Understanding SharePoint Development Choices

SharePoint is an incredibly powerful platform that can support a wide variety of business scenarios. While many solutions can be configured using out of the box features, it may be necessary to develop customizations in order to meet all of the business requirements for a project. Customizations can come in many forms that range in complexity from simple visualizations in Web Part displays, custom forms, and timer jobs, all the way to a custom service application that extends the application to provide fully packaged solutions.

These customizations can be created using a number of different techniques, so it is important to understand the strengths and limitations of each of the development paths. It is also important to understand the development tools that are available and what they can produce.

In the sections that follow, you will find a brief overview of the different customization options that are available, tools that can be used to create them, as well as some additional considerations when choosing a development path.

Server-side development

The **Server Object Model** (**Server OM**) is an application programming interface that supports full interaction with all of the SharePoint configuration and data, and is organized according to the conceptual system hierarchy, which allows easy navigation for people familiar with SharePoint. All of the main objects are available with properties, methods, and sub-objects accessible. There is also a series of context objects that make it easy to find or set your starting point, such as the site the user is on when the solution is called.

Frequently used Server OM objects and collections include:

Server OM Object	Description
`SPFarm`	The object representation for the entire SharePoint system. Parent object to the `SPServer` and `SPService` collections.
`SPSite`	Represents a site collection.
`SPWeb`	Represents a web or subsite.
`SPList`	Represents a list or library.
`SPListItemCollection`	Represents the items within a list or library.
`SPField`	Represents a single field.
`SPListItem`	Represents a single list item.
`SPContext`	Represents current `HttpContext` information including information on the current site and user.

All code has to be deployed to the server and references the SharePoint objects loaded on the local server. Using the Server OM, you have full access to the SharePoint farm and underlying services, allowing you to build robust solutions and automate configuration changes. A few examples include custom Web Parts, Application Pages, event receivers, custom timer jobs, custom workflows, custom workflow activities, custom web services, or you could even create a custom service application that can extend the services of the SharePoint platform.

With the 2010 version there are now two options for developing solutions:

- Farm solutions
- Sandboxed solutions

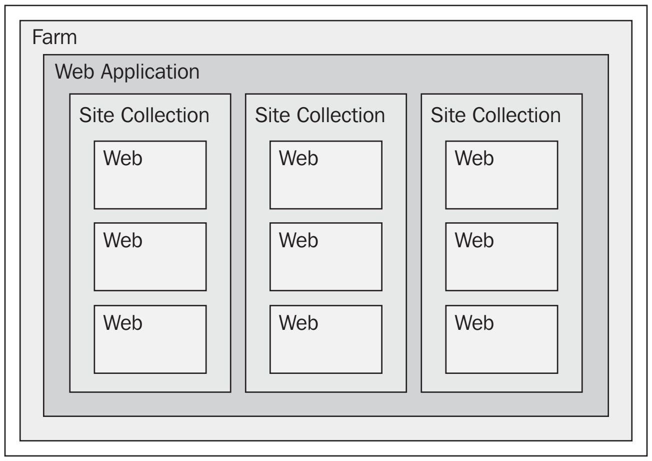
Farm solutions

Creating farm solutions has been the traditional way to create solutions in SharePoint, going back all the way to the earliest versions. The Server OM provides access to the system configuration and data, allowing developers the ability to create solutions or automate configuration changes.

When deployed, the solution's objects are either loaded into the server's **Global Assembly Cache** (**GAC**) or into the `bin` directory of the web application(s). This allows farm solutions to run in the same memory as SharePoint, which means the developer can have full access to the server, farm configuration, and content. Farm solutions do not have the overhead of the sandbox solution's process overhead. It also allows the developer to access additional caching features available inside of SharePoint to optimize the solution's performance. This does add risk, as poorly written or tested solutions can impact overall system performance, or potentially crash the system because farm solutions are treated as full trust solutions.

Farm solutions can only be deployed by a farm administrator, and in many cases the governance policy may also dictate a formal review, or a change management process can be followed before the farm administrator is allowed to deploy the solution.

Farm solutions are scoped for activation at a set level; Farm, Web Application, Site Collection, or Web, allowing the developer to set it based on how and where the solution will be used. See the following diagram for an illustration. A generally best practice is to scope a solution as narrowly as possible to provide more control on where it is activated. It is also important to note that solutions scoped for Site or Web can be activated by their respective owners based on site permissions.



Good candidates for farm solutions include:

- Custom master pages, page layouts, and global CSS
- Complex Web Parts
- Complex Application Pages
- Complex workflows
- Complex event receivers
- Global navigation providers
- Complex cross-site or cross-web application rollups

Sandboxed solutions

Sandboxed solutions were introduced with SharePoint 2010 and provide an option for deploying partially trusted solutions to individual site collections, without the need to involve the full farm administrator in the deployment process.

Unlike the farm solutions previously discussed, these are partially trusted solutions and there are some limitations. Where the farm solutions run in the same memory space as the main SharePoint services and application pools, sandboxed solutions run in an isolated memory pool, with resource quotas that can be managed by administrators. This isolation means that poorly written or tested solutions cannot slow down or crash your main server processes. If a solution is having trouble running, it will be automatically disabled. In addition, the resource throttling and quotas can limit the amount of resources each solution can consume within the site collection. This safe mode approach can speed up the development cycle, but should not remove the need to properly test your solutions.

This separation does come at a cost though, and there is some process overhead involved with communicating across the different worker processes. In cases where the solution will be actively used by a large number of people, it may be more difficult to meet any scalability requirements.

Unlike farm solutions, developers or site administrators do not need administrative access to the physical servers or to the farm's Central Administration to deploy their solutions; they only need access to upload the solution to the Solution Gallery for a specific site collection. This provides custom solution capabilities to organizations that have very strict change control policies, or are prevented from deploying custom solutions as in cloud-based environments.

As the solutions are only partially trusted and uploaded to a specific site collection there are limitations.

Common limitations include that they:

- Cannot access resources outside of the farm
- Cannot access a database directly
- Cannot call unmanaged code
- Cannot connect to or access resources in a different site collection
- Cannot write data to the local server

For any solution that fits cleanly into a single site collection, and does not need to reach out to resources outside of that boundary, sandboxed solutions offer additional flexibility.

Good candidates for sandboxed solutions include:

- Web Parts that only require access to the local site collection
- Event receivers that only require access to the local site collection
- Application Pages that only require access to the local site collection
- List Definitions

Connecting to SharePoint through web services

SharePoint also includes a comprehensive set of web services that can also be used to interact with system configuration and content. The available services cover both the base SharePoint functionality covered in the SharePoint Foundation version, as well as the extended functionality available in SharePoint Server as shown in the next table.

The services come in two formats: SOAP based services and new WCF based services that support REST.

Web service	**Version**
Alerts	Foundation and Server
Authentication	Foundation and Server
Copy	Foundation and Server
Forms	Foundation and Server
Lists	Foundation and Server
Meetings	Foundation and Server
People	Foundation and Server

Web service	Version
Permissions	Foundation and Server
Site Data	Foundation and Server
Users and Groups	Foundation and Server
Versions	Foundation and Server
Views	Foundation and Server
Web Part Pages	Foundation and Server
Webs	Foundation and Server
Published Links Service	Server
Search	Server
User Profile Service	Server
Workflow	Server
SocialDataService	Server
TaxonomyClientService	Server

These services provide an effective integration option, because they can be called from any network accessible computer written in any technology that can support the standard services. As an example, the integrations built into MS Office applications use these web services to integrate with SharePoint, as well as the list extraction support built into SQL Server's Integration Services.

Good candidates for using SharePoint Web Services:

- Consume data in InfoPath forms
- Integrate with SharePoint from non-SharePoint systems
- Support client-side customizations

Client-side development

Client-side development can be leveraged to provide rich, dynamic content and experiences. With client-side development all of the code runs from the client which in most cases is the end user's web browser. The code is written in a scripting language such as JavaScript, ECMAScript, or in the case of Silverlight, using XOML and references script libraries instead of server-based objects.

There are two common options for interacting with SharePoint from client-side solutions: the Client Object Model and the jQuery libraries through the SharePoint Web Services.

Client Object Model

The **Client Object Model** (**Client OM**) is an application programming interface that supports extensive, but not full interaction with the SharePoint configuration and data. It was added with the SharePoint 2010 release to provide an easy way to interact with SharePoint configuration and data, without the need for code deployed to the server. It puts a simple wrapper around the previously discussed web services, simplifying the interaction and removing the need to create and parse the XML needed to interact with the web services. The client-side code can be called from Silverlight, ECMAScript, or via your .NET code.

The Client OM can be used to include the Ajax enabled features allowing content and data refreshes, or loads without the need for a page post back which reloads or redirects the page.

The Client OM is limited in that it cannot connect or perform calls outside of the site context that it is called from. This means that it cannot be used for cross-site calls or for aggregating content from multiple web applications or site collections.

There are also some functions available in the Server OM that are not supported in the web services, and therefore are not supported in the Client OM due to security boundaries.

Frequently used Client OM objects and collections include:

Object	**Description**
`ClientContext`	Represents current context, and is used as the main entry point for interacting with the Client OM.
`Site`	Represents a site collection.
`Web`	Represents a web or subsite.
`WebCollection`	Represents the collection of webs within the site collection.
`List`	Represents a list or library.
`ListCollection`	Represents the collection of lists within the site collection.
`ListItemCollection`	Represents the items within a list or library.
`Field`	Represents a single field.
`FieldCollection`	Represents the collection of fields.
`ListItem`	Represents a single list item.

Good candidates for Client OM solutions include:

- Modular visualizations or content, similar to a Web Part
- Enhancing DataForm Web Parts
- Silverlight solutions
- Page-level customizations and enhancements

Using jQuery

The jQuery library (`http://jquery.com/`) can be leveraged to support client-side development and Ajax enabled functions. This JavaScript-based library is completely cross-browser compliant, and offers a rich set of selectors and advanced support for animations that can be used to provide a much richer interface than static HTML. This library is completely platform independent, and there is nothing SharePoint specific about this library. As there are a number of Ajax methods that make it easy to consume web services, you can use the library to interact with SharePoint through its web services.

In its base format, you would need advanced knowledge of all of SharePoint's Web Services. To help simplify the development effort, the SPServices library (`http://spservices.codeplex.com/`) was created to provide a wrapper around SharePoint's Web Services. This library was created in the summer of 2009 originally for the SharePoint 2007 version, but it is also fully functional with SharePoint 2010. While it conceptually offers the same functionality as the Client OM, it could be used for solutions that need to be backward compatible with 2007, or in cases where there is an `SPServices` method not supported in the current Client OM.

Good candidates for jQuery solutions include:

- Modular visualizations or content, similar to a Web Part
- Enhancing DataForm Web Parts
- Page-level customizations

Deploying and managing client-side customizations

There are a number of ways to deploy and manage client-side code inside of SharePoint:

- Use the Content Editor Web Part with a reference to your content or script
- Edit the page directly with SharePoint Designer

It is important to note that editing pages directly with SharePoint Designer will put the page in an unghosted state, which means that the page will no longer reference the common version of the page, and instead store a version of the page in the content database that the site is stored in. This change will have a small impact on performance, but can also complicate future upgrades and should therefore be done with caution.

It is important to consider how changes and versions will be managed. If you are adding code and functionality to pages, you should have versioning configured for the library in which the pages are located, as well as any libraries used to store your scripts. This will allow you to compare versions, or to recover a prior version, if things go wrong.

Many environments have very strict change management or deployment guidelines for any code that is deployed to the server by administrators. One of the big advantages to the client-side development approach is that system users with appropriate access can deploy their customizations themselves without the involvement of the administrators or change control group. This typically means that customizations can be created and deployed very quickly, perhaps without the formality required for fully trusted code deployed to the server. While there is some risk that a defective customization will cause an error or unexpected results, the impact is typically localized to the page the customization is used on, or the content it interacts with.

SharePoint development tools

Microsoft has a number of tools that can be used to create business solutions including Visual Studio, SharePoint Designer, and InfoPath Designer.

Visual Studio

Visual Studio 2010 provides a robust environment for developing solutions for SharePoint 2010. Unlike previous versions, there is native support for most of the base project templates. It also natively supports the packaging and deployment process, producing the full feature and package objects, and also supports automated deployment and retraction to/from the local SharePoint server to support development efforts.

The available templates include:

- Visual Web Part
- Web Part
- Application Page

- Sequential Workflow
- State Machine Workflow
- Event Receiver
- Module
- Business Data Connectivity Model
- Content Type
- List Definition from Content Type
- List Definition
- List Instance
- User Control

In addition to the default templates provided by Microsoft, there is an additional community driven project that can add additional or customized project templates under the **Community Kit for SharePoint Development** (**CKS:Dev**) project available on CodePlex (`http://cksdev.codeplex.com/`).

The available templates with the CKS:Dev package include:

- Fluent UI Visual Web Part
- Basic Site Page
- Starter Master Page
- Branding
- Blank Site Definition
- WCF Service
- Full Trust Proxy Operation
- SPMetal Definition
- Custom Action Group
- Custom Action
- Hide Custom Action
- Delegate Control
- Basic Service Application
- SharePoint PowerShell Commandlet
- SharePoint PowerShell Pipe Binding

In addition to any server-side development, Visual Studio can also provide a rich development experience for client-side development of scripts and XSL, though it cannot be used to edit a SharePoint page directly, or publish scripts to SharePoint outside of a SharePoint WSP package.

SharePoint Designer

SharePoint Designer 2010 is a free tool targeted to site administrators and information workers. It excels as a tool that can be used for site-level administration and customization. Since it connects to and interacts directly with a specified site, customizations can be deployed directly by the user, without administrator assistance. This makes SharePoint Designer an incredibly important tool in either cloud environments or environments with a strict server change control policy.

Through SharePoint Designer, you can create or customize the following:

Customization	**Description**
DataForms	Add and configure the DataForm Web Part on a page. The control generates standard XSL for the user with multiple display options.
Data Connections	Add and maintain Data Connections available on the site.
External Content Types and External Lists	Define External Content Types and External Lists that can load Line of Business data for use on the site.
Master Pages and Page Layouts	Create or edit Master Pages and Page Layouts.
Sequential Workflows	Create or edit workflows using the graphical interface.

InfoPath Designer

InfoPath Designer 2010 is a tool that is included with the Office 2010 Professional Plus suite. It can be used to create electronic forms that can be submitted to a SharePoint library. The forms can be configured to open in the local InfoPath Filler client. With SharePoint Server Enterprise you have the option of deploying the forms as Web-enabled forms to Forms Services. InfoPath Designer can also be used to customize the default SharePoint list and library forms for New Item, Display Item, or Edit Item pages, allowing much more robust forms far more easily than creating custom Application Pages in Visual Studio.

The InfoPath Designer tool is targeted at developers, information workers, and Power Users able to generate office forms and templates. It provides an easy to use graphical interface for defining form fields, and easy to use property windows to configure business rules and back end data connections. It has the ability to easily connect to data sources including SharePoint lists, SQL databases, and web services including SharePoint's own web services.

When publishing the forms, you have the ability to publish it as a Content Type along with the ability to specify the form fields you would like to include as Site Columns or Library Columns. This allows you to map form fields and data to metadata available within the library. In many cases InfoPath forms are developed in conjunction with custom workflows to automate the form submission and processing previously handled by paper based processes.

For advanced cases, there is also support for including managed code inside of the InfoPath forms. While including this code complicates the overall development and deployment process, it does offer flexibility to organizations that do not have developers available to create custom Application Pages in Visual Studio.

Note that including managed code is not supported in Office 365 or many cloud based environments.

Choosing a development path

In the previous sections we reviewed the capabilities along with some advantages and disadvantages of each of the development and customization paths. Choosing the right path for your solution requires that you consider your environment and the requirements of the solution.

Environment considerations

There are a number of environment variables to consider that have an immediate impact on your development options.

Cloud environments

Since cloud based SharePoint environments such as Office 365 do not support Farm Solutions, sandboxed solutions and client-side code represent the sole methods for creating customizations. Even if you are not in the cloud today, it would be a good idea to consider if that is a possibility in the next few years. If you have a simple solution that can be accommodated with either sandboxed solutions or client-side code, then it might be a good idea to "cloud proof" your customizations to reduce the hassle and rework later.

Governance, change management policies, and server access

In many organizations there are either governance or change management policies that can define capabilities or potentially rule out some of the development options. That should be given strong consideration before choosing a development path. In cases where there is a robust change management policy for anything deployed to the server, the stated **Service Level Agreement** (**SLA**) should be taken into consideration, since it can sometimes take weeks for changes to make their way through the entire life-cycle process of plan, build, test, and deploy. Those constraints can make it very difficult to provide updates or respond to change quickly, so in those cases it can be very important to choose a path where you can be responsive and deploy quickly.

It is also important to consider how your teams are organized and who has what access to the SharePoint system's central administration. In some organizations, access is tightly controlled and given to a small set of administrators who do not create customizations. In other organizations the administrators are also the developers and have access to both create and deploy solutions as needed.

Solution reuse

When making a choice on the development path, it is also important to consider how or where the solution will be used. The approach may be completely different when developing a reusable solution expected to be deployed on sites throughout multiple farms, when compared to the approach for a simple solution intended to be used on only one site collection. For the reusable solution it will be critical that updates be deployed centrally with a solution package update built within Visual Studio, but for a single use customization it may be acceptable to manually apply it where it is needed, using either browser customizations, through SharePoint Designer, or with InfoPath Designer in the case of forms development.

Another example is when working on customizations to master pages and CSS. While this can effectively be applied to a single site collection using SharePoint Designer, it would be extremely difficult to try and maintain it manually across 1,000 site collections throughout the organization. Using Visual Studio to package and deploy the solutions will not only make the solution more maintainable, but it will also give the developer the option of including additional code that can automate additional tasks through the use of feature receivers and event receivers. Within the context of the existing example, this can include automatically setting the Master Page when the feature is activated, as well as when any new sites are provisioned.

Scalability of solutions

When you are architecting or designing business solutions it is important to consider the overall scalability requirements. Consider how many users will be using the system, and what the overall performance requirements are. Designing a feature to support ten concurrent users is much different than designing a feature for five hundred concurrent users. Also, in cases of content rollup, as the number of data sources or sites increase, the performance of the feature will change substantially. In some cases, even commercially available rollup features fail completely, when trying to aggregate content from more than one hundred sites.

Strongly typed server-side code, running in a capable environment, will always perform better than weakly typed client-side code. In addition to actual code execution, another downside of client-side development is that since everything is running on the client, all of the content needs to be downloaded to the client. That means there is potentially extra content being transferred, but not displayed. In addition, client-side development does not have the ability to cache data the way server-side development does.

For simple solutions or in small environments, the scalability of the solution may not be a real concern and far less important than the value provided by the easy deployment options. For more complex or heavily used solutions, the scalability of server-side development should be given serious consideration.

Additional best practice and scalability information can be found on MSDN here: `http://msdn.microsoft.com/en-us/sharepoint/ff660756.aspx`.

Application Lifecycle Management

Application Lifecycle Management (**ALM**) is a software development process that supports the combined management of all application requirements, source, build, and testing processes. For organizations with a focus on ALM, there may be a requirement to include all formal projects and development in the ALM tool or environment such as **Team Foundation Server** (**TFS**). Development tools like Visual Studio provide full integration with standard ALM tools including TFS, which means that solutions created in Visual Studio, using the Server OM, will work transparently with ALM processes.

For client-side development or anything done in SharePoint Designer, there is currently no support for ALM, so things are not so easy. To support ALM with the other SharePoint development tools, the developer will need to manually keep source files synchronized through a manual copy and paste process. This can make managing larger, more complex projects, using these tools and technologies, a lot more tedious. In cases where you have a lot of code, or a mixture of client and server code, you can also consider deploying the client-side scripts to the server as part of a solution package, but in doing so you lose the flexibility and convenience of end user deployed customizations.

Additional best practice information regarding ALM can be found on MSDN here: `http://msdn.microsoft.com/en-us/sharepoint/dd552992`.

Summary

SharePoint developers have a rich set of development options and tools that can be leveraged to develop and deploy rich business solutions. These solutions have the potential to increase the overall value of SharePoint and to provide capabilities not available in many organizations.

For development options we covered server-side development with the Server OM for both Farm and Sandbox solutions, connecting to SharePoint through Web Services, and client-side development using either the Client OM or the jQuery libraries through SharePoint web services directly to provide a rich experience. Additional details can be found on the MSDN website `http://msdn.microsoft.com/en-us/sharepoint` or in the SharePoint Software Development Kit documentation `http://msdn.microsoft.com/sharepoint/gg637004.aspx`.

For tools, we covered the robust features available in Visual Studio 2010, SharePoint Designer 2010, and InfoPath Designer 2010, and how the tools match up against the development options.

I encourage everyone to spend some time working through each of the development paths so that they can better understand for themselves the pros and cons of these options, as well as how their personal skills can be best leveraged to build and maintain these solutions.

This book is not intended to provide a deep dive into all of the development options and techniques, but will provide a number of sample solutions that can be used as blueprints for many common projects. The development paths outlined earlier will be used to create these solutions throughout this book, giving you the opportunity to become familiar with the tools and techniques.