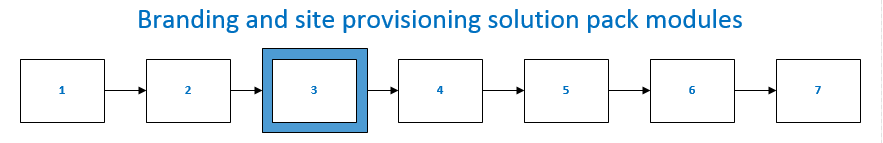
Module 3: Development and Design tools and packages

# About SharePoint development design tools and packaging design assets before you provision sites

Module 3 introduces SharePoint design tools and packaging technologies.



This module:

* **Defines** key terms and concepts related to SharePoint design tools and packaging technologies.
* Describes **packaging** technology options and the capabilities and constraints of each.
* Includes **code examples** that demonstrate how to use the remote provisioning pattern to read and apply branding assets to SharePoint sites.

# Key terms and concepts

Table 1 defines terms and key concepts that apply in this module.

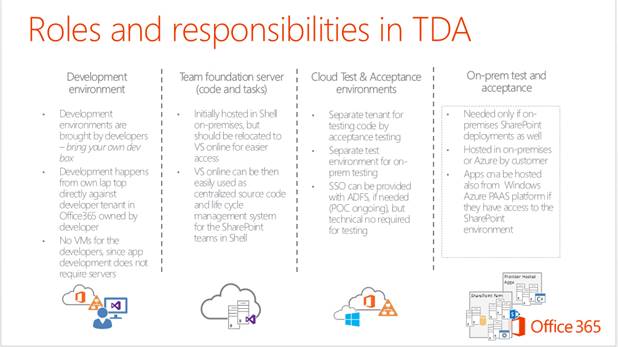
**Table 1. Key terms and concepts**

|  |  |  |
| --- | --- | --- |
| **Term** | **Definition** | **Guidance** |
| [Design Manager](http://msdn.microsoft.com/en-us/library/office/jj822363.aspx) | A feature activated in SharePoint publishing sites or Team sites with publishing enabled used to import and manage site branding assets and export them to a design package. | Use Design Manager to import branding assets created in other tools, such as Adobe PhotoShop or Adobe DreamWeaver, into SharePoint.  SharePoint Designer is not available for use with OneDrive for Business or SharePoint Team sites where publishing is not enabled. |
| [Design Package](http://msdn.microsoft.com/en-us/library/office/jj862342.aspx) | Designed for use with SharePoint 2013 Publishing sites, the design package contains branding assets stored in Design Manager. |  |
| Remote provisioning | A model that provisions sites by using templates and code that runs outside SharePoint in a provider-hosted app. | [Site provisioning techniques and remote provisioning in SharePoint 2013](http://blogs.msdn.com/b/vesku/archive/2013/08/23/site-provisioning-techniques-and-remote-provisioning-in-sharepoint-2013.aspx) (Vesa Juvonen)  [Self-service site provisioning using apps in SharePoint 2013](http://blogs.msdn.com/b/richard_dizeregas_blog/archive/2013/04/04/self-service-site-provisioning-using-apps-for-sharepoint-2013.aspx) (Richard di Zerega) |
| Root web | The first web inside of a site collection. | The root web is also sometimes referred to as the Web Application Root. |
| [Sandboxed solutions](http://msdn.microsoft.com/en-us/library/ff798382.aspx) | .wsp files that contain assemblies, other non-compiled components, and an XML manifest file. A sandbox solution uses partial trust code. |  |
| SharePoint Designer 2013 | An HTML designer and design asset management tool for managing branding elements in SharePoint. In SharePoint 2013, SharePoint Designer mainly supports custom workflows. | [What’s changed in SharePoint Designer 2013](http://msdn.microsoft.com/en-us/library/office/jj728659.aspx)?  [What’s new with SharePoint 2013 site development](http://msdn.microsoft.com/en-us/library/office/jj163942.aspx)? |
| .wsp | A SharePoint solution file | A .wsp is a .cab file that categorizes site assets and organizes them with a manifest.xml file.  [Solutions overview](http://msdn.microsoft.com/en-us/library/office/aa543214(v=office.14).aspx) |

# Development and integration SKUs

The “Development environment considerations” section of the [SharePoint Server 2013 Application Lifecycle Management](http://msdn.microsoft.com/en-us/library/office/dn567995(v=office.15).aspx" \l "DevEnvironment) article on MSDN describes the varieties of development, testing, build, and deployment processes and resources that you’ll want to create when you use SharePoint 2013 as a development platform. Figure 1 lists the separate responsibilities of all the participants in a standard testing, development, and acceptance process.

Figure 1: Roles and responsibilities in testing, development, and acceptance.



The specific number and types of tenants and SKUs that you’ll need will depend on your specific needs, but in most cases you’ll need at least these 2-3 tenants:

* A developer tenant. You can go to the [Sign up for an Office 365 Developer Site](http://msdn.microsoft.com/en-us/library/office/fp179924(v=office.15).aspx) article to learn about the different ways in which you can sign up for and provision a developer SKU. That article points out that you can provision developer sites on an E1 or E3 SKU, but it is a best practice for developers to provision and use their own developer sites, to avoid mixing production and developer data. A separate developer SKU ensures a safe, isolated development environment.
* An integration/testing tenant. You’ll use this site to make sure that new apps and functionality work across more than one site collection and against the services and data of the production environment. You should configure this environment so that it gets capabilities that are in preview. (You can do this by choosing **Service Settings** in your tenant admin console and then selecting **First Release** under the **Updates** setting.) This is also the site where you could run automated testing with Visual Studio online and any other continuous integration testing.
* A production tenant. Only tested, accepted, and approved apps can be released to this tenant. You could create a developer site on this tenant to develop and test apps whose impact is going to be relatively isolated and small in scope, but you should generally avoid mixing production and development environments.

# Design and development tools

After you create site branding assets such as HTML, images, CSS files, and JavaScript files, you need a way to get them into SharePoint.

* Design using the tools you know.
* Decide which SharePoint design tools to use based on your needs and expertise.
* Understanding and using SharePoint design packages and .wsp files.

## Design using the tools you know

Use standard, familiar web design and development tools to create SharePoint site branding assets such as HTML, CSS, images, and JavaScript. For example, you can use Adobe DreamWeaver and Adobe PhotoShop to design the HTML, CSS, JavaScript, and image files you’ll use to brand SharePoint sites. Alternatively, you can use SharePoint Designer 2013 to create, manage, and customize branding assets, or create custom solutions in Visual Studio 2013.

## Understanding and using SharePoint design packages and .wsp files

Depending on whether you’re using Design Manager to create a design package or you’re packaging branding assets in a .wsp file using another tool, your branding assets will be in either a fixed and predictable state or a less fixed and predictable state.

### Design Packages

[Design packages](http://msdn.microsoft.com/en-us/library/office/jj862342.aspx) are .wsp files created by Design Manager that follow predictable rules for packaging design assets. They are essentially [sandboxed solutions](http://msdn.microsoft.com/en-us/library/ff798382.aspx).

The design package includes all files that have been *customized*. For example, if you create a page layout that uses a custom content type, the design package includes the page layout, the custom content type it uses, and all custom site columns. The design package also includes several files related to any composed looks that have been applied to your SharePoint site, including files uploaded to the following locations:

* Site assets library
* Style library
* Master Page Gallery

If you applied composed looks to a site before you applied custom branding, the design package will include files with *.themedcss* and *.themedpng* extensions.

To apply the branding assets in a design package to a SharePoint site, export the design package and use the remote provisioning pattern to apply the contents of the design package.

SharePoint 2013 includes the following APIs that you can use to work with design packages.

**Table 1. API index for design package APIs**

|  |  |  |  |
| --- | --- | --- | --- |
| **SSOM** | **CSOM** | **JSOM** | **REST** |
| [DesignPackage](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.publishing.designpackage.aspx) | [DesignPackage](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.publishing.designpackage.aspx) | [DesignPackage](http://msdn.microsoft.com/en-us/library/office/jj994637.aspx) |  |
| [DesignPackageInfo](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.publishing.designpackageinfo.aspx) | [DesignPackageInfo](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.publishing.designpackageinfo.aspx) | [DesignPackageInfo](http://msdn.microsoft.com/en-us/library/office/jj954520.aspx) |  |

### Using the DesignPackage CSOM to apply the contents of design packages to a SharePoint site

The following example demonstrates how to use the Design Package APIs in the remote provisioning pattern to apply the contents of design packages to a SharePoint site.

**Note** This code was specifically designed for use with Publishing sites. While it is possible to use the Design Packages API to apply branding to Team sites that have the Publishing feature enabled, doing so can introduce long-term supportability issues.

using Microsoft.SharePoint.Client;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using Microsoft.SharePoint.Client.Publishing;

namespace ProviderSharePointAppWeb

{

public partial class Default : System.Web.UI.Page

{

protected void Page\_PreInit(object sender, EventArgs e)

{

Uri redirectUrl;

switch (SharePointContextProvider.CheckRedirectionStatus(Context, out redirectUrl))

{

case RedirectionStatus.Ok:

return;

case RedirectionStatus.ShouldRedirect:

Response.Redirect(redirectUrl.AbsoluteUri, endResponse: true);

break;

case RedirectionStatus.CanNotRedirect:

Response.Write("An error occurred while processing your request.");

Response.End();

break;

}

}

protected void Page\_Load(object sender, EventArgs e)

{

// Use TokenHelper to get the client context and Title property.

// To access other properties, the app may need to request permissions

// on the host web.

var spContext = SharePointContextProvider.Current.GetSharePointContext(Context);

// Publishing feature GUID to use the infrastructure for publishing

Guid PublishingFeature = Guid.Parse("f6924d36-2fa8-4f0b-b16d-06b7250180fa");

// The site-relative URL of the design package to install.

// This sandbox design package should be uploaded to a document library

// for practical purpose this can be a configuration setting in web.config.

string fileRelativePath = @"/sites/devsite/brand/Dev.wsp";

//string fileUrl = @"https://SPXXXXX.com/sites/devsite/brand/Dev.wsp";

using (var clientContext = spContext.CreateUserClientContextForSPHost())

{

// Load the site context explicitly or while installing the API, the path for

// the package will not be resolved.

// If the package cannot be found, an exception is thrown.

var site = clientContext.Site;

clientContext.Load(site);

clientContext.ExecuteQuery();

// Validate whether the Publishing feature is active.

if (IsSiteFeatureActivated(clientContext,PublishingFeature))

{

DesignPackageInfo info = new DesignPackageInfo()

{

PackageGuid = Guid.Empty,

MajorVersion = 1,

MinorVersion = 1,

PackageName = "Dev"

};

Console.WriteLine("Installing design package ");

DesignPackage.Install(clientContext, clientContext.Site, info, fileRelativePath);

clientContext.ExecuteQuery();

Console.WriteLine("Applying design package");

DesignPackage.Apply(clientContext, clientContext.Site, info);

clientContext.ExecuteQuery();

}

}

}

public bool IsSiteFeatureActivated( ClientContext context, Guid guid)

{

var features = context.Site.Features;

context.Load(features);

context.ExecuteQuery();

foreach (var f in features)

{

if (f.DefinitionId.Equals(guid))

return true;

}

return false;

}

}

}

### Using FileCreationInformation to upload branding assets and a master page to a Team site

You can upload the required master pages for Team sites using the [FileCreationInformation](http://msdn.microsoft.com/EN-US/library/office/microsoft.sharepoint.client.filecreationinformation.content.aspx) class and assign the master page to a specified site.

//Assign to content byte[].

var fileCreationInformation = new FileCreationInformation();

//Allow the document to be overwritten.

fileCreationInformation.Content = documentStream;

//Upload URL

fileCreationInformation.Overwrite = true;

fileCreationInformation.Url = siteURL + [masterpageUrl] + masterpageName;

Microsoft.SharePoint.Client.File uploadFile = documentsList.RootFolder.Files.Add(

fileCreationInformation);

### Programmatically installing and uninstalling a Design Package

The SharePoint 2013 [CSOM for design packages](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.publishing.designpackage_members.aspx) includes functionality that you can use to install and uninstall design packages and export design packages to SharePoint Online sites.

Use the [Install](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.publishing.designpackage.install.aspx) method to install the design package onto a site.

public static void Install(

ClientRuntimeContext context,

Site site,

DesignPackageInfo info,

string path

)

[DesignPackageInfo](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.publishing.designpackageinfo.aspx) specifies metadata that describe the contents of the design package that will be installed. string path is the path to the design package that will be installed.

Use the [UnInstall](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.publishing.designpackage.uninstall.aspx) method to uninstall the design package from the site.

public static void UnInstall(

ClientRuntimeContext context,

Site site,

DesignPackageInfo info

)

If you need to brand a Team site with the Publishing feature enabled or a Publishing site on SharePoint Online, you can use the [ExportEnterprise](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.publishing.designpackage.exportenterprise.aspx) method or the [ExportSmallBusiness](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.publishing.designpackage.exportsmallbusiness.aspx) method to export design packages for site templates to the Solution Gallery. Use the **ExportSmallBusiness** method with the small business site template, and use the **ExportEnterprise** method for all other site templates.

public static ClientResult<DesignPackageInfo> ExportEnterprise(

ClientRuntimeContext context,

Site site,

bool includeSearchConfiguration

)

packageName is a string that represents the name of the design package.

This method also includes the option to include the search configuration in the design package.

public static ClientResult<DesignPackageInfo> ExportSmallBusiness(

ClientRuntimeContext context,

Site site,

string packageName,

bool includeSearchConfiguration

)

All design package methods operate at the site collection level.

## SharePoint Online design tool options

The tools you can use to brand a SharePoint Online depend on your SharePoint Online edition and the type of site you want to build.

The Small Business edition includes one Team site and one public site. It does not include a Publishing site. You can use the Site Builder app in SharePoint Online to customize public site branding.

The Enterprise edition includes a Team site collection at the root web application for the domain that does not include Publishing. It does not include a public site. Use Design Manager to manage SharePoint site branding elements for the Publishing site in the SharePoint Online Enterprise edition.

## Conclusion

In this module, you received basic guidance to help choose the correct SharePoint design tools to manage site branding assets. You can use the remote provisioning pattern to apply the contents of a design package to a SharePoint site, or you can use [FileCreationInformation](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.filecreationinformation.aspx) to apply branding elements.