Sitecore CMS - Organization And Structure For Multiple MVC Projects

Version : 8.6 Draft
Created : 2012-03-21 11:47am
Modified : 2016-09-12 03:13pm
Authored By : William Chang

To support multiple sites in a single web project, the best practice is to organize the CMS tree structure and file structure in a way that it will isolate completely different sites and prevent spaghettification. This strategy is based on Modular Architecture or **Component-Based Architecture** principle.

Tree Structure

- sitecore
    - Content
        - YourContainerName
            - YourSiteName0001
            - YourSiteName0002
            - YourSiteName0003
        - Shared
    - Layout
        - Layouts
            - YourContainerName
                - YourSiteName0001
                - YourSiteName0002
                - YourSiteName0003
        - Placeholder Settings
            - yourcontainername_yourmeaningfulname (all lowercase, underscore separator)
        - Renderings
            - YourContainerName
            - Shared
        - Sublayouts
            - YourContainerName
            - Shared
    - Media Library
        - YourContainerName
            - YourSiteName0001
            - YourSiteName0002
            - YourSiteName0003
        - Shared
    - System

        - Dictionary
            - YourContainerName
                - YourSiteName0001
                - YourSiteName0002
                - YourSiteName0003
        - Marketing Center
            - Goals
                - YourContainerName
        - Modules
            - Web Forms for Marketers
                - YourContainerName
        - Workflows
            - YourContainerName
    - Templates
        - Branches
            - User Defined
                - YourContainerName
        - User Defined
            - YourContainerName
            - Shared

File Structure

- webroot
    - _assets/
        - YourContainerName/
            - css/
            - fonts/
            - img/
            - js/
            - lib/
            - themes/
                - YourThemeName
        - Shared/
    - App_Config/
        - Include/
            - ORG.YourContainerName.config
            - ORG.YourContainerName.YourMeaningfulName.config (period separator)
    - Areas/
        - YourContainerName/
            - Controllers/
            - Views/

        - Shared/
    - sitecore modules/
        - YourContainerName/
        - Shared/
    - support/
        - YourContainerName/
        - Shared/

The "_assets" directory must contain all front-end or client-side files. The "lib" directory within the "_assets" directory is for bundled third-party files (eg jQuery plugins). The "support" directory is for miscellaneous files, for example any code files (eg .vb, .cs) and physical pages (eg .html, .aspx, .ascx, .ashx) not referenced by the CMS. The "Shared" directories are for multiple distinct projects to reuse basic components, widgets, modules, and user controls. Files under the "Shared" directories must be designed to be reusable and portable (with few or no dependencies).

Naming Conventions

A consistent and relevant naming convention will reduce frustration, increase accessibility, and discoverability. Spaces, hyphens, and other special characters do not translate well in a web environment and programmatically mapping fields like code generated files for entities or models, so avoid using them.

Examples of Sitecore CMS Data Template

Template Names (UpperCamelCase, not for Display Name) : HomePage, HeaderNavigation
Section Names (Capitalize All Words) : Meta, Content
Field Names (UpperCamelCase, not for Display Name) : MetaKeywords, ContentBody

Examples of Sitecore CMS Layout

YourMeaningfulNameLayout.cshtml

Examples of Sitecore CMS Sublayout

YourMeaningfulNameSublayout.ascx

Examples of Sitecore CMS Rendering

YourMeaningfulNameRendering.cshtml

Namespace (for Microsoft Visual Studio Solution, Projects, Assembly Name)

A namespace is an abstract container or environment created to hold a logical grouping of unique identifiers or names. They provide a means of grouping logically related identifiers into corresponding namespaces, thereby making the CMS more maintainable and predictable.

Examples of Namespace

```
namespace ORG.YourProjectName {
   public class HomeController : BaseController {}
}
```

Examples of Microsoft Visual Studio Project (.csproj)

ORG.YourProjectName.Business
ORG.YourProjectName.Service
ORG.YourProjectName.Web

Examples of Assembly Name (.dll)

ORG.YourProjectName.Business.dll
ORG.YourProjectName.Service.dll
ORG.YourProjectName.Web.dll

Configuration (for Settings, References, Constants)

All settings and constants should not be stored in a web.config, app.config, and global class file. Because the CMS heavily uses the web.config file and maintaining the web.config in each upgrade for every environment is cumbersome and time-consuming. The CMS has an auto-include feature to create and alter configurations without modifying the web.config file. All custom configurations must use the auto-include feature which are stored in the "App_Config/Include" folder containing the ".config" file extension.

Example of Configuration File (XML)

Webroot\App_Config\Include\ORG.Sites.config

```
<configuration xmlns:patch="http://www.sitecore.net/xmlconfig/">
  <sitecore>
    <settings>
      <setting name="ORG.YourContainerName.Item1" value="Hello" />
      <setting name="ORG.YourContainerName.Item2" value="World" />
    </settings>
  </sitecore>
</configuration>
```

Get Configuration Setting Programmatically

Please make sure to add a reference, Sitecore.Kernel.dll, to your project before using the CMS configuration API.

```
// Get named setting with default.
string value = Sitecore.Configuration.Settings.GetSetting("ORG.YourContainerName.Item1",
"YourDefaultStringValue");
```

Logging (for Reporting, Exception Handling, Try-Catch Statements)

The CMS has a log feature to write messages to an auto-generated log file and all log files are centralized in one location for administrators to review, so there is no need for another log implementation. Please make sure to add a reference, Sitecore.Kernel.dll, to your project before using the CMS log API.

```
Sitecore.Diagnostics.Log.Info("ORG.YourContainerName : YourMeaningfulMessage",
"ORG.YourContainerName");

Sitecore.Diagnostics.Log.Warn("ORG.YourContainerName : YourMeaningfulMessage",
"ORG.YourContainerName");

Sitecore.Diagnostics.Log.Debug("ORG.YourContainerName : YourMeaningfulMessage",
"ORG.YourContainerName");

Sitecore.Diagnostics.Log.Error("ORG.YourContainerName : YourMeaningfulMessage",
"ORG.YourContainerName");
```

Microsoft Visual Studio Solution File Structure

The physical structure shown below is a basic guide on how a VS Solution should be organized. Depending on the requirements having an "ORG.YourProjectName.Core" project is optional. The "ORG.YourProjectName.Web" project is a web application project type, which follows the physical structure under the webroot directory of the CMS.

- ORG.SitecoreSites/
    - ORG.YourProjectName.Files/
        - Documents/
            - README.txt
            - Functional_Specifications.txt
            - Technical_Requirements.txt
            - Information_Architecture.txt

                - Wireframes/
                    - Wireframe001.psd
                - Compositions/
                    - Composition001.psd
            - External/
                - ThirdParty.dll
            - SitecorePackages/
                - YourContainerName.zip
    - ORG.YourProjectName.Core/
        - Helpers/
        - Utilities/
        - ORG.YourProjectName.Core.csproj
    - ORG.YourProjectName.Business/
        - Managers/
        - Services/
        - ORG.YourProjectName.Business.csproj
    - ORG.YourProjectName.SitecoreExtension/
        - Events/
        - Pipelines/
        - ORG.YourProjectName.SitecoreExtension.csproj
    - ORG.YourProjectName.Web/
        - _assets/
            - YourContainerName/
        - App_Config/
            - Include/
                - ORG.Shared.config
                - ORG.Sites.config
                - ORG.YourContainerName.config
        - Areas/
            - YourProjectName/
                - Controllers/
                - Views/
        - sitecore modules/
            - YourContainerName/
        - support/
            - YourContainerName/
        - ORG.YourProjectName.Web.csproj
    - packages/
        - repositories.config
    - ORG.SitecoreSites.sln

The "Documents" directory is for documentation of the project development and design. The "README.txt" is the minimal requirement for documentation, it must contain detailed

instructions on dependencies, setup, and deployment to an empty webroot directory. The "SitecorePackages" directory is for exporting and importing items to the CMS which contain content, media, layouts, and data templates.

Microsoft Team Foundation Server (TFS) Source Control Repository Structure

The physical structure shown below is a basic guide on how a TFS source control should be organized.

- ORG_CMS_COLLECTION
    - SRC
        - Drupal
        - Sitecore
            - DEV Branch
                - ORG.SitecoreSites/
                    - ORG.YourProjectName.Web/
                    - ORG.SitecoreSites.sln
            - MAIN Branch
            - RELEASE Branch
        - Sitefinity
        - Umbraco
        - WordPress

Git Source Control Repository Structure

The physical structure shown below is a basic guide on how a Git repository should be organized.

- ORG_CMS_SITECORE_REPOSITORY
    - dev Branch
        - build/
            - Build.bat
            - Build.ps1
            - UninstallSitecore.bat
            - UninstallSitecore.ps1
        - docs/
        - src/
            - ORG.YourProjectName.Web/
            - ORG.SitecoreSites.sln
        - tools/
        - .gitignore
        - README.md
    - master Branch

- ORG_CMS_UMBRACO_REPOSITORY
- ORG_CMS_WORDPRESS_REPOSITORY