

[DOCS](#)[CONTACT SALES](#)[Web](#)[Choose a Page](#)[GETTING STARTED](#)[WEB INTEGRATIONS](#)

Getting started on Web Integrations

[SHAREPOINT](#)[ON-PREMISES](#)[NEW PROJECT](#)

Integrate into SharePoint On-Premises as a File Handler

This guide will walk you through the steps necessary to integrate PSPDFKit for Web into SharePoint on-premises environments as a file handler.

File handlers are simple XML files that instruct SharePoint to redirect to your URL of choice whenever a file with a particular extension is clicked. By using a file handler, you'll override SharePoint's default behavior so that PDF documents stored in SharePoint are opened directly inside your application. Any edits to the opened files using your application can then be directly saved back to SharePoint.

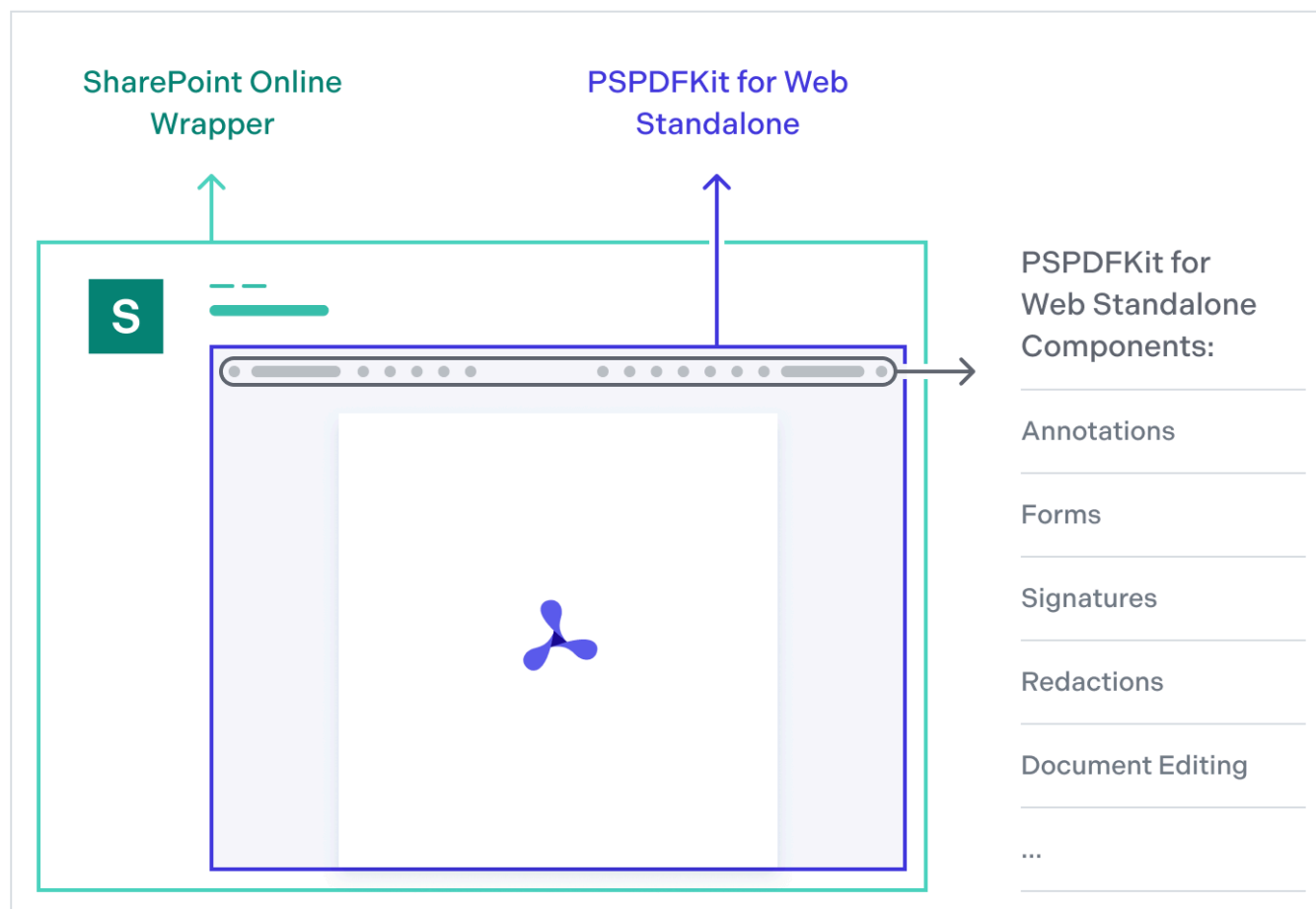
PSPDFKit for SharePoint is a wrapper on top of PSPDFKit for Web Standalone, which is a client-side JavaScript library for viewing and editing PDF documents directly in a web browser.

[ASK AI](#)

PSPDFKit for SharePoint shares the same APIs as PSPDFKit for Web Standalone, so use the Web documentation when customizing your SharePoint application.

TRY OUR NO-CODE APP

PSPDFKit for SharePoint



Requirements

Hardware and operating system requirements are determined by the requirements of the SharePoint version you're targeting, as well as the requirements of the Visual Studio version used for development. Both SharePoint Server and Visual Studio have to be installed on the developer machine.

The required Visual Studio versions for targeting different SharePoint versions are summarized in the following table. Up to Visual Studio 2017, the matching Microsoft Office Developer Tools needs to be manually downloaded and installed. For Visual Studio 2019 and 2022, the relevant components can be selected during installation of Visual Studio.

Development tool	Targeted SharePoint version			
	SP2013	SP2016	SP2019	SPSE
Visual Studio 2012 with Microsoft Office Developer Tools for Visual Studio 2012	V	X	X	X
Visual Studio 2013 with Microsoft Office Developer Tools for Visual Studio 2013	V	X	X	X
Visual Studio 2015 with Microsoft Office Developer Tools for Visual Studio 2015	V	V	X	X
Visual Studio 2017 with Microsoft Office Developer Tools for Visual Studio 2017	V	V	X	X
Visual Studio 2019 with Office Developer Tools for Visual Studio component	V	V	V	X
Visual Studio 2022 with Office Developer Tools for Visual Studio component	V	V	V	V

SharePoint Permissions Considerations

To develop SharePoint solutions, you must have sufficient permissions to run and debug them. Before you can test a SharePoint solution, take the following steps to ensure you have the necessary permissions:

- 1 Add your user account as an Administrator on the system.
- 2 Add your user account as a Farm Administrator for the SharePoint server.

In SharePoint Central Administration, click the Manage the farm administrators group link.

On the Farm Administrators page, click the New button.

- 3 Add your user account to the `WSS_ADMIN_WPG` group.

You can find more information by clicking the link that corresponds to your version of Visual Studio:

Visual Studio 2012

Visual Studio 2013

Visual Studio 2017

Visual Studio 2019

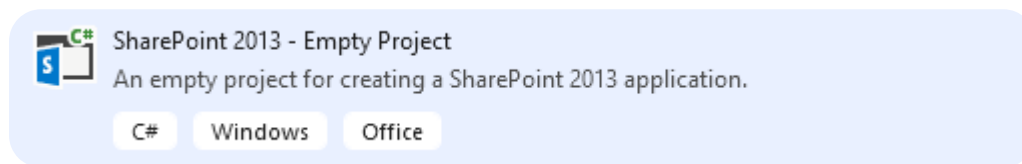
Visual Studio 2022

1 Creating a New Project

A new project has to be created in Visual Studio. The project can then be deployed to and debugged in your selected SharePoint environment. The project can also be published to create an installable SharePoint solution package (.wsp).

The example here uses Visual Studio 2019, but SharePoint project templates are also available in earlier versions (minimum Visual Studio 2013).

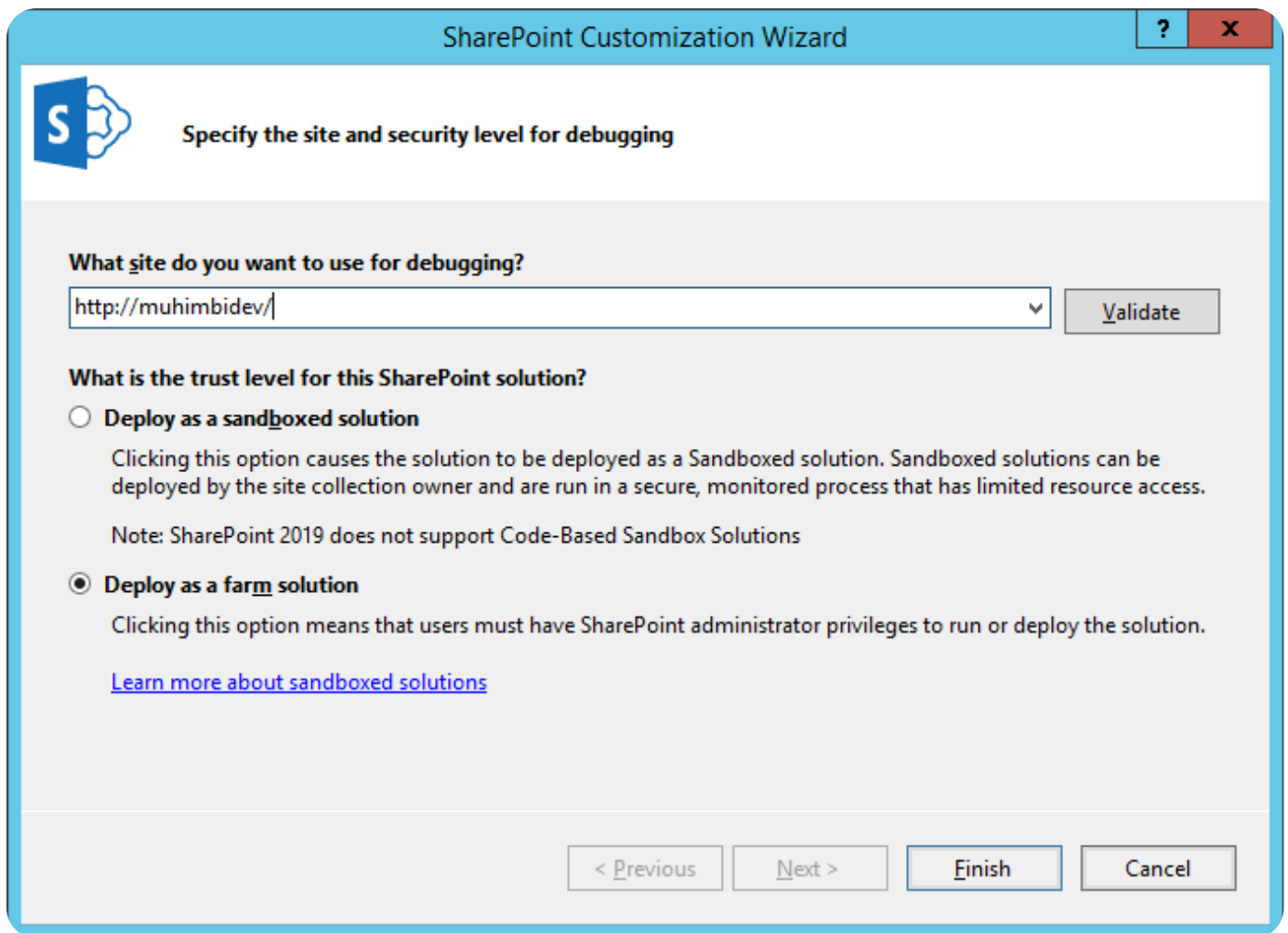
- 1 Create a new empty SharePoint project matching the desired SharePoint version (SharePoint 2013, SharePoint 2016, or SharePoint 2019).



- 2 Give your project the name `OpenPDFSharePoint` , set its location on your disk, select the desired .NET Framework version, and click Create.

A screenshot of the 'Configure your new project' dialog box in Visual Studio. The dialog has a title bar with a maximize button and a close button. The main title is 'Configure your new project'. Below the title, there are three tabs: 'SharePoint 2013 - Empty Project' (selected), 'C#', 'Windows', and 'Office'. The 'Project name' field contains 'OpenPDFSharePoint'. The 'Location' field is a dropdown menu showing 'C:\repos' with a folder icon button to its right. The 'Solution name' field contains 'OpenPDFSharePoint'. There is an unchecked checkbox labeled 'Place solution and project in the same directory'. The 'Framework' dropdown menu shows '.NET Framework 4.7.2'. At the bottom right, there are two buttons: 'Back' and 'Create'.

- 3 Select your development SharePoint site for debugging, and pick Deploy as farm solution. Click Finish. This will create the skeleton of your project.



SharePoint Customization Wizard

Specify the site and security level for debugging

What site do you want to use for debugging?

What is the trust level for this SharePoint solution?

☐ Deploy as a sandboxed solution

Clicking this option causes the solution to be deployed as a Sandboxed solution. Sandboxed solutions can be deployed by the site collection owner and are run in a secure, monitored process that has limited resource access.

Note: SharePoint 2019 does not support Code-Based Sandbox Solutions

☒ Deploy as a farm solution

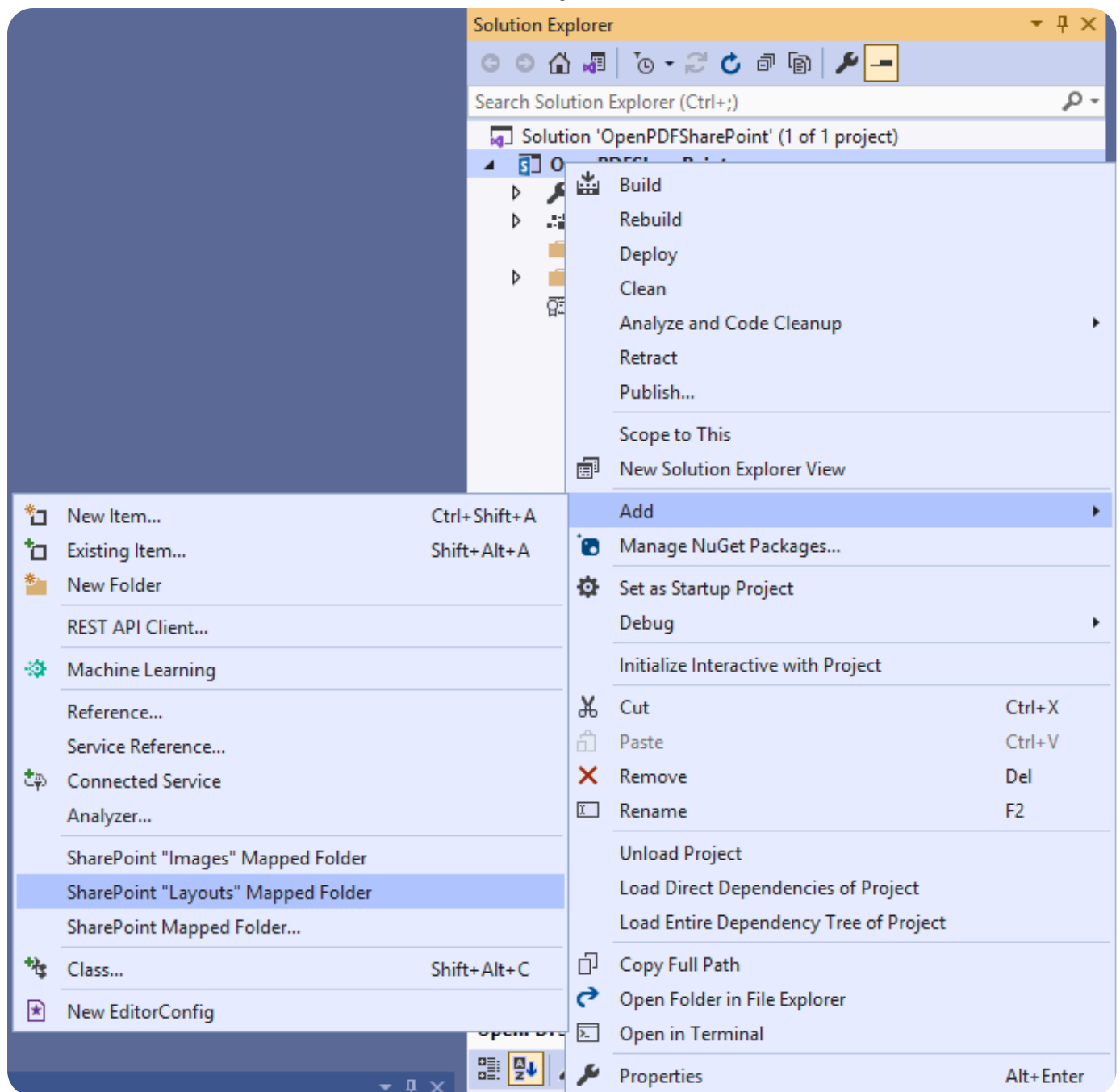
Clicking this option means that users must have SharePoint administrator privileges to run or deploy the solution.

[Learn more about sandboxed solutions](#)

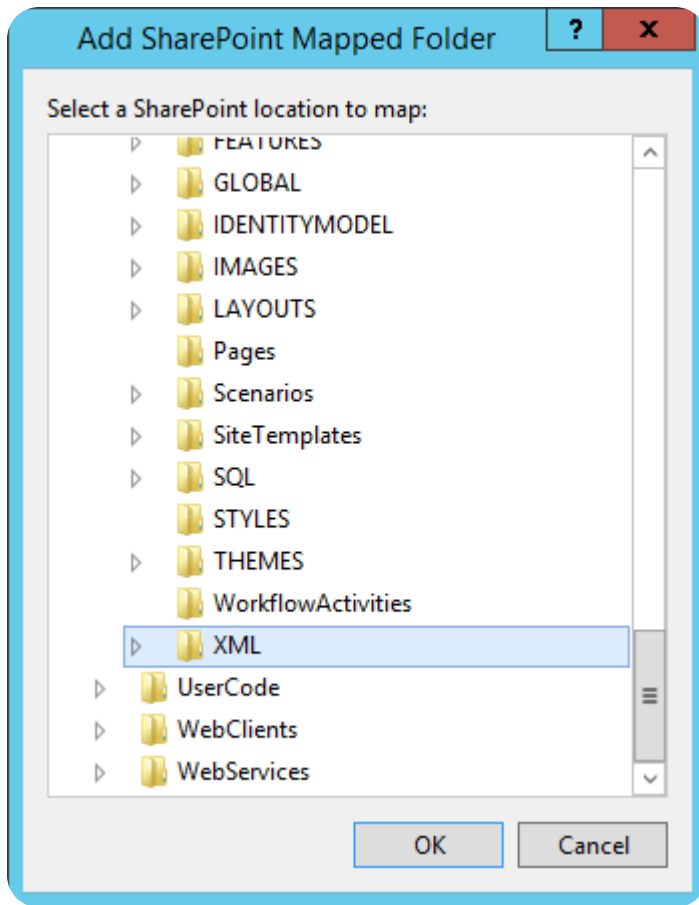
2 Creating the Folder Structure

Some folders in your project have to be created or mapped to SharePoint folders in advance.

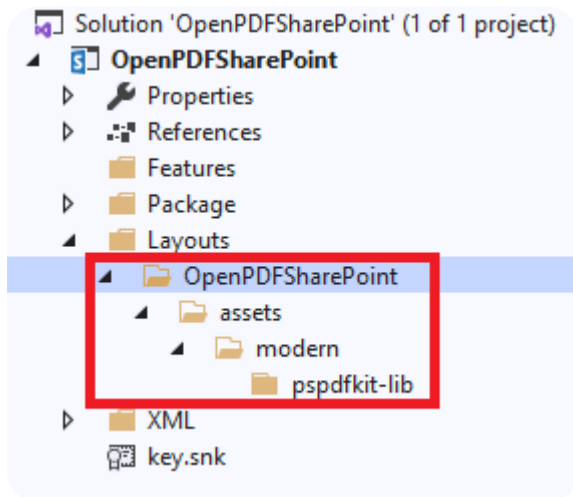
- 1 In the Solution Explorer window, right-click the `OpenPDFSharePoint` project and select Add > SharePoint “Layouts” Mapped Folder. This will create the `Layouts` folder and a folder for your code (`OpenPDFSharePoint`) inside. It also automatically updates the package details.



- 2 Right-click the `OpenPDFSharePoint` project in the Solution Explorer window again and select `Add > SharePoint Mapped Folder`. Select `{SharePointRoot} > TEMPLATE > XML` from the tree structure in the popup dialogue and click OK.



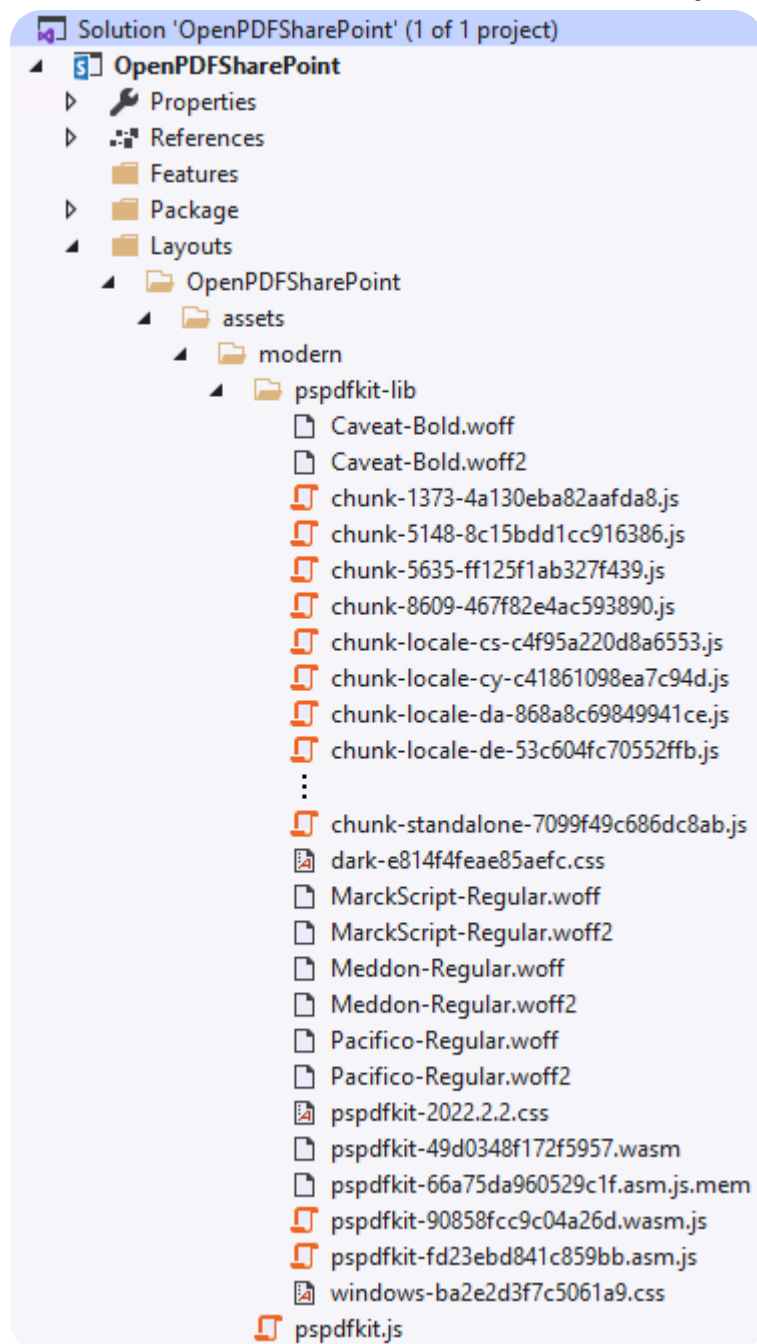
- 3 Using the context menu again (right-click on the newly created `OpenPDFSharePoint` folder inside the `Layouts` folder), select `Add > New Folder`. Create the following nested subfolders inside the `OpenPDFSharePoint` folder: `assets > modern > pspdfkit-lib`.



3 Adding the PSPDFKit Libraries to the Project

The content (the `.wasm` file and the JavaScript, CSS, and font files) mentioned in this next part is the PSPDFKit Web Standalone product. It enables the browser to render and edit PDF files.

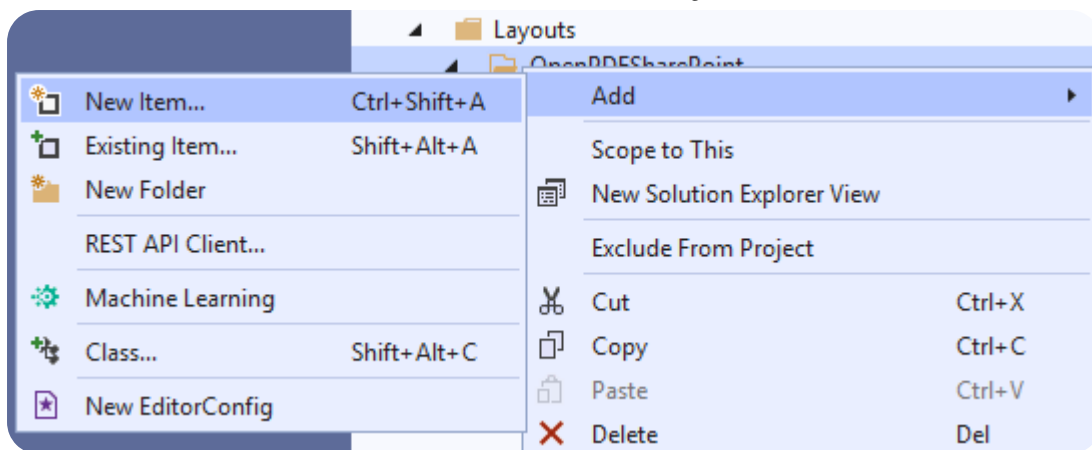
- 1 Download the latest PSPDFKit Web Standalone libraries from <https://my.nutrient.io/download/web/latest>.
- 2 The download will start immediately and will save a `.tar.gz` archive like `PSPDFKit-Web-binary-2022.2.2.tar.gz` to your computer.
- 3 Once the download is complete, extract the archive.
- 4 Right-click on the `Layouts > OpenPDFSharePoint > assets > modern` folder in the Solution Explorer window and select `Add > Existing Item...`
- 5 Navigate to the extracted `package > dist > modern > pspdfkit.js` file, select it, and click `Add`.
- 6 Now, right-click on the `Layouts > OpenPDFSharePoint > assets > modern > pspdfkit-lib` folder in the Solution Explorer window and select `Add > Existing Item...` again.
- 7 Navigate to the extracted `package > dist > modern > pspdfkit-lib` folder, select all the files in the folder, and click `Add`. The result will look like this:



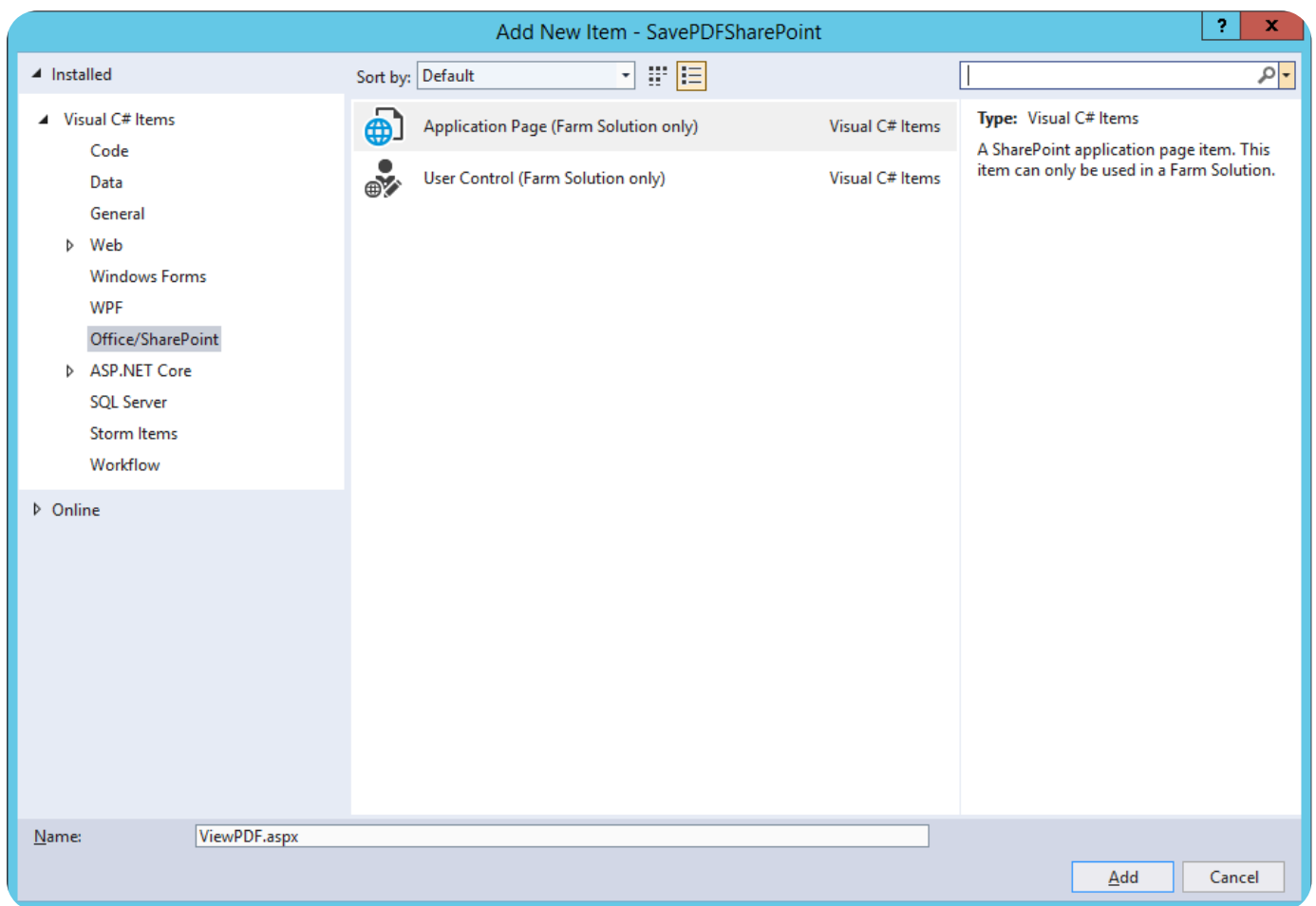
4 Adding a PDF Viewer Page

This SharePoint application page will encapsulate the PSPDFKit Web Standalone component and configure it to display the selected PDF file. It'll also save the changes back into the SharePoint library.

Right-click the `OpenPDFSharePoint` folder inside the `Layouts` folder in the Solution Explorer window and select `Add > New Item`.



Select Application Page (Farm Solution only) and name it `ViewPDF.aspx` . Click Add.



In the `ViewPDF.aspx` file, identify the main content placeholder:

```
1 <asp:Content ID="Main" ContentPlaceholderID="PlaceHolderMain" runat="
2 </asp:Content>
```

You have to add three things inside the main content placeholder.

- 1 Add an empty `<div>` element with a defined width and height to where PSPDFKit will be mounted:

```
<div id="pspdfkit" style="width: 100%; height: calc(100vh - 210px);"></div>
```

- 2 Add a reference to the PSPDFKit JavaScript file:

```
<script type="text/javascript" src="assets/modern/pspdfkit.js"></script>
```

- 3 Add the following script:

```
1 <script type="text/javascript">
2   // !!! Enter your license key here !!!
3   let licenseKey = "paste your license here";
4
5   // Load PDF Editor
6   PSPDFKit.load({
7     container: "#pspdfkit",
8     document: "<%= fileURL %>",
9     licenseKey: licenseKey
10  })
11    .then(function (instance) {
12      console.log("PSPDFKit loaded", instance);
13    })
14    .catch(function (error) {
15      console.error(error.message);
16    });
17 </script>
```

If you have a PSPDFKit license, then paste it into the script (at the beginning) between the quotation marks. Otherwise, let the `licenseKey` variable be an empty string.

⋮

You must explicitly set the type attribute of the script elements to `"text/javascript"`. Otherwise, they'll be ignored.

Modify the `ViewPDF.aspx.cs` code behind the file to include the `fileURL` public property and some new code in the `Page_Load` event:

```

1  using Microsoft.SharePoint;
2  using Microsoft.SharePoint.WebControls;
3  using System;
4
5  namespace OpenPDFSharePoint.Layouts.OpenPDFSharePoint
6  {
7      public partial class ViewPDF : LayoutsPageBase
8      {
9          /// <summary>
10         /// Gets the file URL extracted from the URL key.
11         /// </summary>
12         public string fileURL { get; private set; }
13
14         /// <summary>
15         /// Page load event.
16         /// </summary>
17         protected void Page_Load(object sender, EventArgs e)
18         {
19             // Extract the file URL for the SharePoint file.
20             fileURL = Request.QueryString["fileURL"];
21         }
22     }
23 }

```

5 Adding a File Handler

A SharePoint file handler can be used to link PDF files to the PSPDFKit PDF editor. The file handlers are simple XML files at a specific location. They instruct SharePoint to redirect to your URL of choice whenever a file with a particular extension is clicked.

Add an `.xml` file named `serverfilesPSPDFKit.xml` to the `XML` folder in the solution with the following content:

```

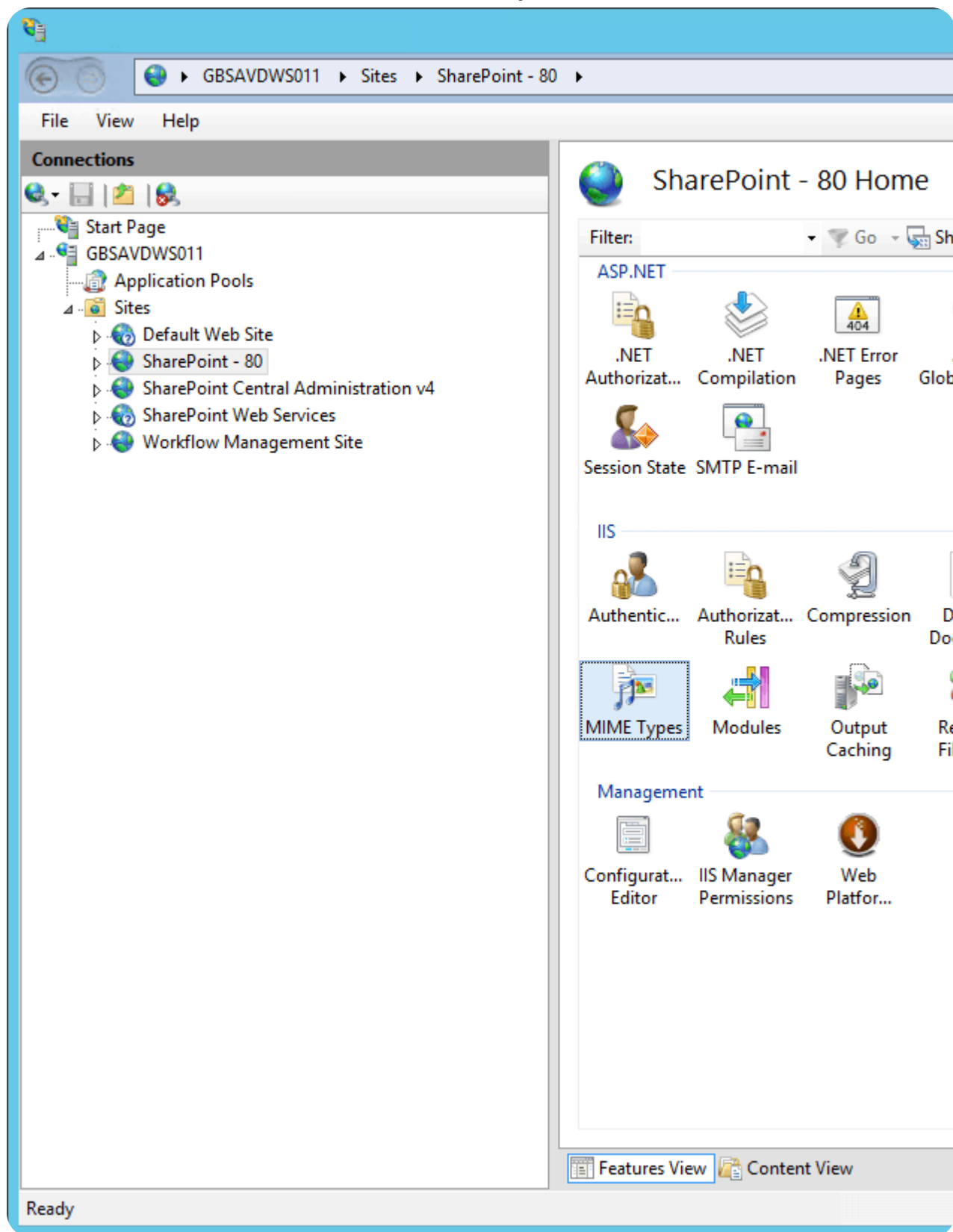
1  <?xml version="1.0" encoding="utf-8" ?>
2  <ServerFiles>
3      <!-- Redirect any links to a PDF file to the PDF Editor application page
4      <Mapping FileExtension="pdf" RedirectUrlTemplate="/_layouts/15/OpenPDFSh
5  </ServerFiles>

```

6 Enabling the WASM MIME Type

Before the above example can be used, verify that `.wasm` files are served by your SharePoint web server.

Start the Internet Information Services (IIS) Manager. Navigate to the website that belongs to your SharePoint instance. Double-click MIME Types in the Features View.



If `.wasm` isn't in the list of extensions, click the Add button in the Actions pane. In the Add MIME Type popup window, enter `".wasm"` (under File name extension:) and `"application/wasm"` (under MIME type:).

Was this helpful?

YES

NO

MIME Type

?

x

File name extension:

.wasm

MIME type: Contact us

application/wasm