Web **parts 开发**

<https://Sppnp.sharepoint.com/sites/Group/_layouts/15/workbench.aspx>

SharePoint client-side solution is **HTML/TypeScript based**,

1. Run Yeoman SharePoint Generator, 建立一个project.

Yo @microsoft/sharepoint

Preview 效果build and run it on **a local web server**：

1. The client-side toolchain uses HTTPS endpoint by default.

**install the developer certificate** and preview your web part:

gulp trust-dev-cert

**rebuild project :**

gulp serve

  这时create a local, node-based HTTPS server on localhost:4321

(gulp serve 这个命令在本地启动一个Workbench文件，用来调试刚才创建的这个WebPart)

支持在本地直接进行调试，不要求你安装SharePoint Server，也不需要你真的拥有SharePoint

Online的环境

**Gulp**: A task runner to handle build process tasks such as

* Bundling and minifying JavaScript and CSS files.
* Running tools to call the bundling and minification tasks before each build.
* Compiling SASS files to CSS.
* Compiling TypeScript files to JavaScript.

**在命令行中>code . 打开VS Code 的WebParts**

**Visual Studio Code** provides built-in support for gulp and other task runners. Select Ctrl+Shift+B on Windows or Cmd+Shift+B on Mac to debug and preview your web part.

**SharePoint Workbench** is a developer design surface that enables you to quickly preview and test web parts without deploying them in SharePoint.

假设我已经有了一个SharePoint Online的网站，我想在它里面直接去调试这个WebPart，我可以简单的这样做：

1. 打开任何一个SharePoint Online站点，将其地址复制下来，例如 https://microsoftapc.sharepoint.com/teams/Samplesiteforares
2. 在上面的地址后面追加一段地址 /\_layouts/15/workbench.aspx，在浏览器中访问这个地址后，你将看到一个跟刚才那个本地调试界面很类似的页面

**部署SPFx WebPart**

1. Any client-side web part should extend the **BaseClientSideWebPart** class

因为，**BaseClientSideWebPart** implements the minimal functionality that is required to build a web part. It provides many parameters to validate and access read-only properties such as :

* **displayMode**
* web part properties
* web part context
* web part **instanceId**,
* the web part **domElement**
* and much more.

To learn more about how to work with the property pane and property pane field types, see [Make your SharePoint client-side web part configurable](https://docs.microsoft.com/en-us/sharepoint/dev/spfx/web-parts/basics/integrate-with-property-pane).

**在property pane中加入新的功能**

第一步：import

@microsoft/sp-webpart-base

加入properties to the property pane: a checkbox

第二步：maps the fields to typed objects

因为增加了新的properties，我们需要在**IHelloWorldWebPartProps interface 中加入这些新的properties的属性。**

**第三步：**

**在getPropertyPaneConfiguration** method中adds the new property pane fields and maps them to their respective typed objects.

第四步：

在render() method中，加入properties to the web part properties, you can now access the properties in the same way you accessed the **description** property.

第五步：

在manifest.json中设置默认值

.manifest.json 中defines the web part metadata such as version, id, display name, icon, and description

### Connect Web part to Sharepoint

**通常在Sharepoint中我们可以获得如下数据：**

* Web title
* Web absolute URL
* Web server-relative URL
* User sign-in name

**用this.context.pageContext 可以调用以上资源**

**本地测试：**

**Get the page context**

**在render中加入**

<p class="${ styles.description }">Loading from ${escape(this.context.pageContext.web.title)}</p>

**保存 + gulp serve**

**在**[**https://localhost:4321/temp/workbench.html**](https://localhost:4321/temp/workbench.html)

**在Sharepoint上测试：**

**保证本地gulp serve运行**

**https://mygame.sharepoint.com/\_layouts/workbench.aspx**

[**用LucasSong@mygame.onmicrosoft.com**](mailto:用LucasSong@mygame.onmicrosoft.com)**登陆**

**添加web part时，可以看到hello world**

**(虽然是online 测试, 但本地gulp serve向应)**

**案例 List model**

**本地测试：**

需要建一个mock store

import { ISPList } from './HelloWorldWebPart';

**因为在HelloWorldWebPart中有很多exports, 所有必须特指其中的 ISPList，用{ }**

**而且不需要给出后缀 “.ts”,在HelloWorldWebPart。因为它是default module**

案例中给出的代码It exports the **MockHttpClient** class as a default module so that it can be imported in other files.

**线上测试：**

需要用class **spHttpClient** to execute REST API requests against SharePoint.

所以需要import 从@microsoft/sp-http下

import {

SPHttpClient,

SPHttpClientResponse

} from '@microsoft/sp-http';

this.context.spHttpClient.get

发送GET 请求

**EnvironmentType 用来识别web parts运行环境**

### **Deploy Web parts to a SharePoint page**

1. **封装**

**如果用—ship选项，JavaScript files, CSS, and others被封装在.sppkg package 中，命令行：**

$ gulp package-solution

在sharepoint/solution folder下可以找到。

1. 在Sharepoint中创建 site

<https://mygame.sharepoint.com/_layouts/15/sharepoint.aspx>

因为APP需要在站点上运行。Team site

1. Deploy Helloworld package to app catalog

<https://mygame.sharepoint.com/sites/apps/SitePages/Home.aspx>

左边，选Apps for SharePoint

这时进入App for Sharepoint页面：

把刚才生成的**.sppkg 拖拽到 页面上**

(右上角的setting中可以找到Add an App)

1. Install the client-side solution on your Site

* 回到Sharepoint主页，选刚刚建立的site => 进入页面
* New -> App: 选择刚刚加入的APP –> 开始安装

1. 确保安装完成，在site的左边有**Site Contents，点击进入可以查看app的安装进度。**

这时Webpart已经加载到Sharepoint 的网站上了！

注意：现在的web part app需要localhost运行才能用。因为，封装的Webpart App 中\dist的<your-webpart-guid>.manifest.json中的**internalModuleBaseUrls 依旧指向本地。所以，请在terminal 中运行：**

gulp serve --nobrowser

**Add Helloworld Web part to modern page**

1. 回到刚才建的site （HelloWorld）
2. +Add这时就可以选app啦

虽然已经APP可以在页面上工作了，但是依旧要运行localhost。所以需要Hosting Webparts from Office 365 CDN

### Office 365 Content Delivery Network (CDN)

用来管理APP的

除了CDN以外，还有Azure CDN和Sharepoint library

1. 激活为O365tanent CDN
   * Download and install “Sharepoint online Management Shell”
   * Connect to your Sharepoint Online tenant with PowerShell

Connect-SPOService -Url <https://mygame-admin.sharepoint.com>

Set-SPOTenantCdnEnabled -CdnType Public

过程大约 15 分钟

最终：

\*/MASTERPAGE

\*/STYLE LIBRARY

\*/CLIENTSIDEASSETS

如果configuration pending，此文本表示正在 SharePoint Online 和 CDN 系统之间进行配置。

可用下面命令查看：

Get-SPOTenantCdnOrigins -CdnType Public

**Solution 设置：**

**package-solution.json 文件定义了包元数据**

**includeClientSideAssets** 的默认值为 true。也就是说，静态资产会自动打包到 .sppkg 文件中，无需单独通过外部系统托管资产。

Basics

Web part**属性** 独立于 SharePoint，属性值也是由最终用户管理。 SharePoint 框架提供了一组新功能： Managing web part properties' values and integrate them with SharePoint Search.