Sitecore CMS Implementation Best Practices
V 2.0

Technical Best Practices for Implementing Sitecore CMS in Your Website

This white paper was developed by Oshyn, a valued Sitecore Partner.

www.oshyn.com

sitecore®

oshyn
POWERING INNOVATION™

1.888.483.1770 | www.oshyn.com | newbusiness@oshyn.com

Table of Contents

Executive Summary

Sitecore CMS is an extensive Web Content Management (WCM) platform. It offers reduced IT expenditures, a streamlined content lifecycle, and a return of content control to the subject matter experts. The newest incarnation of Sitecore CMS version 6.0 is a mature product that incorporates standard social media components such as wikis, blogs, RSS syndication and "e-mail a friend" features.

Based on standard .NET technologies, Sitecore offers customers a seemingly clear path to implementation. Though, like any other project implementation, there is careful planning required. And only by planning carefully, you're able to execute to perfection. Every Sitecore implementation requires you to outline the basic features and then create templates and components based on these features. Oshyn has extensive experience with implementing many content management systems for various clients. This experience allows us to understand the overall business goals and consequently create meaningful, user-centric features that will help organizations achieve those goals. We understand the key decisions that must be made prior to implementation and have some valuable tips that will ensure your business and technical users continued to have a positive and productive experience with the Sitecore content management system.

This is the second white paper Oshyn has created about Sitecore best practices. While our first Sitecore Best Practices white paper focuses on more general business best practices, this paper dives deeper into more technical side of your Sitecore implementation.

How to read this white paper

In order to help you apply these best practices to your implementation, we've categorized our recommendations as follows:

- Scalability Best Practices
- Template Best Practices
- Presentation Component Best Practices
- Content Structure Best Practices
- Media Assets Best Practices

- Caching Best Practices
- Workflow Best Practices
- Development Best Practices.

For each of the practices, we have mentioned the area it impacts such as:

- *"Content Editor Experience"* – to indicate better content editor usage experience
- *"Site Visitor Experience"* –to indicate it has impact on Site visitor experience
- *"Implementation"* – to indicate that it is required for consideration during implementation
- *"Performance"* – to indicate that the issue has some performance related impact
- *"Security"* – to indicate it has some security considerations
- *"Functionality"* – to indicate it has some impact to the functionality

Scalability Best Practices

In general, CMS servers are distinguished into Authoring/Master and Delivery servers, as the requirements and the users interacting with the Authoring and Delivery servers are usually different.

#	Practice	Rationale	Impact
1	Enable sticky sessions on Authoring Sitecore Servers in case of clustered Authoring servers, as CM instances must be in “InProc” session state mode.	Authoring Sitecore is recommended (by default) to be operating in “InProc” session mode. <sessionState mode="InProc" stateConnectionString="tcpip=127.0.0.1:42424" sqlConnectionString="data source=127.0.0.1;user id=**;password=**" cookieless="false" timeout="20"/> Enabling sticky sessions will cause consecutive requests of the user session stick with one server, avoiding any loss of state for the user.	Content Editor Experience
2	On Authoring Servers, if Upload Watcher application is not required, then disable it. If enabled, then use Upload Filter Tool to restrict the type of files that can be uploaded to the Upload Folder and deny script and execute permissions on this /upload folder. Assuming content process flow happens from Authoring to Delivery Servers, disable the upload watcher and deny script and execute permissions on upload folder.	If Upload Watcher is not disabled, any files when dropped into the /upload folder will automatically be imported as items into Sitecore. If script and execute permissions are not denied, then executables can be uploaded and executed, opening up a security hole. To disable Upload Watcher, Comment this line <add name="SitecoreUploadWatcher" type="Sitecore.Resources.Media.UploadWatcher, Sitecore.Kernel" />	Security
3	In case of clustered Delivery Servers, make sure application Session and Viewstate stored .NET objects are serialize-able and the modules being used work in clustered mode	Any Session storage mechanism used in Clustered scenarios involves serializing and de-serializing objects stored in Session. An error is thrown if the object being stored in session cannot be serialized. Serialization can be done by using the “SerializableAttribute” on the object	Functionality

sitecore®

oshyn
POWERING INNOVATION™

1.888.483.1770 | www.oshyn.com | newbusiness@oshyn.com