ASP.NET Web Deployment using Visual Studio: Deploying Extra Files

docs.microsoft.com/en-us/aspnet/web-forms/overview/deployment/visual-studio-web-deployment/deploying-extra-files

by Tom Dykstra

Download Starter Project

> *This tutorial series shows you how to deploy (publish) an ASP.NET web application to Azure App Service Web Apps or to a third-party hosting provider, by using Visual Studio 2012 or Visual Studio 2010. For information about the series, see the first tutorial in the series.*

Overview

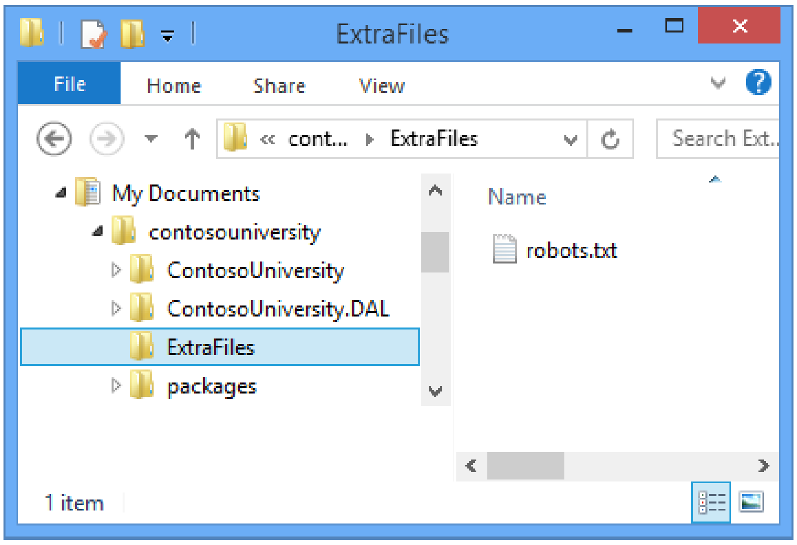
This tutorial shows how to extend the Visual Studio web publish pipeline to do an additional task during deployment. The task is to copy extra files that are not in the project folder to the destination web site.

For this tutorial you'll copy one extra file: *robots.txt*. You want to deploy this file to staging but not to production. In the Deploying to Production tutorial, you added this file to the project and configured the Production publish profile to exclude it. In this tutorial you'll see an alternative method to handle this situation, one that will be useful for any files that you want to deploy but don't want to include in the project.

Move the robots.txt file

To prepare for a different method of handling *robots.txt*, in this section of the tutorial you move the file to a folder that is not included in the project, and you delete *robots.txt* from the staging environment. It is necessary to delete the file from staging so that you can verify that your new method of deploying the file to that environment is working correctly.

1. In **Solution Explorer**, right-click the *robots.txt* file and click **Exclude From Project**.
2. Using Windows File Explorer, create a new folder in the solution folder and name it *ExtraFiles*.
3. Move the *robots.txt* file from the *ContosoUniversity* project folder to the *ExtraFiles* folder.



4. Using your FTP tool, delete the *robots.txt* file from the staging web site.

   As an alternative, you can select **Remove additional files at destination** under **File Publish Options** on the **Settings** tab of the Staging publish profile, and republish to staging.

Update the publish profile file

You only need *robots.txt* in staging, so the only publish profile you need to update in order to deploy it is Staging.

1. In Visual Studio, open *Staging.pubxml*.

2. At the end of the file, before the closing `</Project>` tag, add the following markup:

   XML

   ```
   <Target Name="CustomCollectFiles">
       <ItemGroup>
         <_CustomFiles Include="..\ExtraFiles\**\*" />
         <FilesForPackagingFromProject Include="%(_CustomFiles.Identity)">
           <DestinationRelativePath>%(RecursiveDir)%(Filename)%
   (Extension)</DestinationRelativePath>
         </FilesForPackagingFromProject>
       </ItemGroup>
     </Target>
   ```

   This code creates a new *target* that will collect additional files to be deployed. A target is composed of one or more tasks that MSBuild will execute based on conditions you specify.

   The `Include` attribute specifies that the folder in which to find the files is *ExtraFiles*, located at the same level as the project folder. MSBuild will collect all files from that folder and recursively from any subfolders (the double asterisk specifies recursive subfolders). With this code you could put multiple files, and files in subfolders inside the *ExtraFiles* folder, and all will be deployed.

   The `DestinationRelativePath` element specifies that the folders and files should be copied to the root folder of the destination web site, in the same file and folder structure as they are found in the *ExtraFiles* folder. If you wanted to copy the *ExtraFiles* folder itself, the `DestinationRelativePath` value would be *ExtraFiles\%(RecursiveDir)%(Filename)%(Extension)*.

3. At the end of the file, before the closing `</Project>` tag, add the following markup that specifies when to execute the new target.

   XML

   ```
   <PropertyGroup>
       <CopyAllFilesToSingleFolderForPackageDependsOn>
         CustomCollectFiles;
         $(CopyAllFilesToSingleFolderForPackageDependsOn);
       </CopyAllFilesToSingleFolderForPackageDependsOn>

       <CopyAllFilesToSingleFolderForMsdeployDependsOn>
         CustomCollectFiles;

   $(CopyAllFilesToSingleFolderForMsdeployDependsOn);
       </CopyAllFilesToSingleFolderForMsdeployDependsOn>
   </PropertyGroup>
   ```

   This code causes the new `CustomCollectFiles` target to be executed whenever the target that copies files to the destination folder is executed. There is a separate target for publish versus deployment package creation, and the new target is injected in both targets in case you decide to deploy by using a deployment package instead of publishing.

   The *.pubxml* file now looks like the following example:

   XML

   ```
   <?xml version="1.0" encoding="utf-8"?>

   <Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
     <PropertyGroup>
       <WebPublishMethod>MSDeploy</WebPublishMethod>
       <LastUsedBuildConfiguration>Release</LastUsedBuildConfiguration>
   ```

```
    <LastUsedPlatform>Any CPU</LastUsedPlatform>
    <SiteUrlToLaunchAfterPublish>http://contosou-
staging.azurewebsites.net</SiteUrlToLaunchAfterPublish>
    <ExcludeApp_Data>True</ExcludeApp_Data>
    <MSDeployServiceURL>waws-prod-bay-
001.publish.azurewebsites.windows.net:443</MSDeployServiceURL>
    <DeployIisAppPath>contosou-staging</DeployIisAppPath>
    <RemoteSitePhysicalPath />
    <SkipExtraFilesOnServer>False</SkipExtraFilesOnServer>
    <MSDeployPublishMethod>WMSVC</MSDeployPublishMethod>
    <UserName>$contosou-staging</UserName>
    <_SavePWD>True</_SavePWD>
    <PublishDatabaseSettings>
      <Objects xmlns="">
        <ObjectGroup Name="SchoolContext" Order="1" Enabled="True">
          <Destination Path="Data Source=tcp:sk0264hvc9.database.windows.net,1433;Initial
Catalog=ContosoUniversity-staging;User ID=CU-staging-admin@sk0264hvc9;Password=Pas$w0rd"
Name="Data Source=tcp:sk0264hvc9.database.windows.net,1433;Initial
Catalog=ContosoUniversity-staging;User Id=CU-staging-admin@sk0264hvc9;Password=Pas$w0rd"
/>
          <Object Type="DbCodeFirst">
            <Source Path="DBMigration" DbContext="ContosoUniversity.DAL.SchoolContext,
ContosoUniversity.DAL"
MigrationConfiguration="ContosoUniversity.DAL.Migrations.Configuration,
ContosoUniversity.DAL" Origin="Configuration" />
          </Object>
        </ObjectGroup>
        <ObjectGroup Name="DefaultConnection" Order="2" Enabled="False">
          <Destination Path="Data Source=tcp:sk0264hvc9.database.windows.net,1433;Initial
Catalog=ContosoUniversity-staging;User ID=CU-staging-admin@sk0264hvc9;Password=Pas$w0rd"
Name="Data Source=tcp:sk0264hvc9.database.windows.net,1433;Initial
Catalog=ContosoUniversity-staging;User Id=CU-staging-admin@sk0264hvc9;Password=Pas$w0rd"
/>
          <Object Type="DbDacFx">
            <PreSource Path="Data Source=
(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\aspnet-ContosoUniversity.mdf;Initial
Catalog=aspnet-ContosoUniversity;Integrated Security=True" includeData="False" />
            <Source
Path="$(IntermediateOutputPath)AutoScripts\DefaultConnection_IncrementalSchemaOnly.dacpac"
dacpacAction="Deploy" />
          </Object>
          <UpdateFrom Type="Web.Config">
            <Source MatchValue="Data Source=(LocalDb)\v11.0;Integrated
Security=SSPI;Initial Catalog=aspnet-
ContosoUniversity;AttachDBFilename=|DataDirectory|\aspnet-ContosoUniversity.mdf"
MatchAttributes="$(UpdateFromConnectionStringAttributes)" />
          </UpdateFrom>
          <Object Type="DbFullSql" Enabled="False">
            <Source Path="..\aspnet-data-prod.sql" Transacted="False" />
          </Object>
        </ObjectGroup>
      </Objects>
    </PublishDatabaseSettings>
    <EnableMSDeployBackup>False</EnableMSDeployBackup>
  </PropertyGroup>
  <ItemGroup>
    <MSDeployParameterValue Include="$(DeployParameterPrefix)DefaultConnection-Web.config
Connection String">
      <ParameterValue>Data Source=tcp:sk0264hvc9.database.windows.net,1433;Initial
Catalog=ContosoUniversity-staging;User Id=CU-staging-
admin@sk0264hvc9;Password=Pas$w0rd</ParameterValue>
    </MSDeployParameterValue>
    <MSDeployParameterValue Include="$(DeployParameterPrefix)SchoolContext-Web.config
Connection String">
      <ParameterValue>Data Source=tcp:sk0264hvc9.database.windows.net,1433;Initial
Catalog=ContosoUniversity-staging;User Id=CU-staging-
admin@sk0264hvc9;Password=Pas$w0rd</ParameterValue>
```

```
      </MSDeployParameterValue>
    </ItemGroup>
    <Target Name="CustomCollectFiles">
      <ItemGroup>
        <_CustomFiles Include="..\ExtraFiles\**\*" />
        <FilesForPackagingFromProject Include="%(_CustomFiles.Identity)">
          <DestinationRelativePath>%(RecursiveDir)%(Filename)%
(Extension)</DestinationRelativePath>
        </FilesForPackagingFromProject>
      </ItemGroup>
    </Target>
    <PropertyGroup>
      <CopyAllFilesToSingleFolderForPackageDependsOn>
        CustomCollectFiles;
        $(CopyAllFilesToSingleFolderForPackageDependsOn);
      </CopyAllFilesToSingleFolderForPackageDependsOn>

      <CopyAllFilesToSingleFolderForMsdeployDependsOn>
        CustomCollectFiles;
        $(CopyAllFilesToSingleFolderForPackageDependsOn);
      </CopyAllFilesToSingleFolderForMsdeployDependsOn>
    </PropertyGroup>
  </Project>
```

4. Save and close the *Staging.pubxml* file.

Publish to staging

Using one-click publish or the command line, publish the application by using the Staging profile.

If you use one-click publish, you can verify in the **Preview** window that *robots.txt* will be copied. Otherwise, use your FTP tool to verify that the *robots.txt* file is in the root folder of the web site after deployment.

Summary

This completes this series of tutorials on deploying an ASP.NET web application to a third-party hosting provider. For more information about any of the topics covered in these tutorials, see the ASP.NET Deployment Content Map.

More information

If you know how to work with MSBuild files, you can automate many other deployment tasks by writing code in *.pubxml* files (for profile-specific tasks) or the project *.wpp.targets* file (for tasks that apply to all profiles). For more information about *.pubxml* and *.wpp.targets* files, see How to: Edit Deployment Settings in Publish Profile (.pubxml) Files and the .wpp.targets File in Visual Studio Web Projects. For a basic introduction to MSBuild code, see **The Anatomy of a Project File** in Enterprise Deployment Series: Understanding the Project File. To learn how to work with MSBuild files to perform tasks for your own scenarios, see this book: Inside the Microsoft Build Engine: Using MSBuild and Team Foundation Build by Sayed Ibraham Hashimi and William Bartholomew.

Acknowledgements

I would like to thank the following people who made significant contributions to the content of this tutorial series:

- Alberto Poblacion, MVP & MCT, Spain
- Jarod Ferguson, Data Platform Development MVP, United States
- Harsh Mittal, Microsoft
- Jon Galloway (twitter: @jongalloway)
- Kristina Olson, Microsoft
- Mike Pope, Microsoft
- Mohit Srivastava, Microsoft
- Raffaele Rialdi, Italy

- Rick Anderson, Microsoft
- Sayed Hashimi, Microsoft(twitter: @sayedihashimi)
- Scott Hanselman (twitter: @shanselman)
- Scott Hunter, Microsoft (twitter: @coolcsh)
- Srđan Božović, Serbia
- Vishal Joshi, Microsoft (twitter: @vishalrjoshi)

Previous Next