# Exercise 3: Creating SharePoint Framework Extensions

## Task 1: Create your project

1. From the PowerShell command prompt, change to the C:/LabFiles/SharePoint directory by executing the following command: cd c:/LabFiles/SharePoint
2. Make a new directory for your SharePoint project files by executing the following command: md SPFxAppCustomizer
3. Navigate to the newly created SharePoint directory by executing the following command: cd SPFxAppCustomizer
4. Run the SharePoint Yeoman generator by executing the following command: yo @microsoft/sharepoint
5. Use the following to complete the prompt that is displayed:
   * **What is your solution name?**: SPFxAppCustomizer
   * **Which baseline packages do you want to target for your component(s)?**: SharePoint Online only (latest)
   * **Where do you want to place the files?**: Use the current folder
   * **Do you want to allow the tenant admin the choice of being able to deploy the solution to all sites immediately without running any feature deployment or adding apps in sites?**: No
   * **Will the components in the solution require permissions to access web APIs that are unique and not shared with other components in the tenant?**: No
   * **Which type of client-side component to create?**: Extension
   * **What type of client-side extension to create?**: Application Customizer
   * **What is your Application Customizer name?**: HelloAppCustomizer
   * **What is your Application Customizer description?**: HelloAppCustomizer description
6. After provisioning the folders required for the project, the generator will install all the dependency packages using NPM.
7. Open the project in Visual Studio Code by executing the following command: code .
8. From the Visual Studio Code **Terminal** prompt, execute the following command: gulp trust-dev-cert

* **Note**: Extensions must be tested in a modern SharePoint page unlike web parts, which can be tested in the local workbench. In addition, extensions also require special URL parameters when requesting the page to load the extension from the local development web server.

1. Obtain the URL of a modern SharePoint page.
2. Open the **./config/serve.json** file.
3. Copy the URL of your modern SharePoint page into the **serveConfigurations.default.pageUrl** property.
4. The SPFx build process **gulp** **serve** task will launch a browser and navigate to this URL, appending the necessary URL parameters to the end of the URL to load the SPFx extension from your local development web server.
   1. Run the project by executing the following command: gulp serve
5. When the SharePoint page loads, SharePoint will prompt you to load the debug scripts. This Is a confirmation check to ensure you really want to load scripts from an untrusted source. In this case, that is your local development web server on https://localhost, which you can trust.
   1. Select the button **Load debug scripts**.
   2. Once the scripts load, a SharePoint dialog alert box will be shown: This alert box is shown by the application customizer. Open the application customizer file located at **./src/extensions/helloAppCustomizer/HelloAppCustomizerApplicationCustomizer.ts** and find the **OnInit()** method. Notice the following line in the method that is triggering the dialog to appear: Dialog.alert(Hello from ${strings.Title}:\n\n${message});
6. Stop the local web server by pressing CTRL+C in the console/terminal window.

## Task 2: Test your extension

1. Run the project by executing the following command: gulp serve
2. When prompted, select the **Load debug scripts** button.
3. Notice that when the page loads, the text defined in the public properties is displayed in the header and footer of the page.
4. Stop the local web server by pressing CTRL+C in the console/terminal window.

## Task 3: Deploying your extension to all sites in the SharePoint Online tenant

1. Locate and open the **./config/package-solution.json** file.
   1. Ensure the solution object has a property named **skipFeatureDeployment** and ensure that the value of this property is set to **true**.
2. Locate and open the **./sharepoint/assets/ClientSideInstance.xml** file. This file contains the values that will be automatically set on the Tenant Wide Extensions list in your SharePoint Online tenant’s App Catalog site when the package is deployed.
3. Build and package the solution by running the following commands, one at a time:

* gulp build  
  gulp bundle --ship  
  gulp package-solution –-ship

1. In the browser, navigate to your SharePoint Online tenant App Catalog site.
2. Select the Apps for SharePoint list in the left navigation pane.
3. Drag the generated ./sharepoint/solution/\*.sppkg file into the Apps for SharePoint list.
4. In the **Do you trust spfx-app-customizer-client-side-solution?** dialog box:
   1. Select the **Make this solution available to all sites in the organization** check box.
   2. Notice the message **This package contains an extension which will be automatically enabled across sites…**.
   3. Select **Deploy**.
5. Select **Site contents** in the left navigation pane.
6. Select **Tenant Wide Extensions**. Depending on when your tenant was created the Tenant Wide Extensions list may be hidden. If you do not see the list in the Site Contents then you will have to navigate to it manually. Do this by appending **/Lists/TenantWideExtensions/AllItems.aspx** to the URL of the app catalog site.
7. In a separate browser window, navigate to any modern page in any modern site within your SharePoint Online tenant. You should see the extension appear in the tenant.

* **Note**: It may take up to 20 minutes for a Tenant Wide Extension to get deployed across the SharePoint Online tenant so you may need to wait to fully test whether your deployment was successful.

1. Stop the local web server by pressing CTRL+C in the console/terminal window.

## Review

In this exercise, you learned how to create and deploy a custom header and footer extension that is applied to all sites in a SharePoint Online tenant.