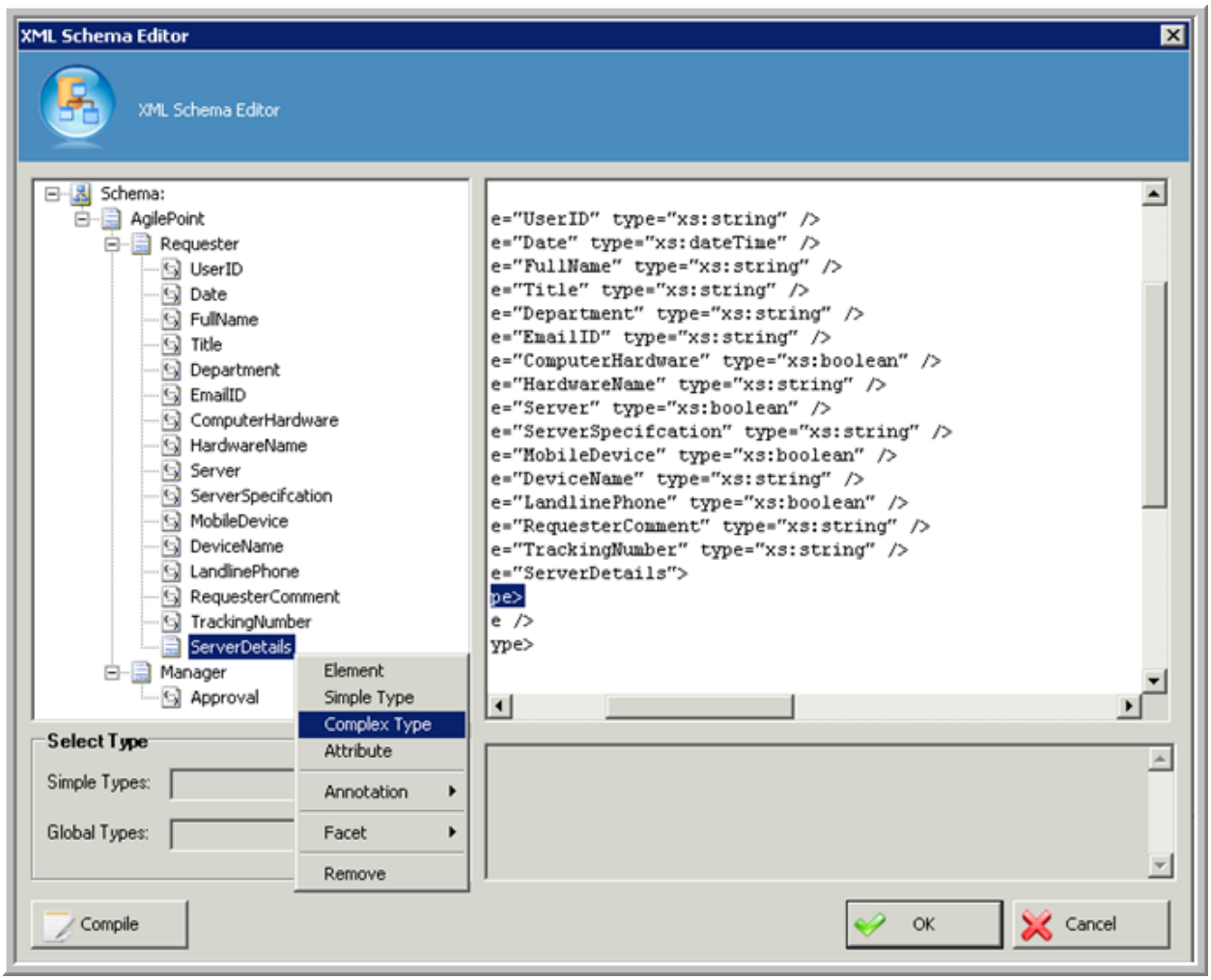
Picture: "Create Schema" dialog box to add "ServerDetails" node

5. Once the node is added to the schema, it should appear in the schema tree structure as shown below.



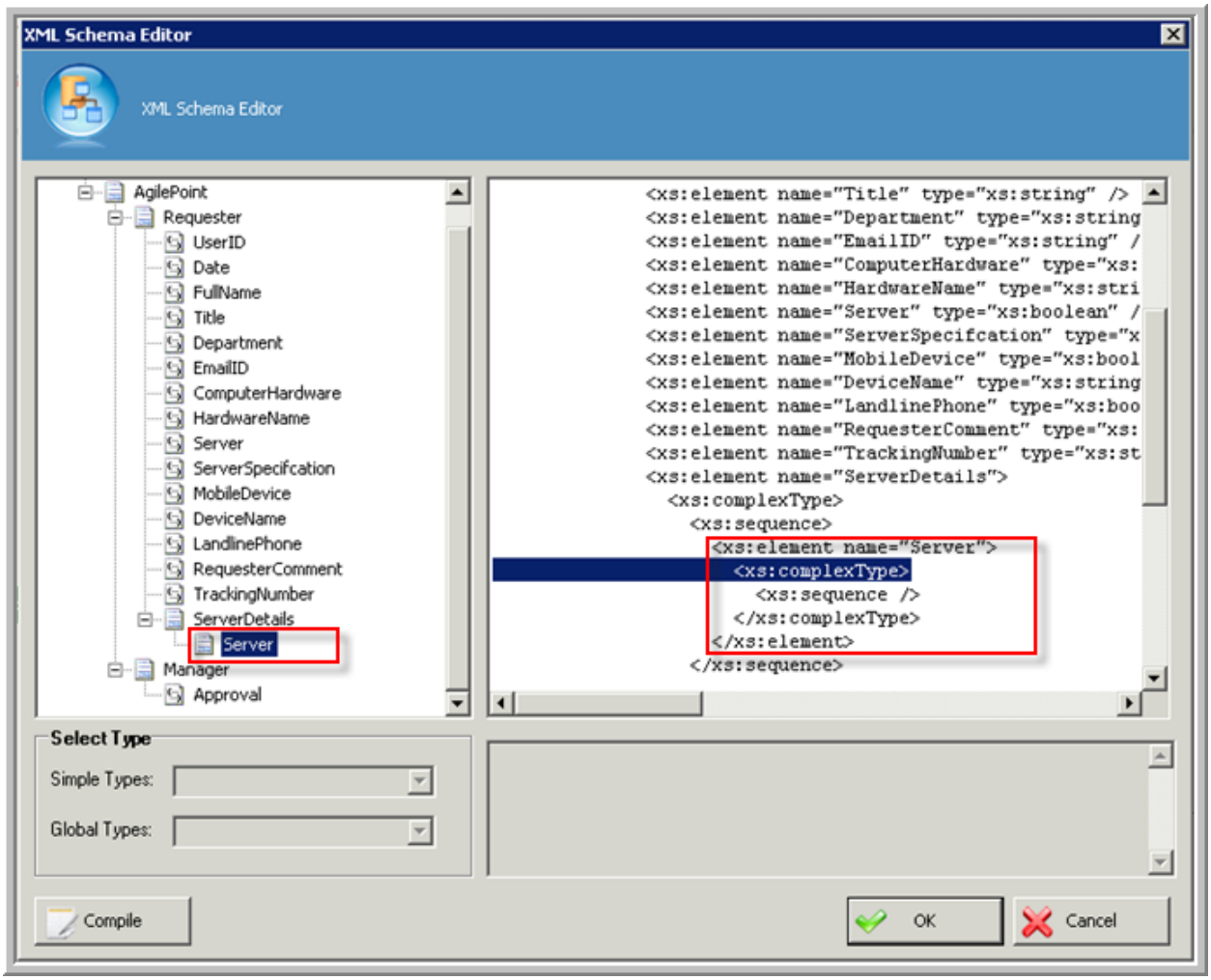
Picture: "ServerDetails" element after creation.

6. Next select the newly added "ServerDetails" node by left click to make it the active node (do not select any node by right click). Then right click on selected node and click "Complex Type" option to add a node for individual Server information. Name the newly created complex type "Server".



Picture: "Create Schema" dialog box to add "Server" node

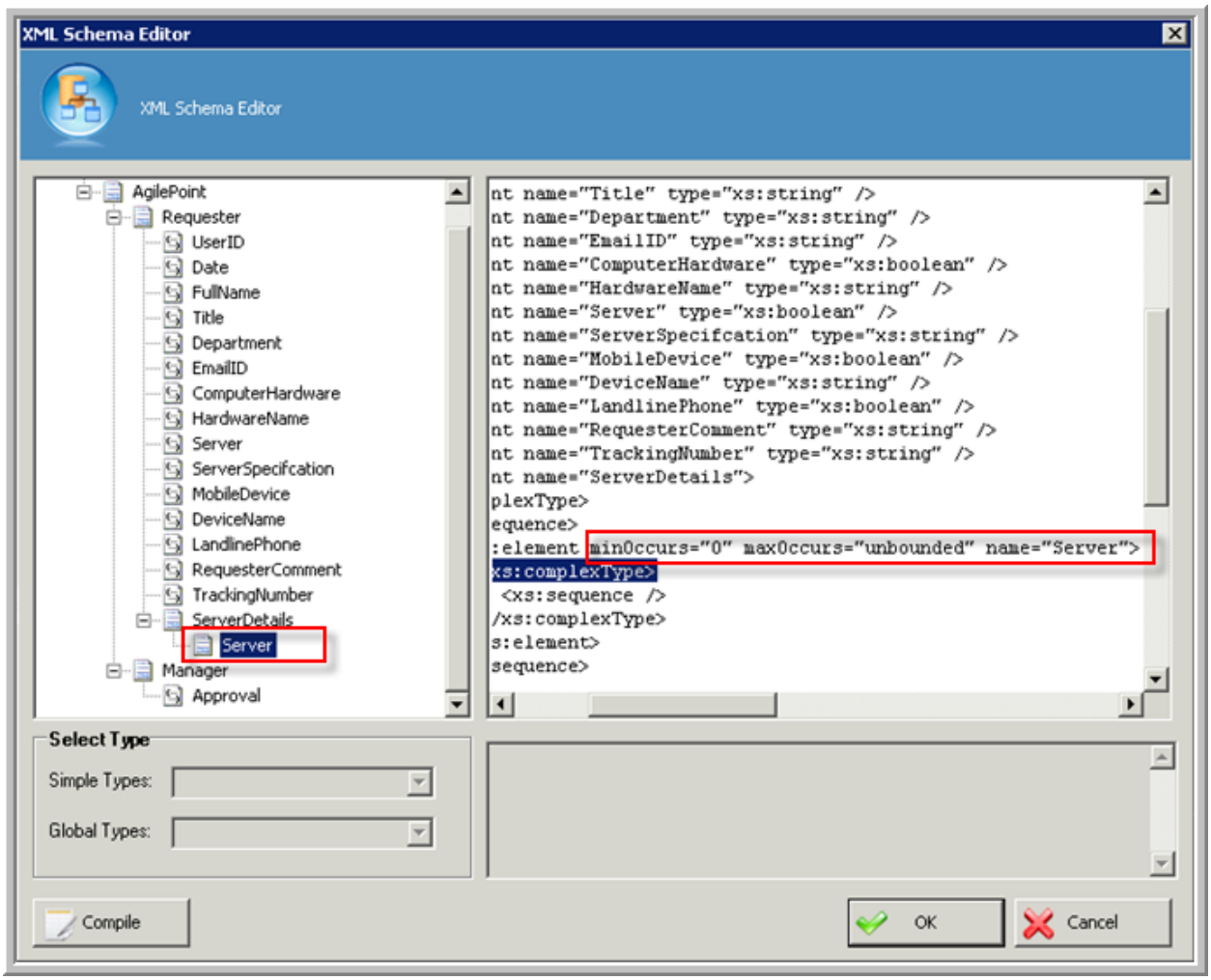
7. Once the node is added to the schema, it should appear in the schema tree structure as shown below.



Picture: "Server" element after creation.

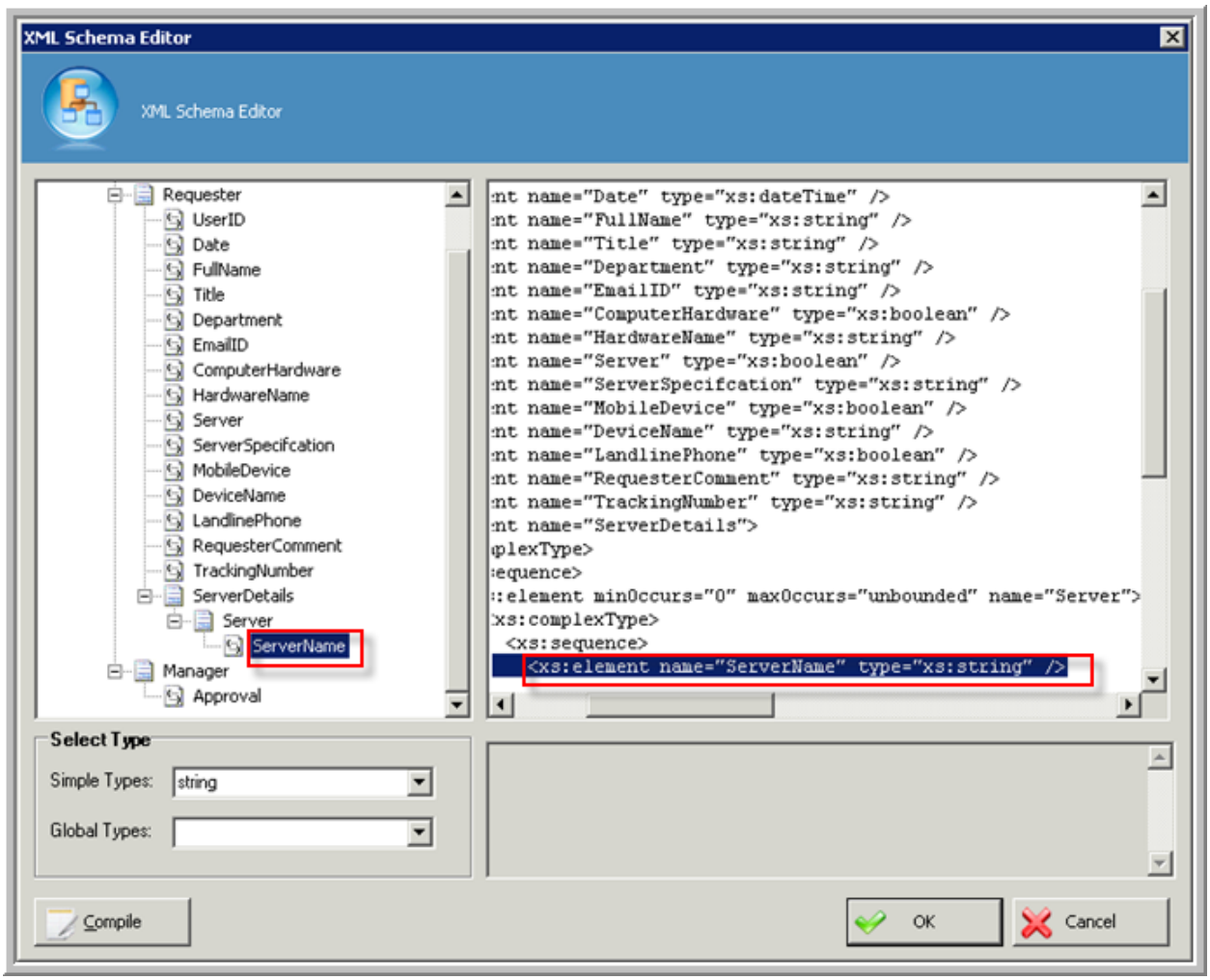
8. Make the "**Server**" node repeatable by adding `minOccurs` and `maxOccurs` XML properties in right hand side XML editor inside "Server" element. Have a look on below code.

```
<xs:element minOccurs="0" maxOccurs="unbounded" name="Server">
```



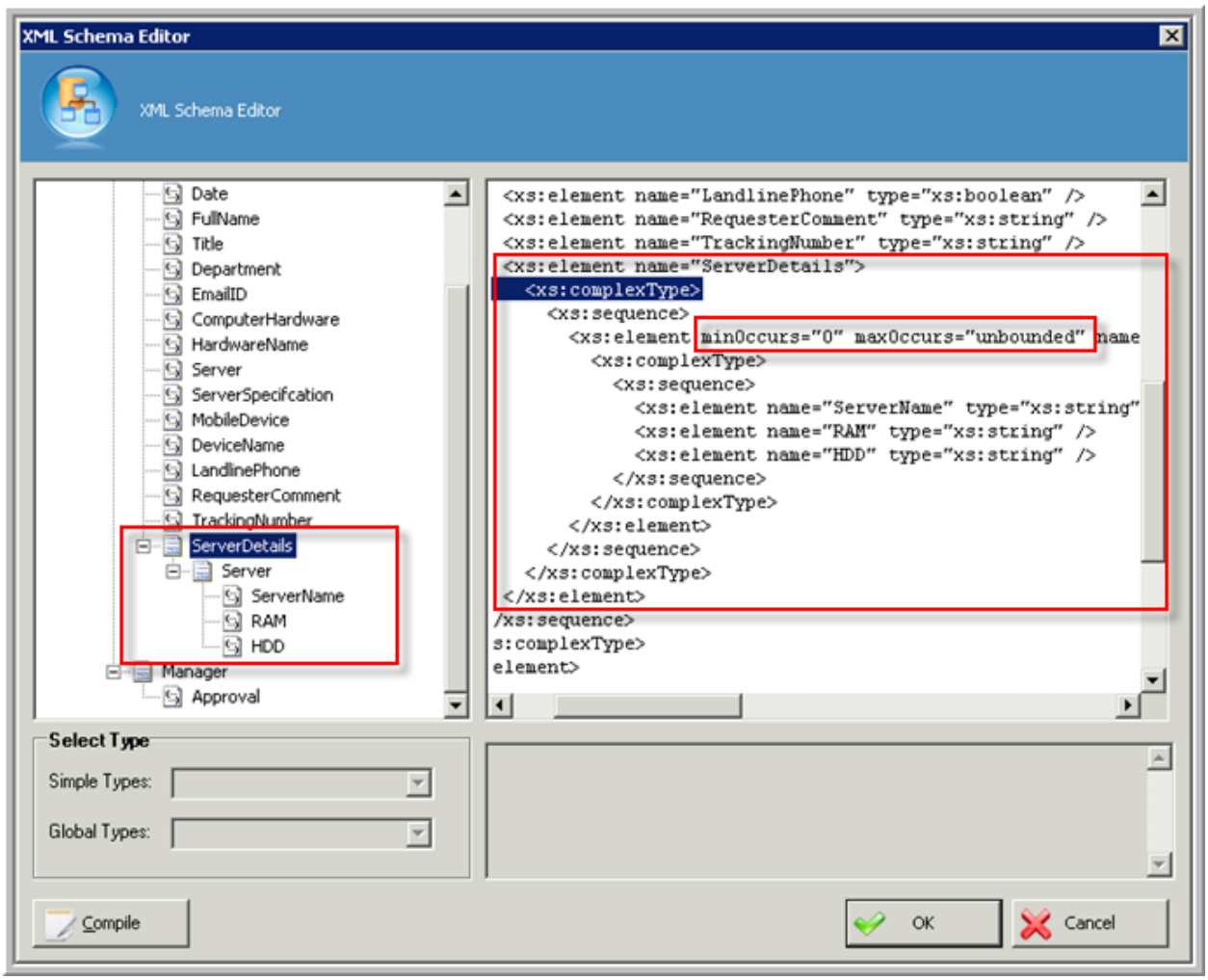
Picture: "Create Schema" dialog box to add "Server" node and make repeatable.

9. Create an element called "ServerName". The default data type of an element is string. If required, the data type can be changed by selecting appropriate data type from the "Simple Types" drop down box at the left hand side panel towards bottom of the screen.



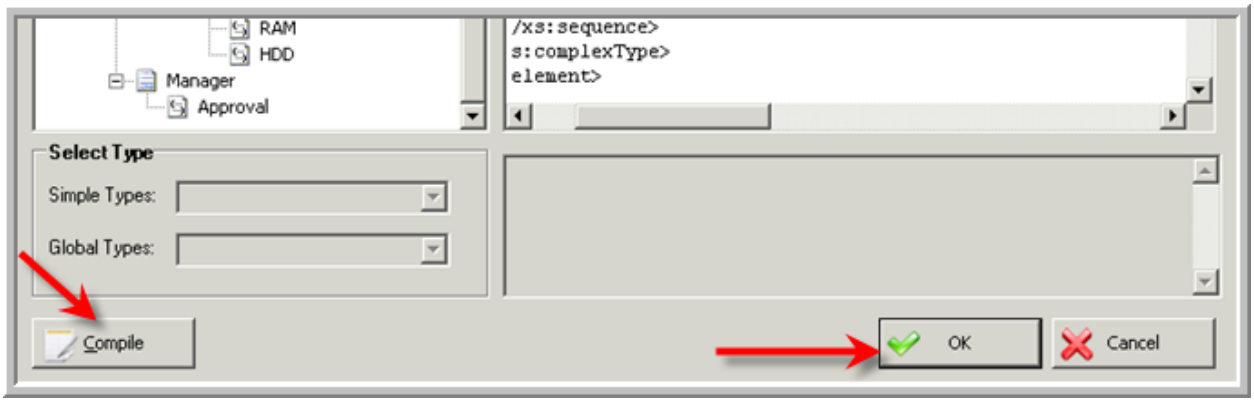
Picture: Creating "ServerName" element in AgilePoint Envision.

10. Repeat the action in previous step to create rest of the fields - RAM (string) and HDD (string) elements under "Server" category. Value in parenthesis indicates data type. Once this step is complete, your schema should show the new nodes as shown below.

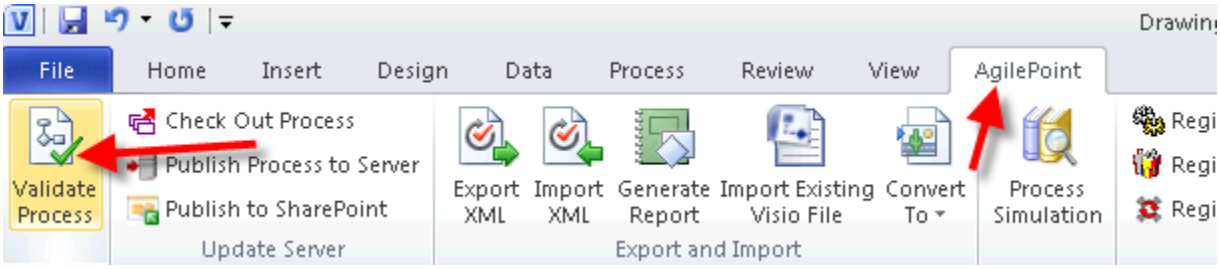


Picture: Repeatable "Server" category to store Servers.

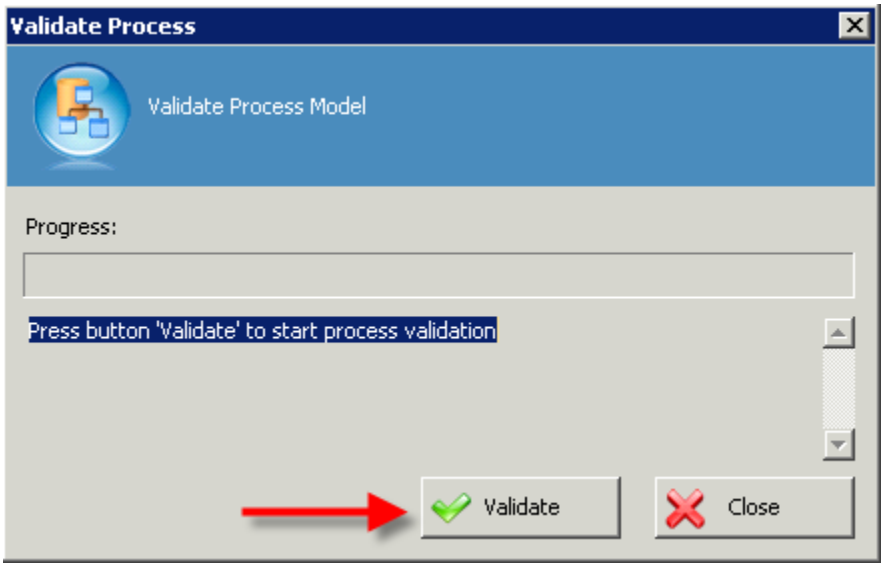
11. After schema creation please click compile button to check if it is a valid schema and then click on OK button as shown below.



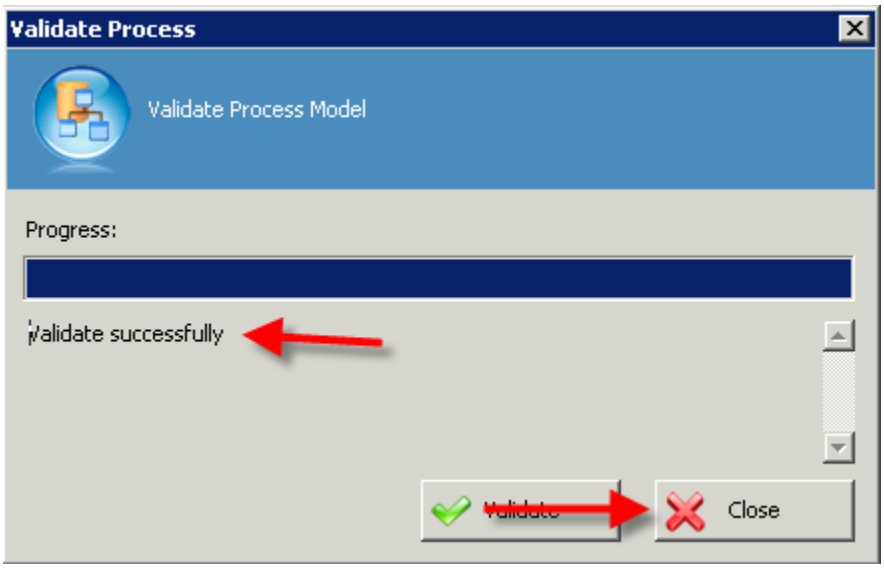
12. Now your updated process is ready to publish to AgilePoint Server. Before that click on “Validate Process” button in AgilePoint ribbon to check if there is no error in the process template and it is good to publish to server.



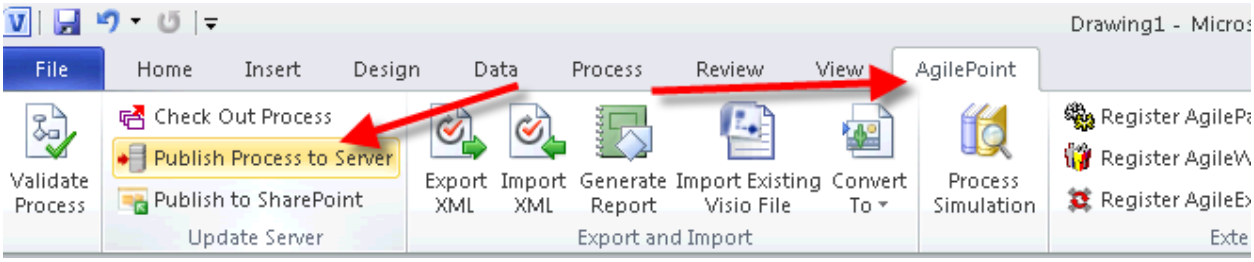
13. Click on Validate button.



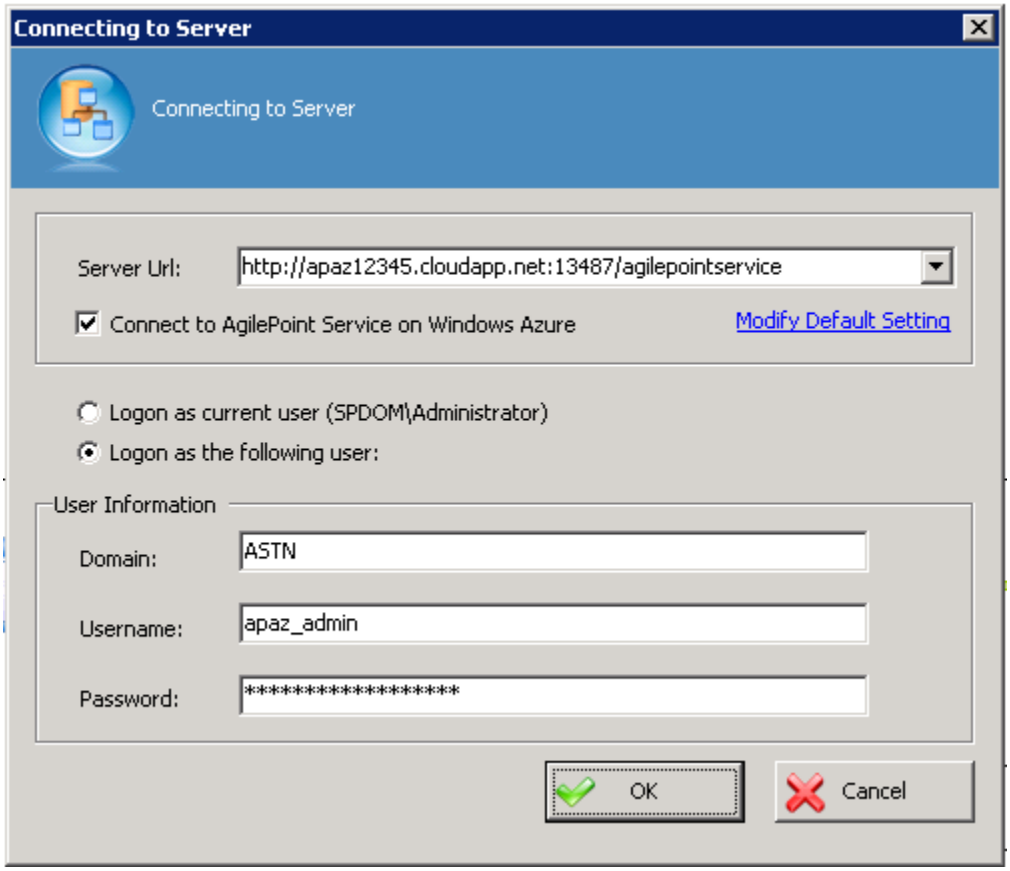
14. Once process is validated successfully click the Close button.



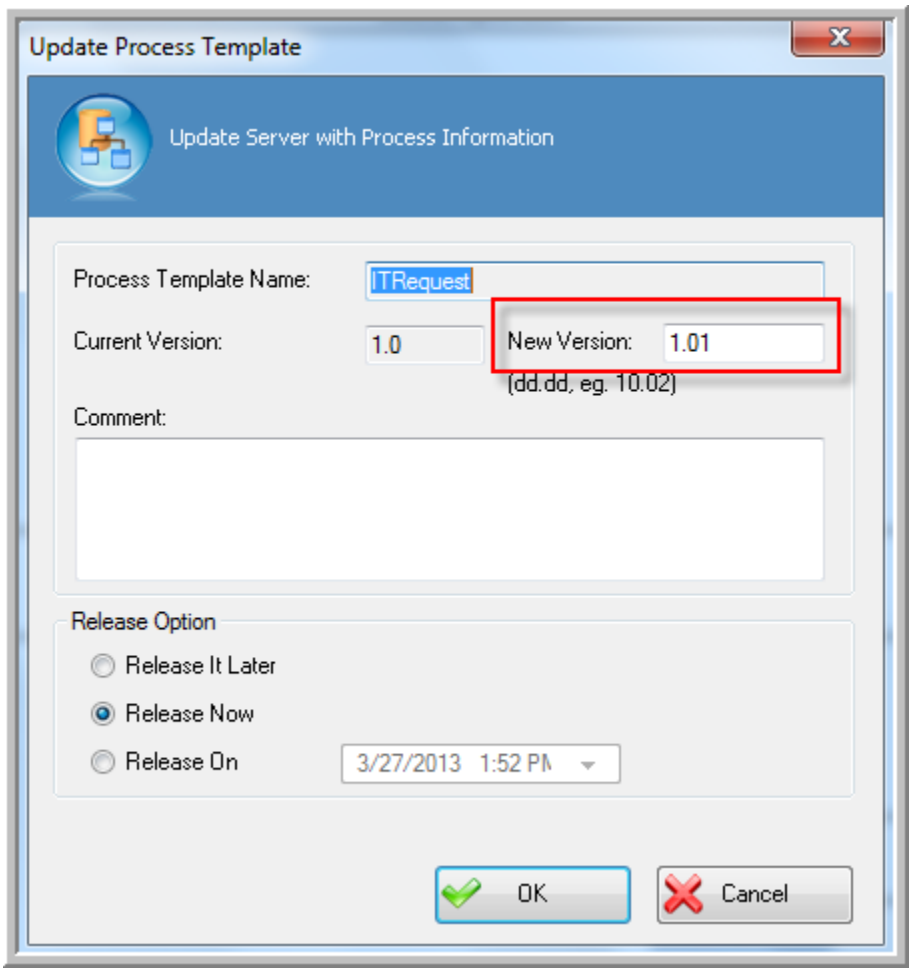
15. Click the “Publish Process to Server” button from the “AgilePoint” ribbon menu.



16. This will open up a dialog to provide your AgilePoint Service URL and credentials to connect to the server. Provide the information as shown below, and click OK.

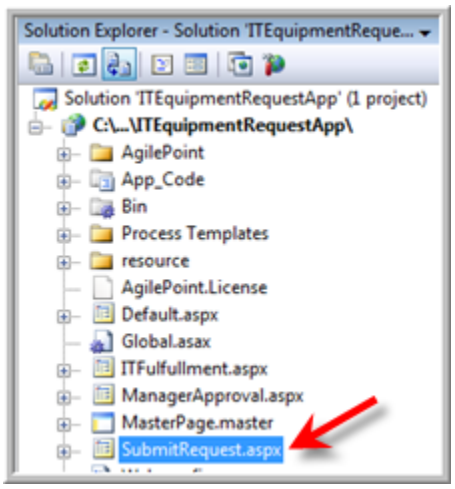


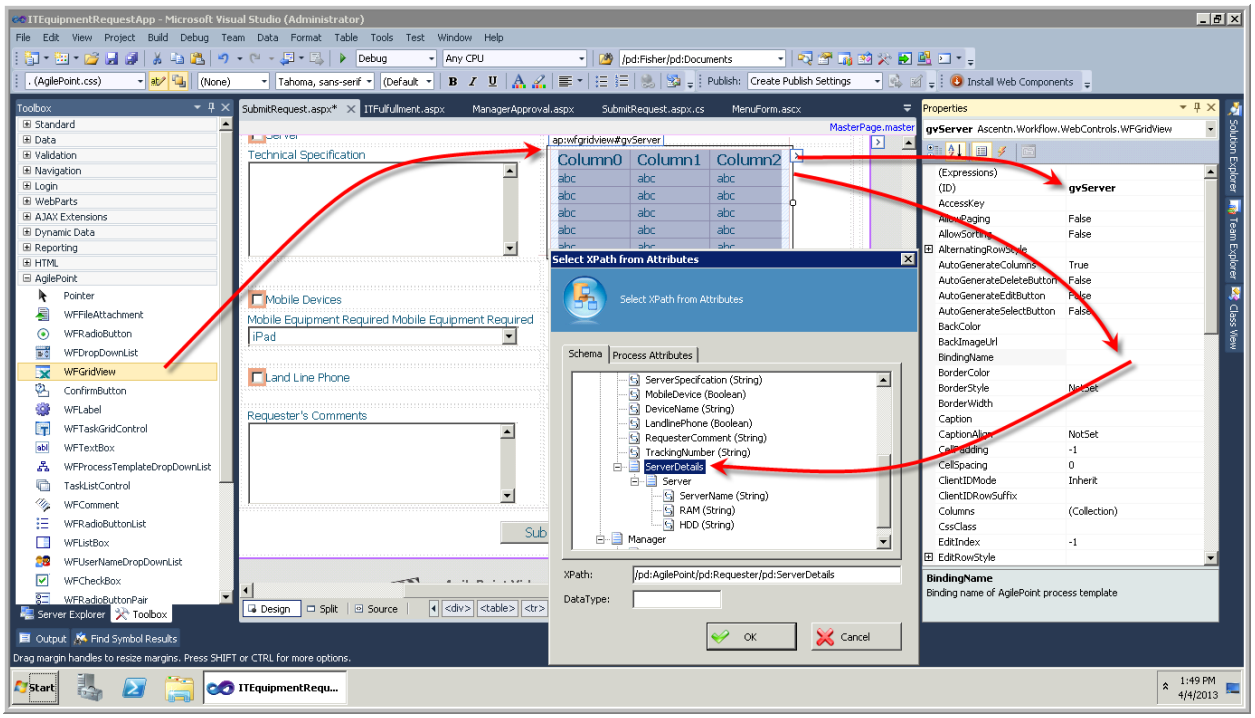
17. Leave the default settings as it is and click the OK button to publish the new version (1.01) of the ITRequest process to AgilePoint Server.



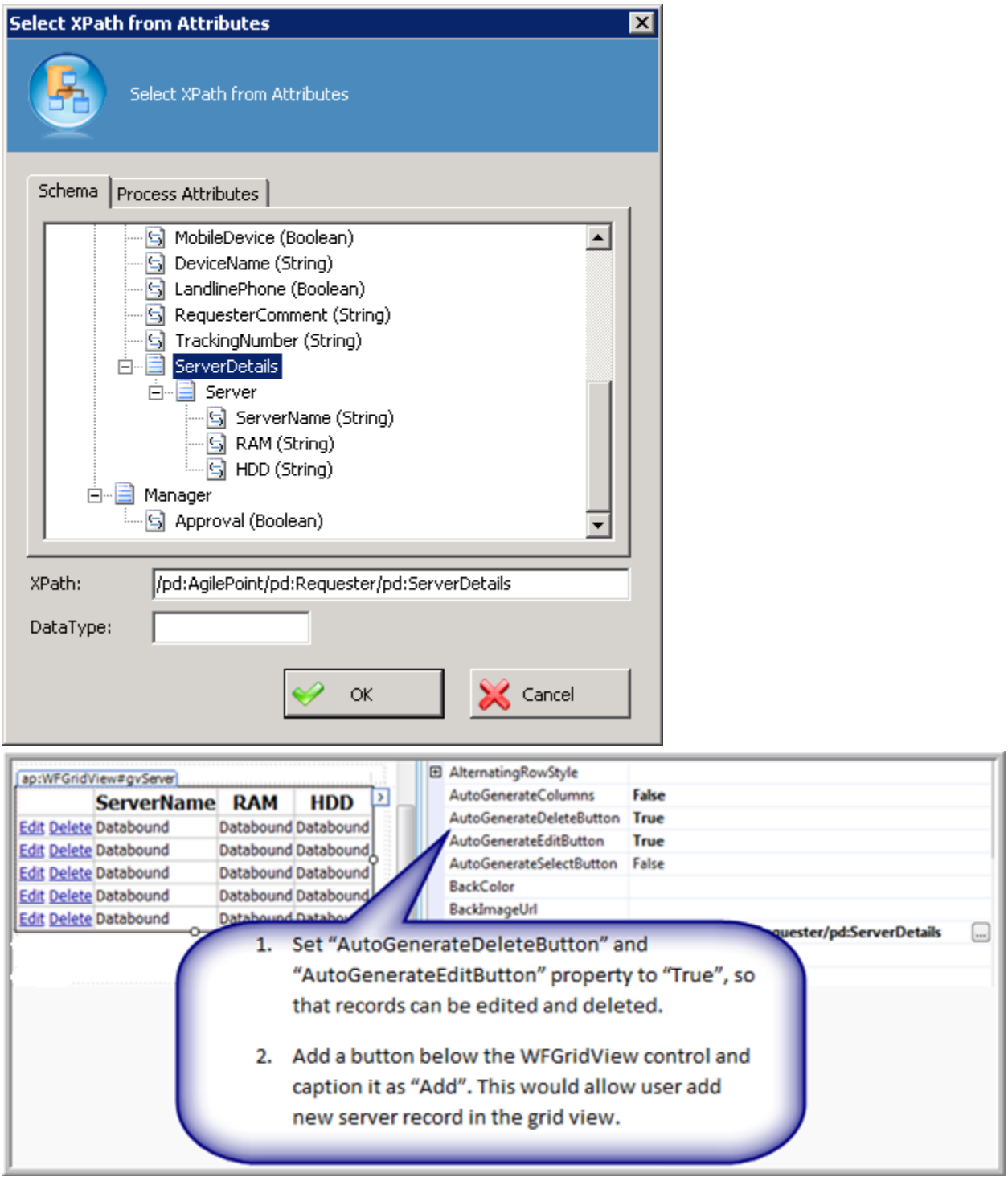
18. Now save the process model in Visio and close AgilePoint Envision then go back to ASP.Net web application.

19. Open **SubmitRequest.aspx**. Drag and drop **WFGridview** control from the AgilePoint toolbox on the page design area next to the server specification text area. Go to properties panel of the **WFGridview** control and provide ID as "**gvServer**" and then select **BindingName** property. Select the process template, "**ITRequest**" in your case, when prompted to select the schema.





20. Select "**ServerDetails**" node from the schema tree. This step should generate GridView layout and html as shown in the picture below.



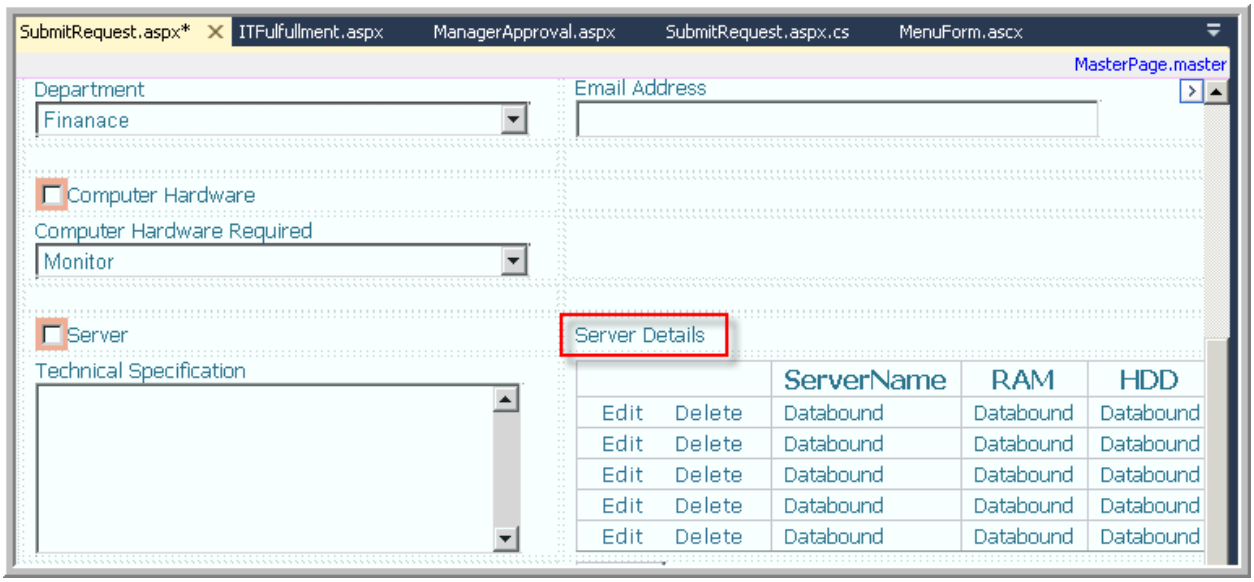
The auto generated HTML code would look something like this.

```
<ap:WFGridView ID="gvServer" runat="server" AutoGenerateColumns="False"
      AutoGenerateDeleteButton="True" AutoGenerateEditButton="True"
      BindingName="/pd:AgilePoint/pd:Requester/pd:ServerDetails"
      EnableModelValidation="True"
      InnerXmlTemplate="PHBkOlNlcnZlcnMgeG1sbnM6cGQ9Imh0dHA6Ly93d3cuYXNjZW50b
```

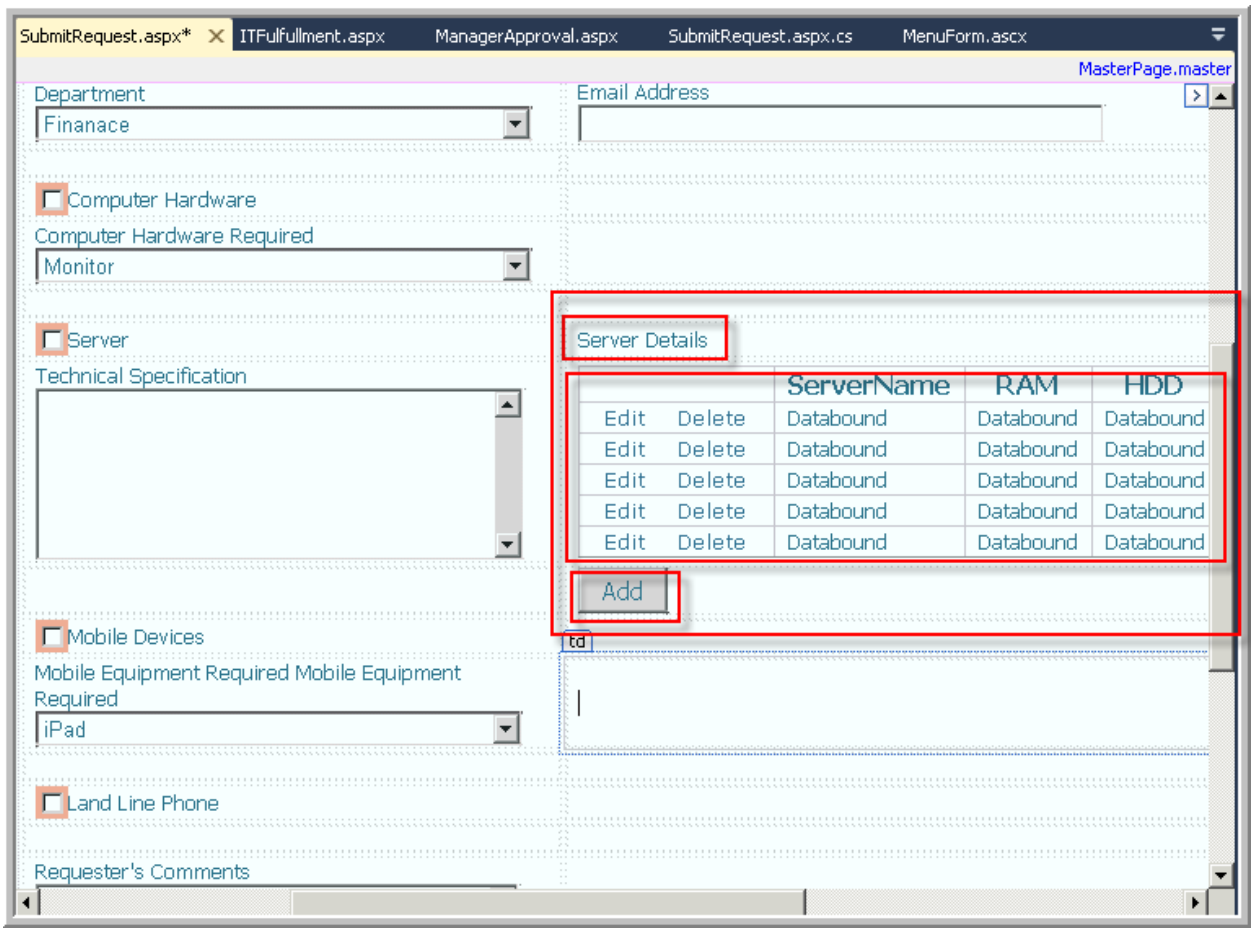


```
i5jb20vYnBtL1hNTFNjaGVtYSI+PHBkOlNlcnZlck5hbWU+U3RyaW5nPC9wZDpTZXJ2ZXJO
YW1lPjxwZDpSQU0+U3RyaW5nPC9wZDpSQU0+PHBkOkhERD5TdHJpbmc8L3BkOkhERD48L3B
kOlNlcnZlcnM+">
<Columns>
<asp:BoundField DataField="pd:ServerName" HeaderText="ServerName" />
<asp:BoundField DataField="pd:RAM" HeaderText="RAM" />
<asp:BoundField DataField="pd:HDD" HeaderText="HDD" />
</Columns>
</ap:WFGridView>
```

21. Once **WFGridview** control has been added to the page and all settings are configured then provide title of the control as "**Server Details**" on top of the **WFGridview** control.



22. Next add one ASP.Net button below the **WFGridView** control for generating an empty row in the **WFGridview** control. After adding the **WFGridView** Control, **Title** and **ASP.Net button** on page. The page would appear as below.



23. When user clicks the "**Add**" button, it should create an empty row in the WFGridView control, so that user can add new Server record. Double-click the "Add" button to generate button click event method. Write the following code in the "Add" button click event method.

```
DataTable dt = gvServer.RepeatingSectionDataSource;
if (dt != null)
{
        DataRow child = dt.NewRow();
        child[0] = "updates";
        child["pd:ServerName"] = "";
        child["pd:RAM"] = "";
        child["pd:HDD"] = "";
        dt.Rows.Add(child);
        gvServer.RepeatingSectionDataSource = dt;
        gvServer.BindRepeatingSectionDataSoure();
}
```

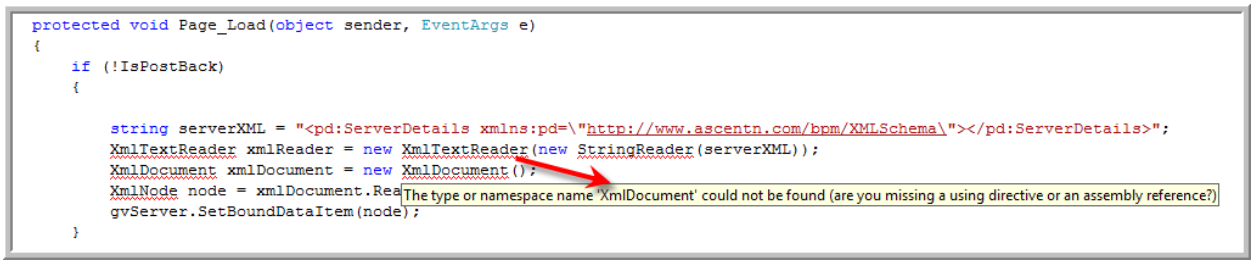
Note: The above code for WFGridView control is NOT using DataSource property and DataBind method to bind with data source, instead it is using AgilePoint custom property "RepeatingSectionDataSource" and custom method "BindRepeatingSectionDataSource". Making a call to this property/method would handle the entire data binding operation. Also, while creating a new row in data table, it's loading existing data source which is bound to the control and it automatically figures out the structure for the new record.

24. Once the design part is done, switch to the code behind file "**SubmitRequest.aspx.cs**". The Page_Load event method has some auto generated code provided by AgilePoint component to save some coding effort. Add the code below after the existing auto generated code in Page_Load method.

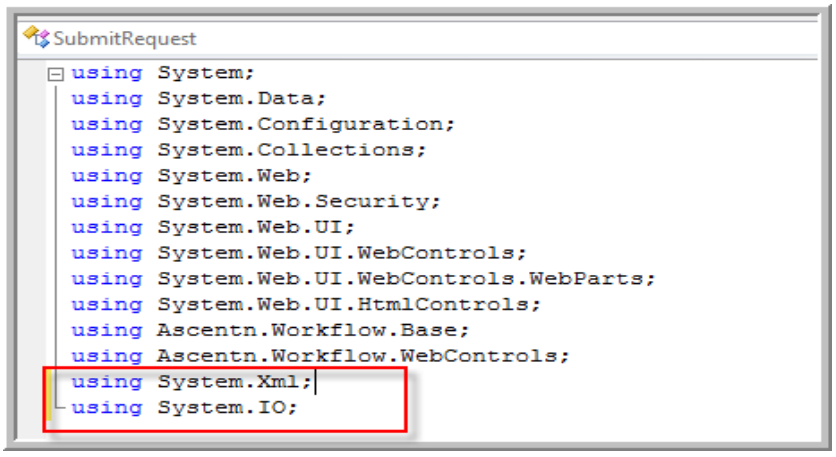
    This step is needed only if the WFGridView control is used on a web form, which starts the AgilePoint process.

```
if (!IsPostBack)
  {

     string serverXML = "<pd:ServerDetails
     xmlns:pd=\"http://www.ascentn.com/bpm/XMLSchema\"></pd:ServerDetai
     ls>";
      XmlTextReader xmlReader = new XmlTextReader(new
     StringReader(serverXML));
      XmlDocument xmlDocument = new XmlDocument();
      XmlNode node = xmlDocument.ReadNode(xmlReader);
      gvServer.SetBoundDataItem(node);
   }
```

If you find any error regarding above metnioned code, like shown below.



Then, please add below mentioned namespace on top the page.

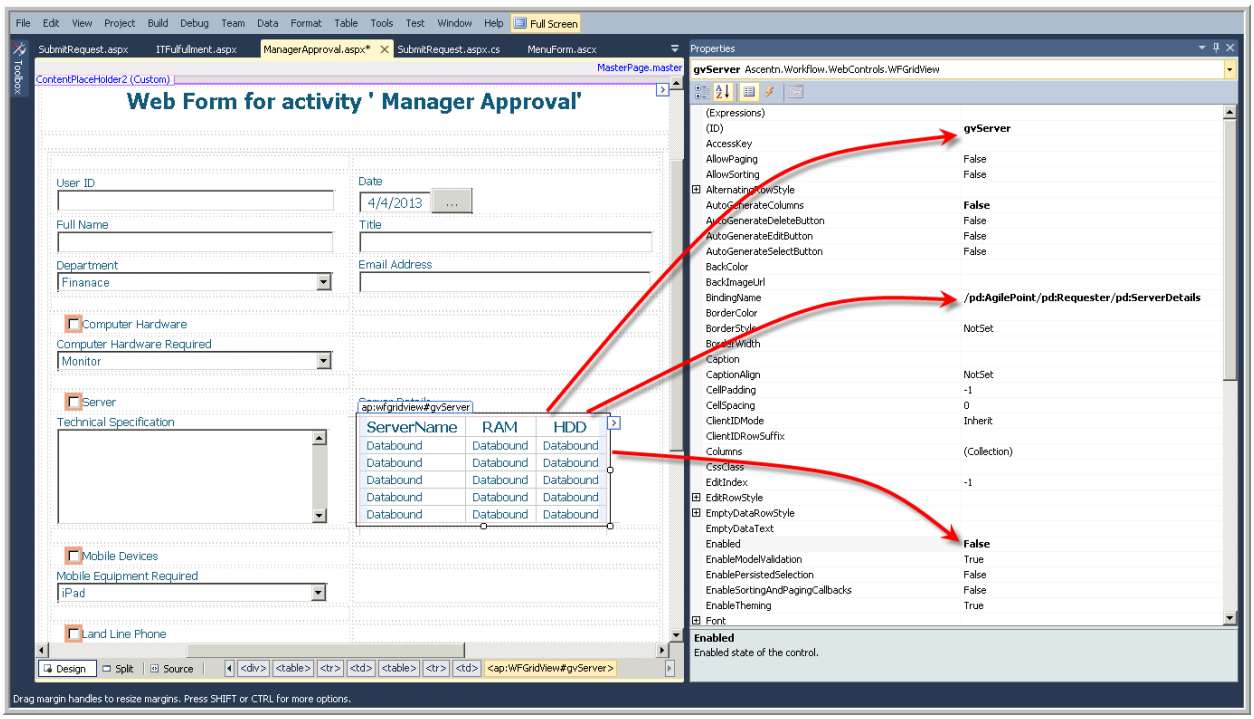



```
public partial class SubmitRequest : Ascentn.Workflow.WebControls.WFWorkSheetPage
{
    protected void btnAdd_Click(object sender, EventArgs e)
    {
        DataTable dt = gvServer.RepeatingSectionDataSource;
        if (dt != null)
        {
            DataRow child = dt.NewRow();
            child[0] = "updates";
            child["pd:ServerName"] = "";
            child["pd:RAM"] = "";
            child["pd:HDD"] = "";
            dt.Rows.Add(child);
            gvServer.RepeatingSectionDataSource = dt;
            gvServer.BindRepeatingSectionDataSoure();
        }

    }
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {

            string serverXML = "<pd:ServerDetails xmlns:pd=\"http://www.ascentn.com/bpm/XMLSchema\"></pd:ServerDetails>"
            XmlTextReader xmlReader = new XmlTextReader(new StringReader(serverXML));
            XmlDocument xmlDocument = new XmlDocument();
            XmlNode node = xmlDocument.ReadNode(xmlReader);
            gvServer.SetBoundDataItem(node);
        }
```

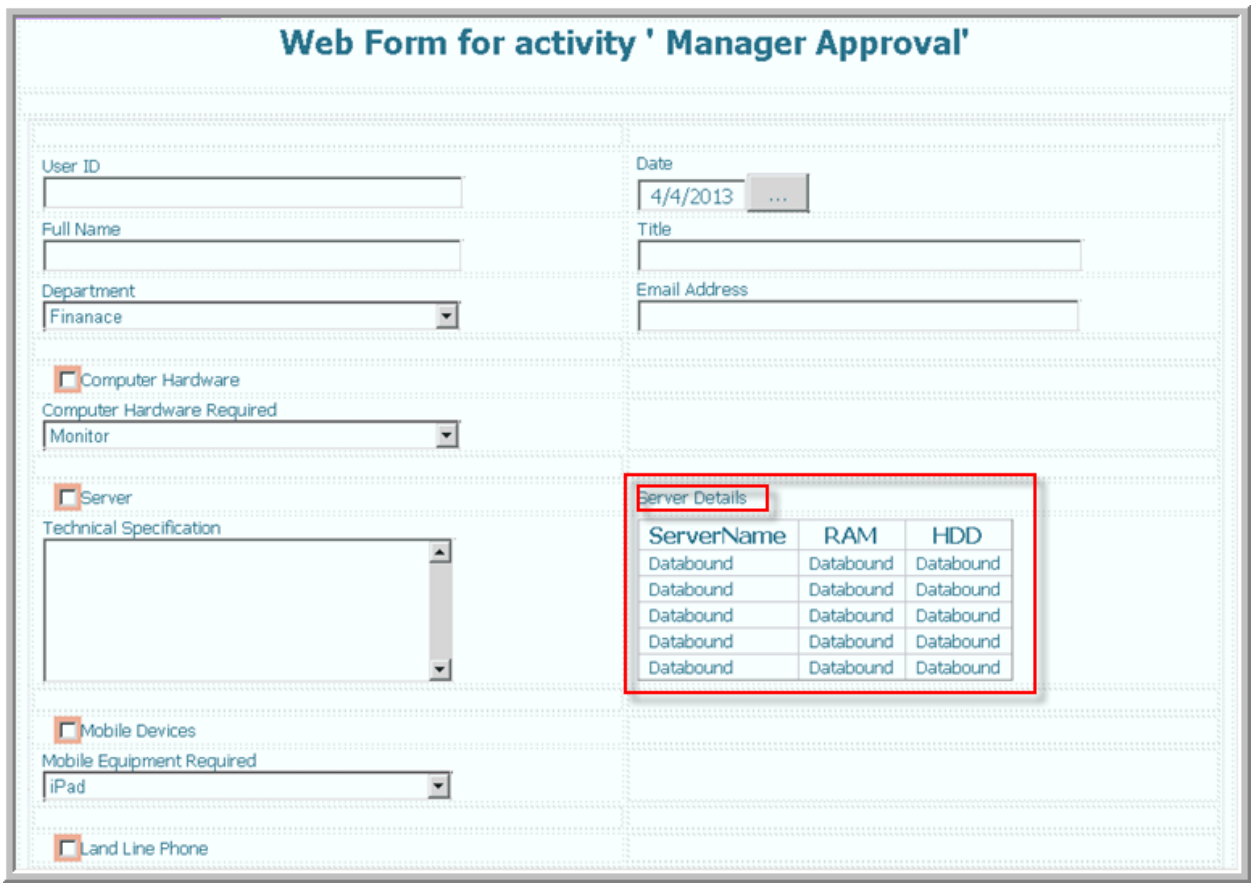
Note – *Above code is for "**Add**" button and "**Page_Load**" event.*

25. Next open "**ManagerApproval.aspx"** page and switch to Design view.

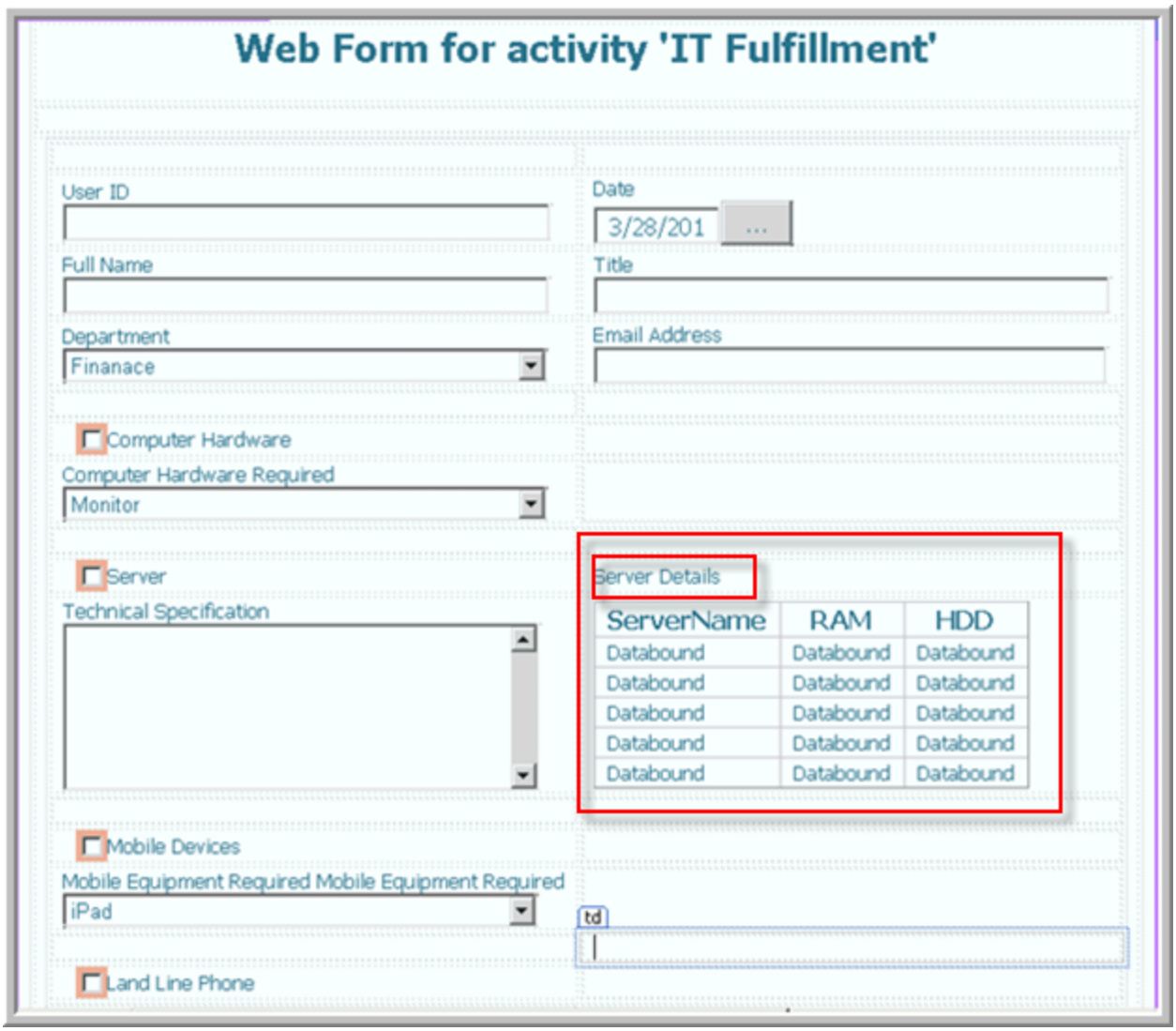
26. Drag and drop **WFGridview** control from the AgilePoint toolbox on the page design area. Go to properties of the control and provide ID as "**gvServer**" and then select **BindingName** property. Select the process template, "**ITRequest**" in your case when prompted to select the schema. Also make **WFGridview** control enabled as "**False**".



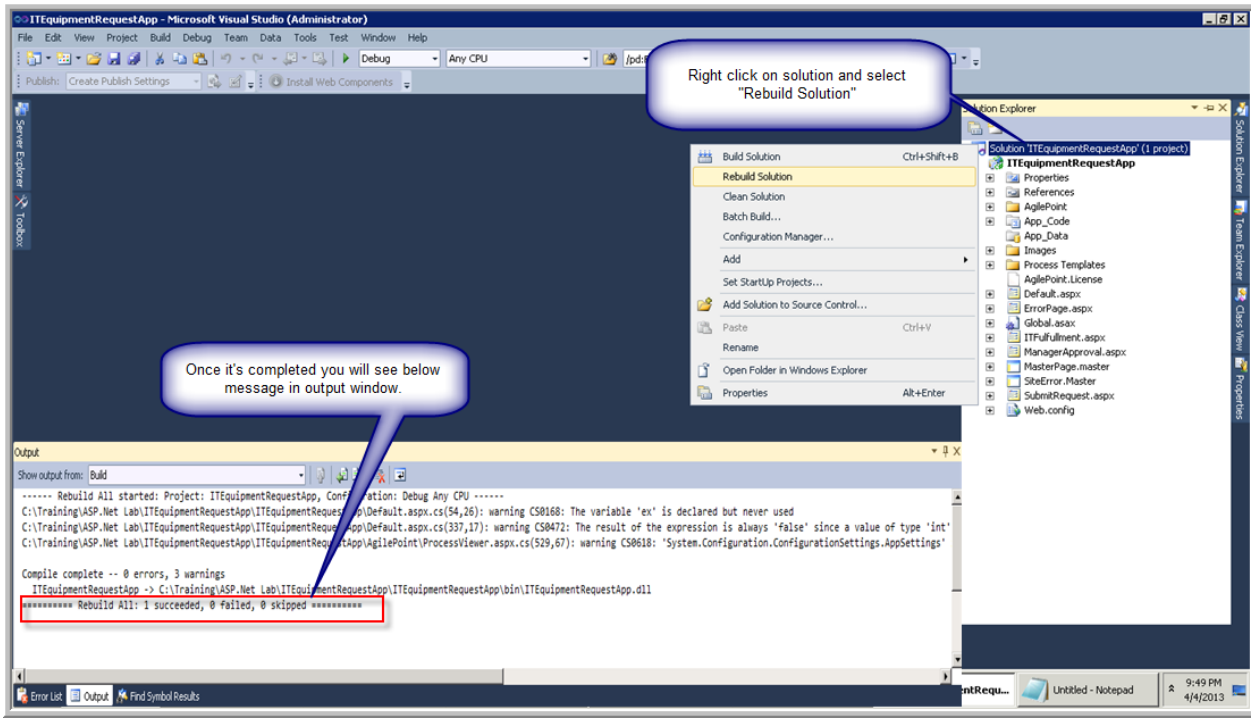
27. Once **WFGridview** control has been added to the page and all settings are configured then provide title of the control as "**Server Details**" on top of the **WFGridview** control. The page would be look like below.



28. Repeat the same steps to add read-only WFGridView to "**ITFulfillment.aspx**" page.

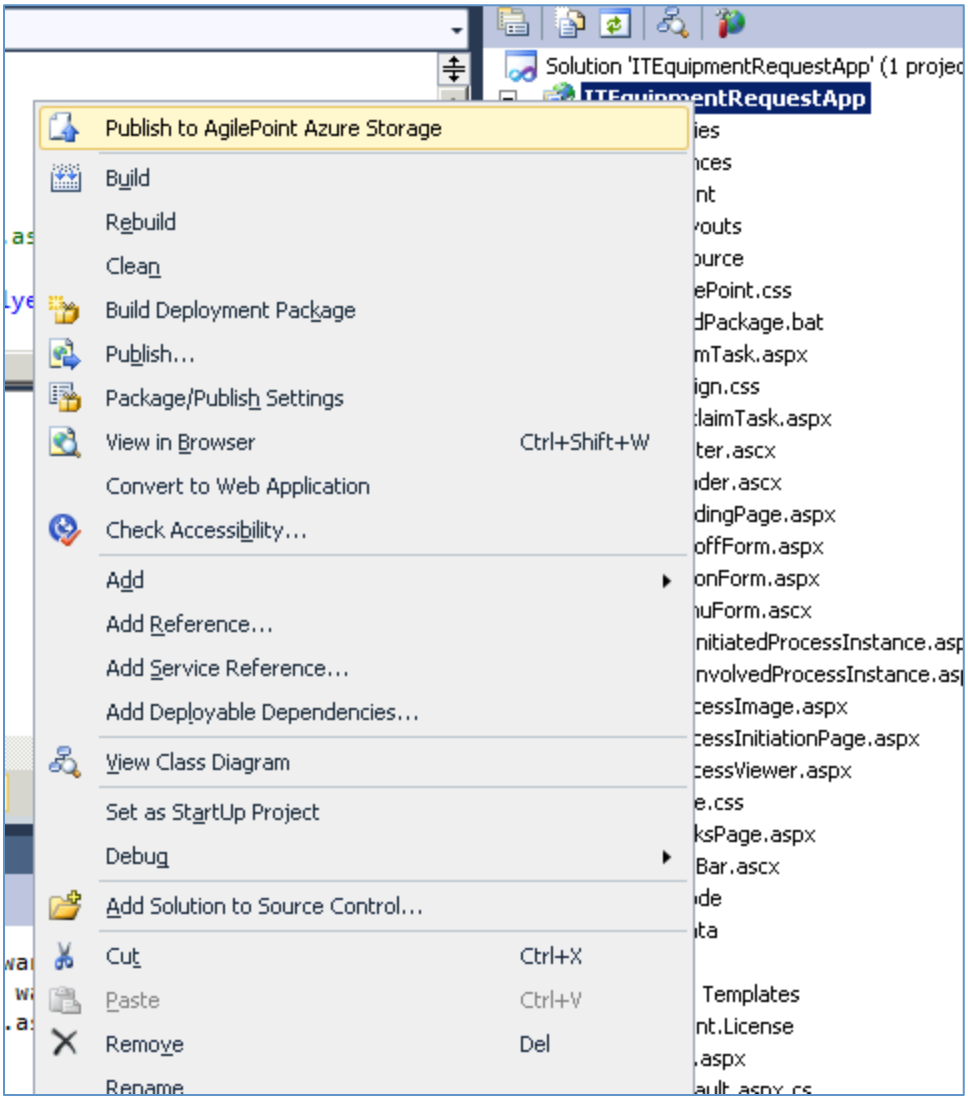


29. Build the web application. (***Shown below.)***

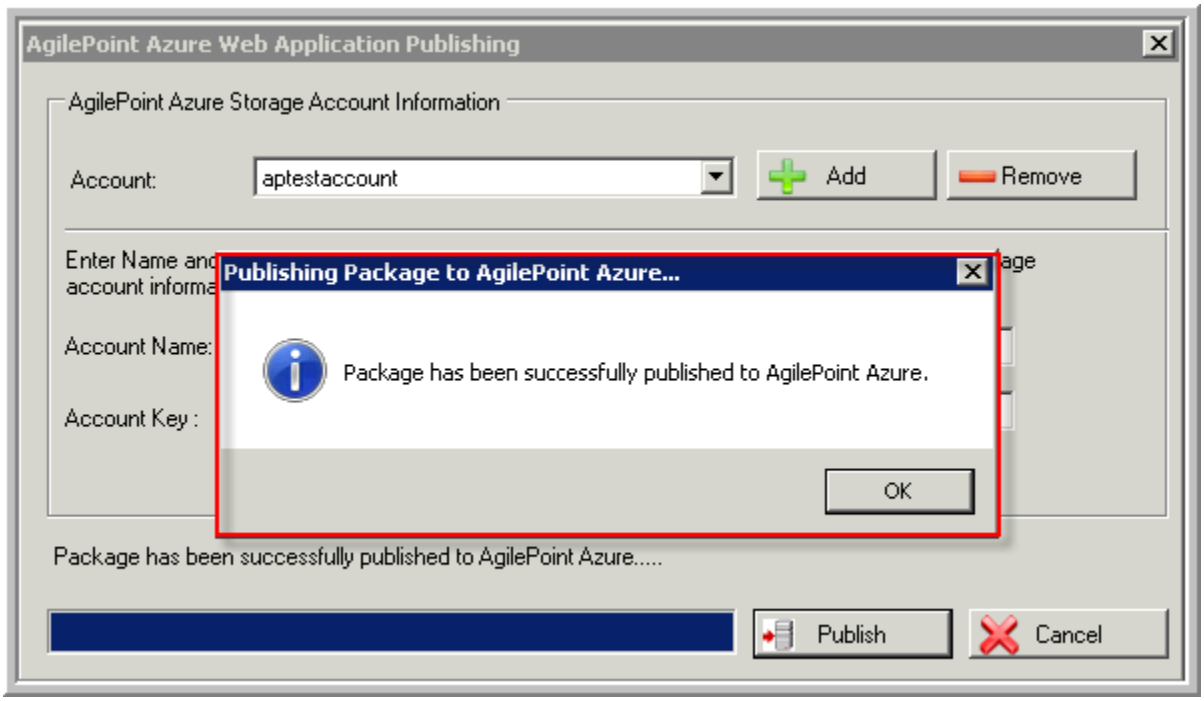


30. If the build operation is successful, you are ready to **re-publish** the web application to your Azure environment. The following are the steps to **re-publishing** the web application.

31. Right click on project in solution explorer and select "**Publish to AgilePoint Azure Storage**"

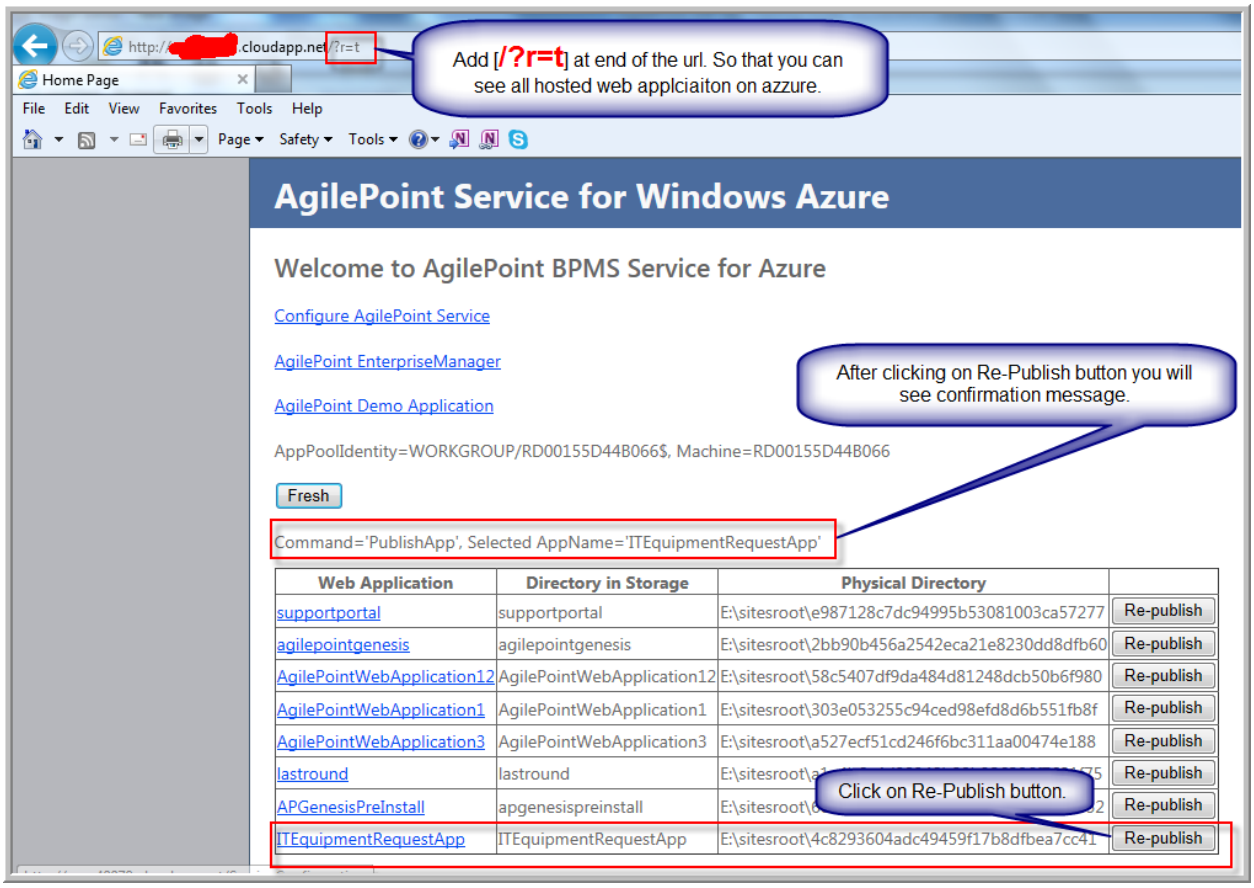


32. It will bring up Web Application Publishing dialog, there you can select the previously saved azure account details and re-publish the web application.



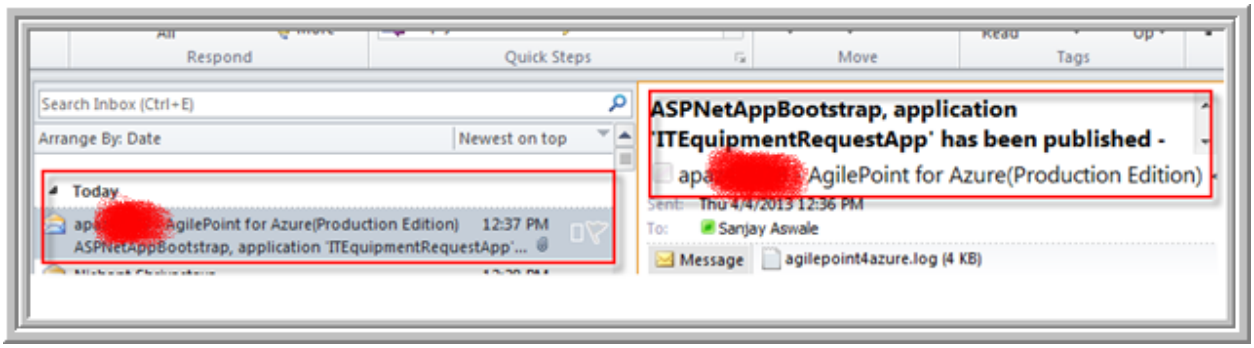
33. Since this is the second time you are publishing the Web Application to storage account, there is no need to restart the WebRole. AgilePoint provides a special mechanism to make your new changes available without having to restart the WebRole. After re-publishing your web application to the azure storage, go to your AgilePoint for Azure home page And change the URL instead of http://apazXXXXX.cloudapp.net/HomePage/ To http://apazXXXXX.cloudapp.net/?r=t

It would list down all the web applications available in your Azure WebRole. Click on "**Re-Publish**" button next to your web application**.** *(In this tutorial our web application is "**ITEquipmentRequestApp**".).* This step usually takes 1-2 sec only. Please refer to below screenshot.



Note: In the upcoming release, this page has been moved to a secured area which can be accessed by admins only.

34. Once you re-publish the web application you will receive email notification regarding your web application being republished in Azure WebRole.

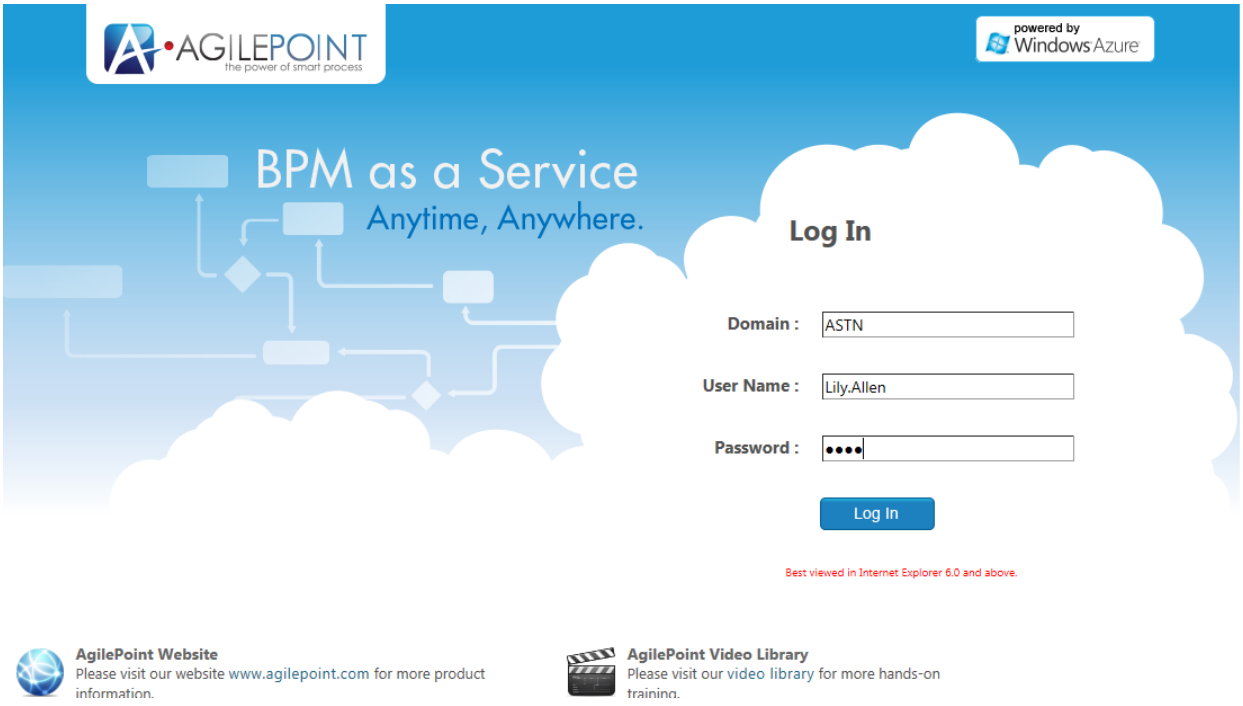


Running Re-Published application

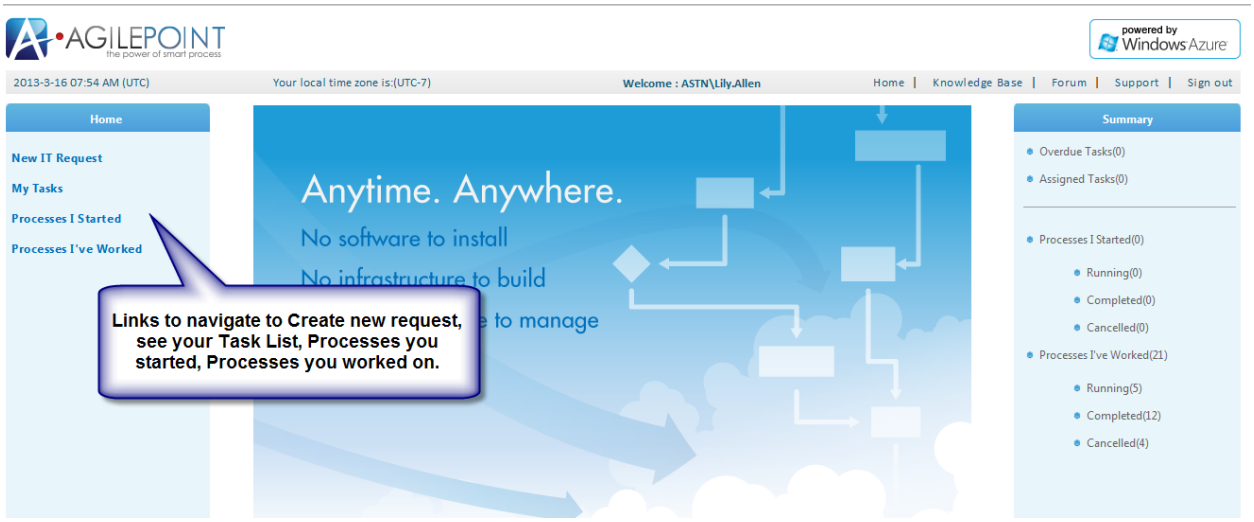
1. Once publish completed you can browse your application and start using it right away.

   The application's URL will be:
   http://[your dns name].cloudapp.net/ITEquipmentRequestApp/Default.aspx
   e.g. http://apaz12345.cloudapp.net/ITEquipmentRequestApp/Default.aspx

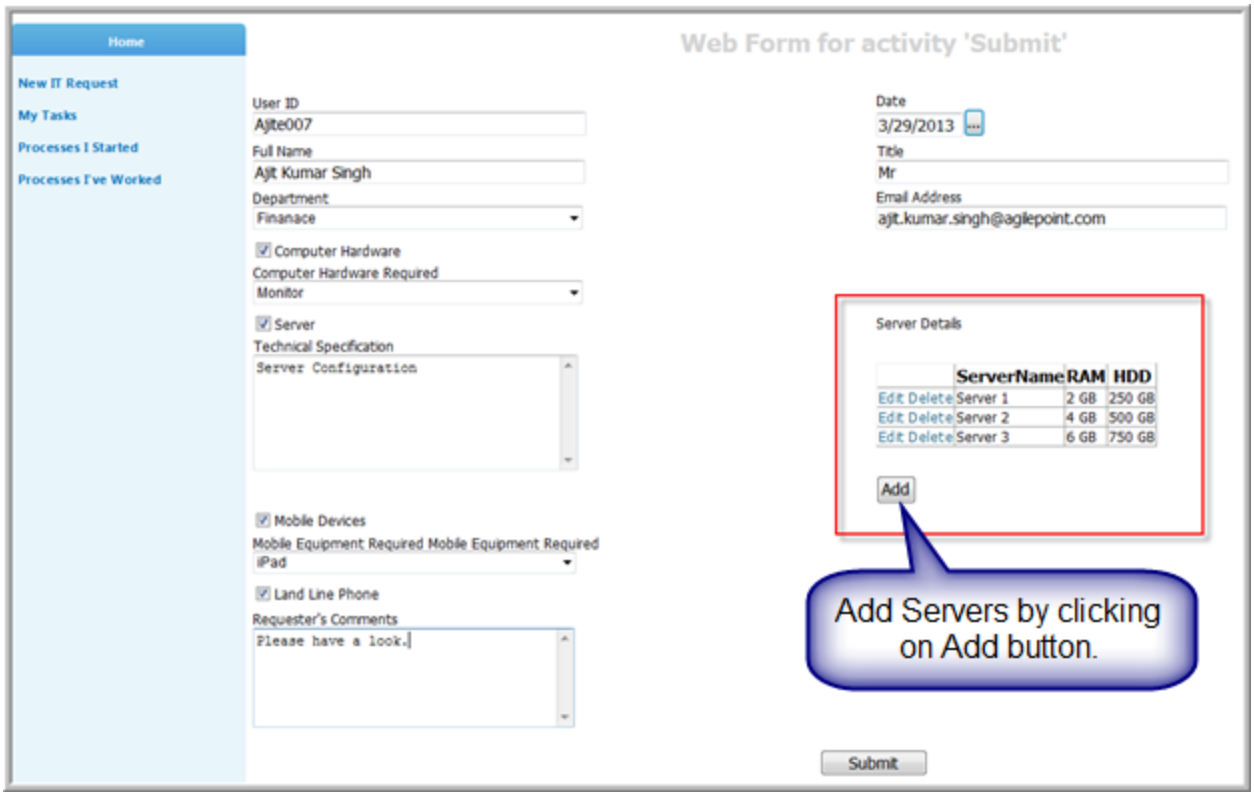
2. The evaluation version of AgilePoint Server on Azure is working against a temporary Domain called ASTN and dummy users are already created in the AgilePoint Server automatically when AgilePoint Server Hosted Service is created. This can be changed for actual production application to client specific domains and active directory connection. Following are the users and their passwords, you can log in your web application using any of these users:
   6. ASTN\Lily.Allen - pass
   7. ASTN\Bob.Hope - pass
   8. ASTN\Bryan.Ferry - pass
   9. ASTN\Michael.Jones - pass
   10. ASTN\Grace.Kelly - pass



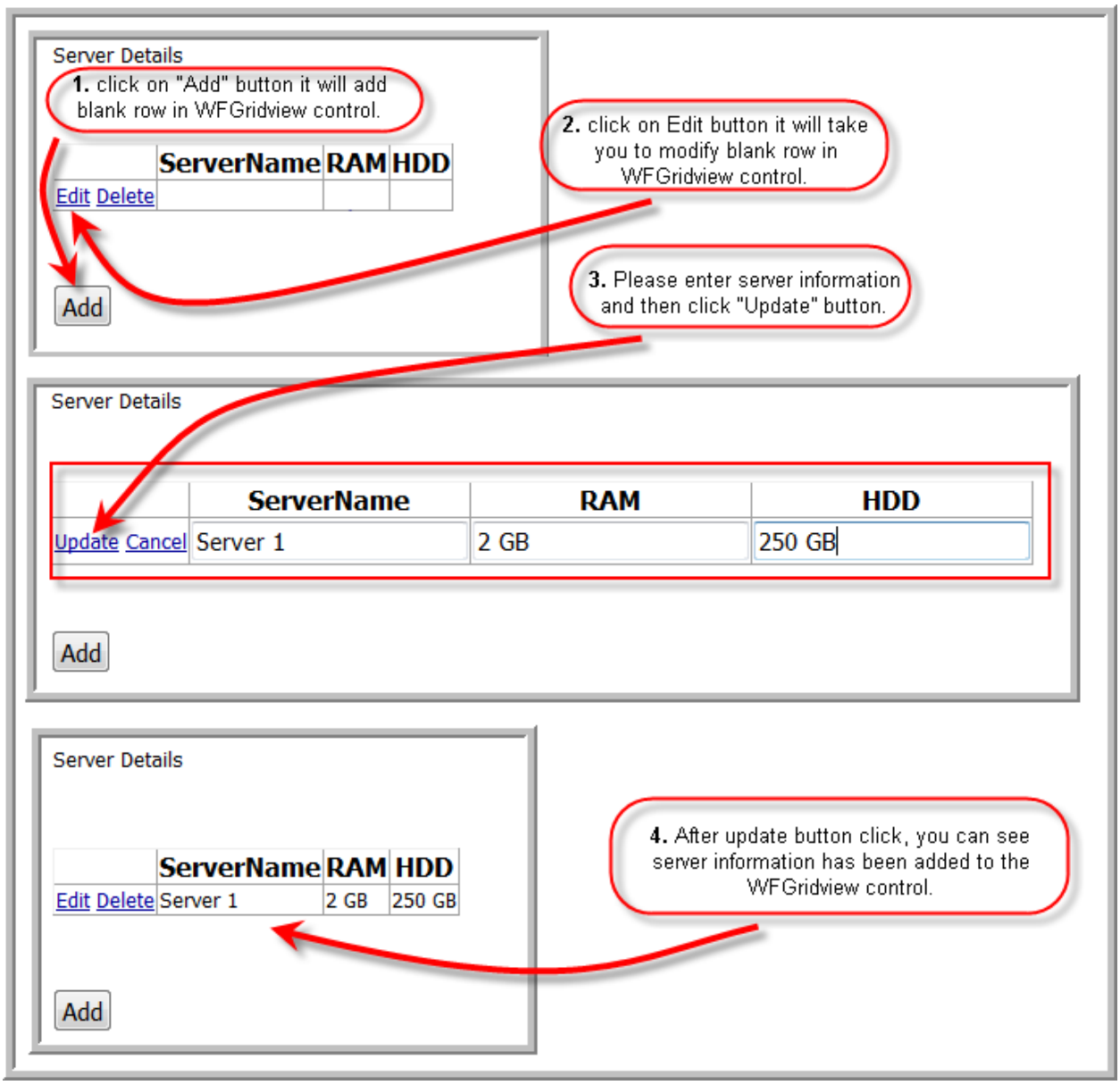
3. After Login, click on "**New IT Request**" link to create new request.



4. You will see the page we designed earlier. Fill in the information and click on Submit.

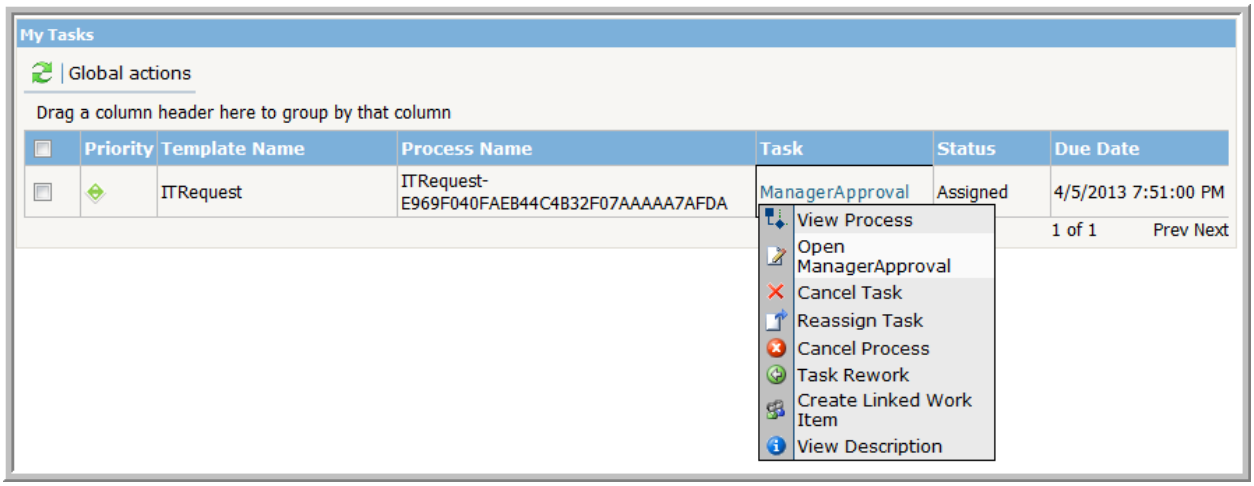


NOTE: Instruction for adding rows in WFGridview control. Please look at the following screenshot.



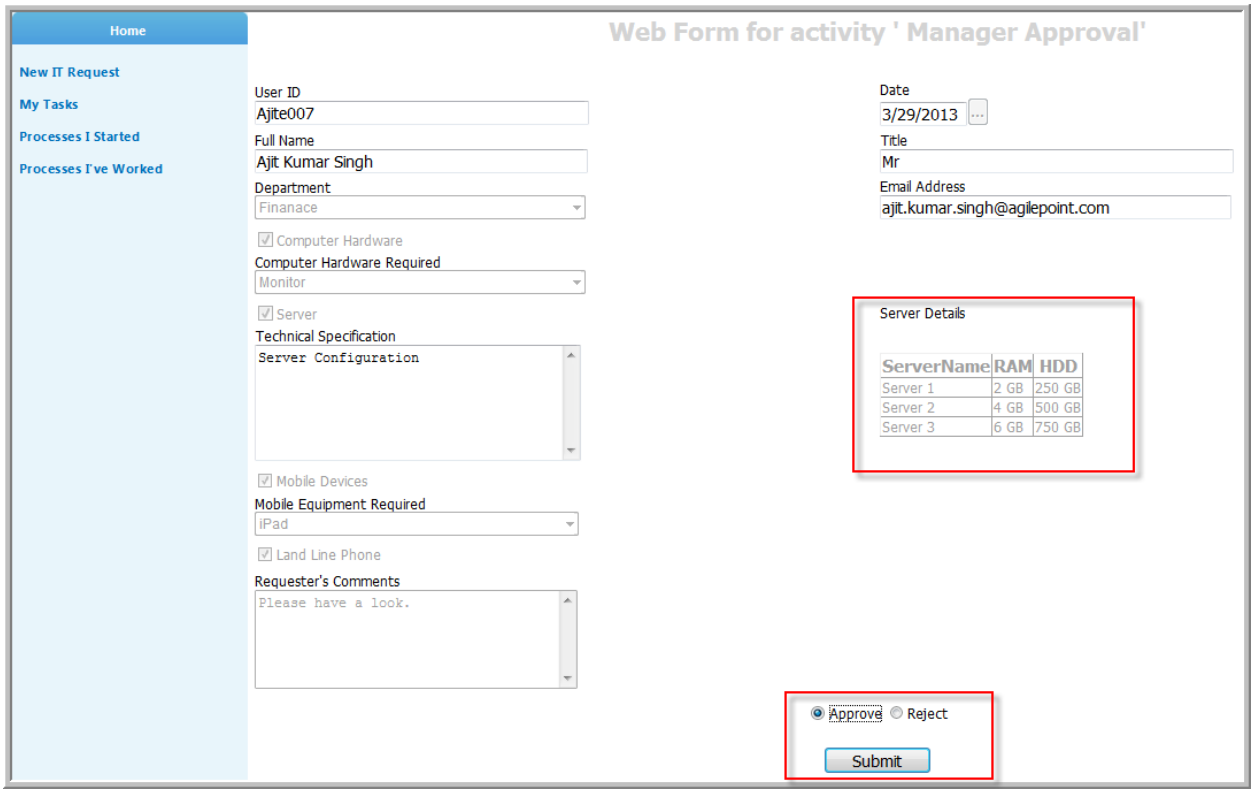
5. After you request submission, it will take you to your task list page where you will see a new task for Manager Approval.

6. Click on the task link to show up available actions you can perform on the task.

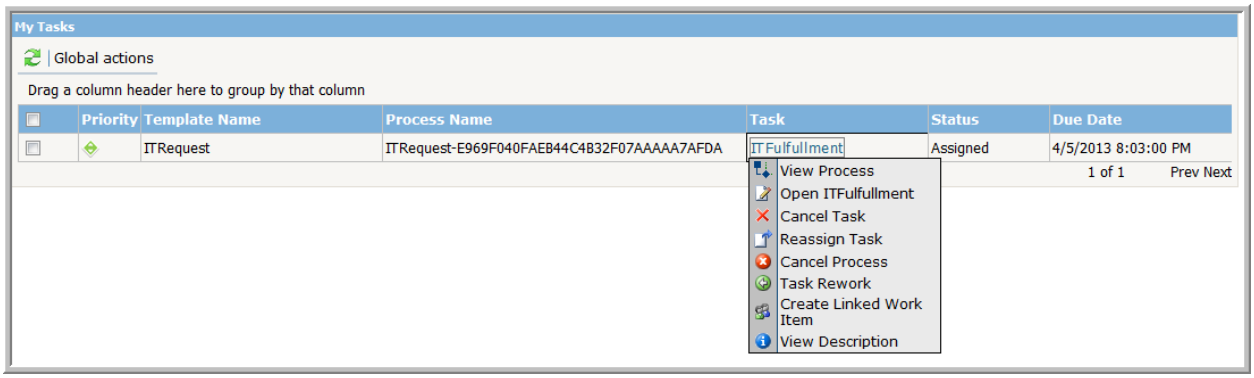


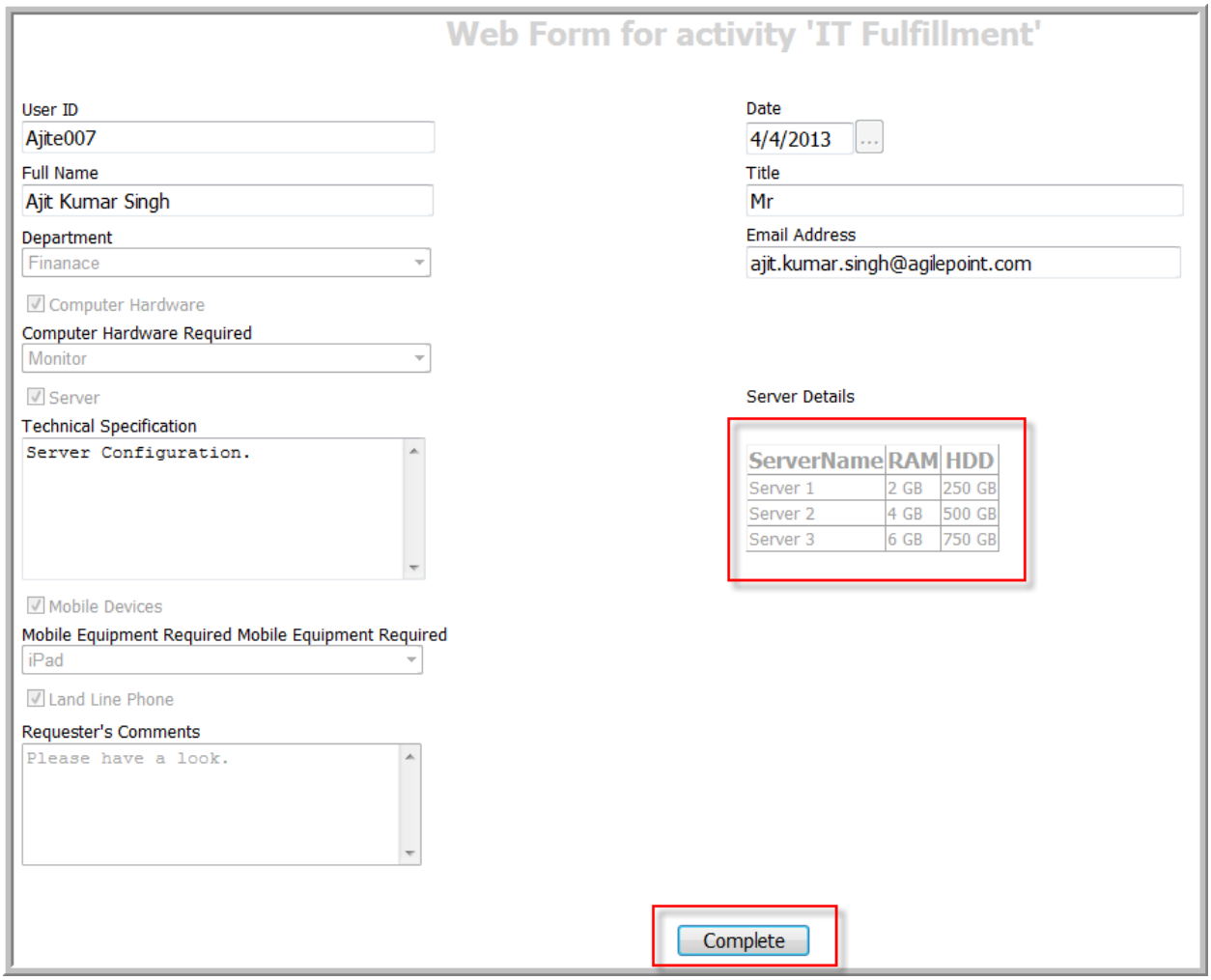
7. Next we will complete the manager approval task. Since in the initial process design we selected same user to be the participant of all 3 manual steps you do not need to log off. This can be changed to unique users or roles after you execute simple demo successfully.

8. Click on "**Open ManagerApproval**" to open the task. You can see all information entered while submitting new request is automatically available in approval screen. This is because of the Binding Name property on each control that ties the data between process and UI.



9. Select Approve radio button and click on Submit.
10. Navigate back to task list, open the ITFulfillment task, and from ITFulfillment page click the Complete button to finish the process.





Summary

ASP.NET development with AgilePoint process is similar to the traditional web development except lot less coding efforts. AgilePoint Envision has provided “Generic” process template for ASP.Net and any other third party form technology. The Generic process template has schema to be defined, which can be used seamlessly for integration with ASP.Net web site using BindingName property on AgilePoint controls. This minimizes lots of plumbing required to make process workflow work. Also, concept of sub process makes cross form technology work together and provides the reusability of process across the server/farm.