



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



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

Copyright

Copyright © 2011 Oshyn, Inc. All Rights Reserved.

Restricted Rights Legend

This document may not, in whole or in part, be photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from Sitecore and Oshyn, Inc. Information in this document is subject to change without notice and does not represent a commitment on the part of Sitecore or Oshyn, Inc.

Trademarks

Sitecore is a registered trademark of Sitecore. All other company and product names are trademarks of their respective owners.



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

Table of Contents

Executive Summary	4
Scalability Best Practices.....	6
Templates Best Practices	9
Presentation Component Best Practices	12
Content Structure Best Practices.....	14
Media Assets Best Practices	16
Security Best Practices.....	17
Caching Best Practices	19
Workflow Best Practices.....	21
Development Best Practices.....	22
Acknowledgements	25
References.....	25



sitecore®

oshyn
POWERING INNOVATION™

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

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



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

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.	<p>Authoring Sitecore is recommended (by default) to be operating in "InProc" session mode.</p> <pre><sessionState mode="InProc" stateConnectionString="tcpip=127.0.0.1:42424" sqlConnectionString="data source=127.0.0.1;user id=**;password=**" cookieless="false" timeout="20"/></pre> <p>Enabling sticky sessions will cause consecutive requests of the user session stick with one server, avoiding any loss of state for the user.</p>	Content Editor Experience
2	<p>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.</p> <p>Assuming content process flow happens from Authoring to Delivery Servers, disable the upload watcher and deny script and execute permissions on upload folder.</p>	<p>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</p> <pre><add name="SitecoreUploadWatcher" type="Sitecore.Resources.Media.UploadWatcher , Sitecore.Kernel" /></pre>	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	<p>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</p>	Functionality



		or by implementing the <code>ISerializable</code> interface. There is a (new <code>CustomObject().GetType().IsSerializable</code> property of the <code>Type</code> class that can be used to identify whether the object is serializable or not.	
4	In case of Clustered Delivery Servers with sticky sessions disabled, then session storage needs to be other than In-Proc, storing session data in ASP.NET State Server, SQL Server, Scale- Out or Windows Server AppFabric. We recommend AppFabric or Scale-Out for redundant distributed caching.	In case of lack of sticky sessions in clustered delivery server mode, consecutive user requests may be served by different servers with loss of state, providing unexpected and unintended results.	Site Visitor Experience
5	Remove database connection to Master DB on Delivery servers	Master DB is the database for Authoring and Web DB is the database for Delivery Servers. Removing any such direct connection would avoid exposing unapproved, unpublished data to end site users and provides for a consistent content process flow from Authoring to Delivery Servers. Authoring server components should be disabled on the Delivery Server.	Security
6	On Delivery Servers, restrict access to Sitecore client interfaces	Authoring servers are the only servers where the client interfaces should be provided to enable authoring. By disabling this access in the Delivery Server environment, security is enhanced by guarding the editorial interfaces from being accessed by website visitors. In general, the authoring servers will be in internal network (restrictive environment), while the Delivery servers are in general in the DMZ exposed to the Internet.	Security
7	Use “ <i>Web Deploy</i> ” option in cases with servers in different domains, in order to propagate file assets from Authoring to Delivery Servers located in different	Web Deploy and DFS are the two options available to synchronize assets between servers. DFS is an option only for servers within a domain, and Web Deploy is the option for servers	Functionality



	domains. If servers are in same domain (such as Delivery Servers within the cluster), then use DFS facilities of the Operating System for file synchronizations.	on different domains.	
8	Use forwarding of security events approach to security caches expiration approach in order to have immediate updates to security caches in clustered environment.	This is required in highly sensitive environment, where user role updates or security restrictions should be enforced on changes.	Security
9	Make sure this is added in order to be able to synchronize the Lucene indexes across environments.	<pre><Engines.HistoryEngine.Storage> <obj type="Sitecore.Data.\$(database).\$(database)HistoryStorage, Sitecore.Kernel"> <param connectionStringName="\$(id)" /> <EntryLifeTime>30.00:00:00</EntryLifeTime> </obj> </Engines.HistoryEngine.Storage> <Engines.HistoryEngine.SaveDotNetCallStack>false </Engines.HistoryEngine.SaveDotNetCallStack></pre>	Functionality
10	Tune the logging level to the "ERROR" level on the delivery servers in general. Change to Debug level if required while debugging the application on the server.	<pre><configuration> <log4net> <root> <priority value="ERROR"/> <appender-ref ref="LogFileAppender"/> </root> </log4net> </configuration></pre>	Implementation
11	Follow further instructions as provided by Sitecore in their Cache and Performance Tuning webinar video.	This would provide for various caching and performance tuning options.	Performance



sitecore®

oshyn
POWERING INNOVATION™

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

Templates Best Practices

#	Practice	Rationale	Impact
1	Group the fields into logical sections based on business sections or data type sections thereby avoiding a huge form, and provide for a compartmentalized form, that can be collapsed as needed.	Better categorization\separation of data fields and sections for editing.	Content Editor Experience
2	Use Data templates for normal data entry templates. Use Branch templates when you have a predefined structure of items, sub-items to be created.	In scenarios, where a predefined structure of items is to be created, then branch templates are to be defined for such structures.	Functionality, Content Editor Experience
3	Chose names which are understood by business users. Display Name and Field Titles are the important ones that Sitecore uses.	Ease of use for Content editors	Content Editor Experience
4	Define separate fields for display name, breadcrumb title, and navigation title fields for a page item.	Item names generally are different from Item Display Names.	Functionality, Content Editor Experience
5	Avoid using a lot of RTE fields within the template and keep the templates small.	A lot of RTE fields cause performance impact when loading in page editor.	Performance
6	Use icons wherever applicable to facilitate visual differentiation	Use of appropriate icons will cause clear distinction of different content types visually.	Content Editor Experience
7	Place the section of fields that are modified frequently at the top of the data template or place them in order they appear on pages.	This would facilitate easy modification of the fields by Content editors.	Content Editor Experience
8	Have default values in the form of standard values.	This would provide for default values for the fields.	Implementation
9	Provide proper context sensitive help for the fields, and sections in templates.	This would provide for ease of use for content editor experience.	Content Editor Experience
10	Make the field names unique between sections, do provide a prefix, suffixes	XSLT and .NET code gets values by field name alone without reference to sections.	Implementation, Content Editor Experience



sitecore®

oshyn
POWERING INNOVATION™

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

	indicating types, for example Title Name, Product Name as field name to avoid conflicts in field names when the templates are inherited by another template.		
11	Avoid Circular Inheritance in templates. Example: C inherits B, B inherits A, A inherits C.	Circular reference would cause for infinite recursion.	Implementation
12	Avoid modifying templates in Sitecore/Templates/System section.	These templates are standard Sitecore templates and are part of core system.	Implementation
13	Centrally manage the layout settings, initial workflow, Insert Options in template standard values than in individual items.	Consider a scenario of applicability at item level with thousands of items. In such a case, any change in these settings would be a humungous task to make changes to these items. In a scenario when it is applied at template level, these setting changes can be easily changed at Standard Values.	Implementation
14	Use Tilda (~) character in source for image fields with caution, as this will allow the user access to the entire media library tree.	Restricting use of ~ and assigning to specific folder for asset elements would make it easy for content editor to upload to respective directory, rather than from being confused with option to select the folder to upload to.	Security, Content Editor Experience
15	Do not use “Blob” type field.	It is internal Sitecore use field.	Implementation
16	For Checkboxes, compare with 1 to check whether the checkbox is selected or not.	Sitecore stores a blank value; therefore, it is not right to check against NULL, String.Empty, or 0.	Implementation
17	Use TreelistEx in comparison to Treelist, in scenarios where you need read-only view and don't need much editing. Use TreelistEx if editing frequency is lower and need read only view of the source items.	Treelist will load all items when rendering while Treelistex will load items based on user interaction.	Implementation
18	Use Droplink and Grouped Droplink instead of Droplist and Grouped Droplist.	Droplist and Grouped Droplist Field Types store dropdown values by item name, while Grouped Droplink stores values by item GUID.	Implementation
19	Do not use the Internal Link, Layout,	These are deprecated – html, link, lookup,	Implementation



	Rules, and Template Field Source field types in your data template.	memo, reference, server file, text.
20	<p>Make basic template components, which then can be inherited by other templates to provide for foundation templates.</p> <p>This would provide for reusable and consistent data templates.</p> <p>For example:</p> <ol style="list-style-type: none"> 1) Create Breadcrumb template inheriting the Standard Template having Breadcrumb Title (Single-Line Text) field. 2) Create Navigation template inheriting the Standard Template having Navigation Title (Single-Line Text), Show in Navigation (Checkbox), and Show in Sitemap (Checkbox) fields. 3) Create PageTitleandText template inheriting the Standard Template having simple Title (Single-Line Text) and Content (Rich-Text) fields. 4) A Foundation template inheriting the Standard, Breadcrumb, Navigation, and PageTitleandText templates. 5) Homepage template inheriting the Standard, Navigation, and PageTitleandText templates <p>Page Link template will be inheriting the Standard, Navigation templates having link (General link) field.</p>	Implementation



Presentation Component Best Practices

#	Practice	Rationale	Impact
1	Store the layouts inside /Sitecore/Layout/Layouts/, sublayouts inside /Sitecore/Layout/Sublayout. You can create sub-directories within these as needed with added access restrictions.	This aligns with the Sitecore architecture and allow for access restrictions.	Implementation
2	Do not have more than 2 or 3 layouts per device or r site and handle the different structures using placeholders and sub-layouts respectively.	The layouts generally will be the Default Layout with content placeholders for: header, content, and footer. The content placeholder will generally bind to two column sub-layout or three column sub-layout based on presentation requirements.	Implementation
3	Assign the layout details in the standard values rather than on items or template definition item.	When applied at item level, any changes to the layouts would require a lot of effort to make changes to all items. As such, when applied at standard values it becomes easier for such modifications.	Implementation
4	If different items based on a template require different layouts, then create a new template that inherits from the existing template.	Templates are easy to duplicate and do not add any cost to the system. This would align with setting of layouts at template standard values.	Implementation
5	Use FieldRenderer object to render the fields on presentation.	This would provide the native CMS inline editing features for the pages. Use DefaultFieldEditor module from shared source for allowing for inline editing of fields that are not displayed.	Implementation
6	Sort the sub layouts containing the placeholders to be before the controls that bind to those placeholders.	Sitecore layout engine builds and binds the controls based on this order.	Implementation
7	Avoid using Layout Presets, unless necessary, by assigning layout details to the template standard values.	Presets are used only when the content editors are given a choice of different layouts.	Implementation
8	Caching options should be configured whenever the controls are used, based on the control definitions.	Refer to the Caching Configuration guide for more details on configuration.	Performance



sitecore®

oshyn
POWERING INNOVATION™

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

9	Use sublayouts, renderings, xslt appropriately.	<p>General rule of thumb for use of different presentation components:</p> <ul style="list-style-type: none"> a. Global.asax, pipeline processor, or event handler - Logic to be done on application start/end, session start/end, request process cycle, event cycle. b. Layouts – Page level code c. Xslt - Module level Code with simple\little logic, ex: breadcrumb d. Sitecore controls – Module level Code with simple logic, where separation of data from presentation is not required. <p>Sub layouts - Module level Code with complex logic</p>	Implementation
10	Site should be compliant to content authoring using Page editor and not require access login to Desktop interface of Sitecore for content editors.	This can be achieved by coding using Field Renderer object or by using sitecore tags like sc:Text, sc:Image tags, which provide for editing in page editor out of the box. The content editor ribbon in page editor can be toggled to display/not display based on need. In general, this should not be displayed and toggled to display when needed.	Content Editor Experience
11	Extend the existing page editor, content editor ribbons as needed	This can be done by creating page edit commands by extending Sitecore.Shell.Applications.WebEdit.Commands.WebEditCommand and registering them in the Sitecore Core database as available option in respective editor. This will minimize the number of steps Content Editors need to do in getting their job done. Identify repeatable tasks and automate the steps for ease of Content Editor.	Content Editor Experience

Content Structure Best Practices

#	Practice	Rationale	Impact
1	Content structure hierarchy in Sitecore is to be decided with great care, as this can impact performance of the site. As standard practice, we would recommend not having more than 25 items under any item node. Try to maintain the hierarchy in such a way that it reduces the number of children per node.	Some typical ways of division are: 1) Year -> Month -> Day -> item 2) Category -> Sub category -> Item 3) Authors -> Alphabets A-Z (first letter of Author Lastname) -> Alphabets A-Z (first letter of Author firstname) -> Author item These above mentioned ways are guidelines and may need to be further subdivided based on project specifics as to limit the number of nodes under each item.	Performance, Content Structure
2	Make sure the indexes are frequently updated.	This would provide for better and faster content editor experience.	Performance, Content Editor Experience
3	Identify the different roles and access restrictions for items in order to provide access based on minimal access requirements.	Application of access restrictions minimizes the number of items accessible to content authors, thereby reducing load times and providing faster editor experience.	Security, Content Editor Experience
4	Plan content structure based on site map. Place all the items which are accessed using URL as descendants of the web site item.	A site map of the site will influence the way the items are structured under site home item. Any folders appearing in the path from home item to the page item will appear in the URL.	Implementation, Content Structure
5	Turn off the visibility of Standard Values on content items, "Entire tree" view for Content Editors in general, which can be toggled as needed by them.	Turning off these would show only sections of the content tree and sections of the content item that the authors are responsible to maintain. This would impact the content editor experience and performance.	Performance, Content Editor Experience
6	Use Clones instead of Proxy items	Proxy items are being deprecated in Sitecore v6.5, and may be removed in future functionality. Clones allow to proxy an item/item sub-structure and allow for overriding the values.	Implementation
7	Identify the security related requirements in multi-site in one instance environment and design your site structure, media structure that	For example content editor 1 of site 1 should be able to edit and view site1 but not site 2 which exists in the same instance.	Implementation, Security



sitecore®



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

	meets the requirements of separation.		
8	Maintain only a few versions of each item in the implementation	Use Version manager or any similar code plug-in to perform this. This would reduce the number of items in the database and hence improve performance.	Performance
9	Separate the content that is to be managed versus the content that is transactional	Do not bring transactional content into the CMS. It is better to maintain a referential key to the external transactional database in the CMS and just maintain the content required to be managed in CMS. One example: user checkouts of products.	Implementation, Content Structure
10	<p>In case of multisites in one instance, separate the structures as follows and appropriately set the websites in web.config <sites> node.</p> <p>Templates structure: /sitcore/Templates/Global, /sitcore/Templates/Site1, /sitcore/Templates/Site2</p> <p>Layouts structure: /sitcore/Layout/Layouts/Global, /sitcore/Layout/Layouts/Site1, /sitcore/Layout/Layouts/Site2</p> <p>Sites structures: /sitcore/Content/GlobalData, /sitcore/Content/Site1, /sitcore/Content/Site2</p> <p>Media Library structures: /sitcore/Media Library/Global, /sitcore/Media Library/Site1, /sitcore/Media Library/Site2</p>	Make respective folders following the mentioned pattern within the site. This kind of site structure aligns with the Sitecore architecture and access restrictions can be applied at the item level to provide access to different site only to the respective site organization.	Implementation, Security, Content Structure



sitecore®



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

Media Assets Best Practices

#	Practice	Rationale	Impact
1	Use database as the storage mechanism for media in general.	Take advantage of the CMS capabilities on media library and to be able to publish assets from Master to Delivery server.	Implementation
2	Use File system in scenarios where the number of assets is large (running into Gigabytes)	This would increase the size of database, causing performance impact, and where the CMS provided facilities can be traded for performance.	Performance, Implementation
3	Organize the media assets in business user friendly way.	This would make it easier for content editors.	Content Editor Experience
4	Optimize the image assets as much as possible.	This would avoid large size images which cause impact on the performance of the rendered site. The media library options will allow for editing of images and providing for different renderings of the same image using Sitecore media library facilities.	Performance
5	Identify the frequency of editing of different images (Content Images versus Template Images) and the roles (Content Author versus IT) responsible for the editing, and design the placement appropriately.	This would provide for editability of respective images by Content Editors.	Implementation
6	In a multisite, one instance scenario, make sure you organize the structure in such a way to meet any restriction requirements related to access of different media library folders to different site authors.	For example: Media library/site1/Images Media library/site2/Images Media library/site3/Images	Implementation

Security Best Practices

#	Practice	Rationale	Impact
1	All non-implemented membership provider methods should throw non-supported exceptions.	It is a good practice to throw this exception for non-supported exceptions.	Implementation
2	Use UserSwitcher wherever required instead of SecurityDisabler when editing programmatically.	This would provide for switching to particular role rather totally disabling the security model.	Implementation, Security
3	Assign security rights to roles and not to users.	Assigning to users will make it difficult to manage security on items, as users may be moving between organizations and require changing the rights very often. Rather providing to role, you can provide that role to all the users who would need access to that item.	Implementation, Security, Content Editor Experience
4	Always use inheritance of rights when applying rights instead of explicitly applying access to each item.	The more granular and more specific the security requirements, the more work in order to apply the security restrictions on those items. In Addition, Explicit Denial of access cannot be overridden in any way. Instead, if inheritance was used, then it could be broken and allowed for denial for some roles and denial for other roles. For example, User AB with role A and role B will not have access to an item that has explicit deny of access set on item for Role A, even though role B was provided access to the item.	Implementation, Security, Content Editor Experience
5	Use locally managed domains in the case of a multiple site implementations in single Sitecore instance.	This will separate the domains and will provide for each site seeing its own domains, rather than all the domains in the system. If not created locally, and as part of Sitecore domain, all sites will be able to view all domains.	Implementation
6	Create the roles in Sitecore Domain instead of specific domain.	This will make them visible to all domains in the system. Make domain specific roles only if required, as you cannot change the domain once created in this way.	Implementation
7	Prevent .config, .xml, .xslt, .mrt files	This will provide for security of the files. This can	Implementation, Security



sitecore®

oshyn
POWERING INNOVATION™

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

	from being served by webserver.	be done by modifying the Metabase.xml to avoid serving this type of mime-types at a global level or at the site level	
8	Make sure /data, /indexes folders are not accessible to anonymous users and are outside of the website directory context.	This will provide security for the folders in these directories and the files resident in these directories.	Implementation, Security
9	Turn off anonymous access to /App_Config, /Sitecore/admin, /Sitecore/debug, /Sitecore/shell/Web Service	This will provide for security of the folders.	Implementation, Security
10	Deny Execute scripts Permissions and disable upload folder on the Delivery Server. On the Authoring Server, enable upload folder if the content authors use it to upload files.	This will provide for security of the upload folders. In addition, install the Upload Filter Tool module available on Sitecore developer network, which can be used to restrict a list of valid file extensions that can be uploaded to the upload folder. Do disable script execute permissions to avoid any scripts from being executed in the upload folder. Disable upload watcher by commenting the line <add type="Sitecore.Resources.Media.UploadWatcher , Sitecore.Kernel" name="SitecoreUploadWatcher"/>	Implementation, Security
11	Disable Sitecore Client RSS feeds or perform security checks of the articles in the feed.	This will avoid any sensitive information from being presented to the consumers of the RSS feeds.	Implementation, Security
12	Provide the minimal set of roles to user accounts for performing their duties.	I.e.: ensure you provide only authoring role to content editor and set access restrictions to items they are responsible for, instead of providing access to the whole content tree with elevated privileges. There are minimalistic content authoring roles, which provide more restrictive authoring options for users who are less Sitecore savvy.	Implementation, Content Editor Experience, Security



sitecore®



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

Caching Best Practices

#	Practice	Rationale	Practice Impact
1	Combined size of all the caches should not exceed available memory.	Exceeding the memory would result in out of memory exceptions.	Implementation, Performance
2	Use Sitecore Debugger, Firebug, /Sitecore/admin/stats.aspx, and /Sitecore/admin/cache.aspx to measure the performance of the sites.	This will fine tune the caching layer and measure the performance of the site.	Implementation, Performance
3	During cache monitoring, if the Delta value keeps changing consistently with the size of the cache being more than 80% of the max size limit for that cache, then consider increasing the max size for that cache by 50%.	This will provide better performance and stabilization for stabilizing the cache store.	Performance
4	Tweak Pre-fetch cache appropriately by caching key items deemed to be required for the site.	Pre-fetch cache is the only cache that is pre-populated by Sitecore, tweaking this cache would enhance performance by retrieving and caching the relevant cache items. For example: identify the items which you would want to be retrieved, and place the GUID of the source item, and Sitecore engine will retrieve the sub items when the app starts up. However, it would impact the startup time of the application.	Implementation
5	Tweak the different layers of caches for your application and perform the respective measurements to see the optimal settings for your site.	<p>There are multiple levels of cache that can be tuned for better performance. Make changes at these levels and observe its impact on the site performance.</p> <ol style="list-style-type: none"> a. Database level: <ol style="list-style-type: none"> i. Pre-fetch Cache ii. Data Cache iii. Items Cache iv. Paths Cache v. Standard Values Cache b. Website level: <ol style="list-style-type: none"> vi. HTML Cache 	Implementation

		<ul style="list-style-type: none"> vii. Registry Cache viii. ViewState Cache ix. XSL Cache x. FileterdItems Cache 	
		<ul style="list-style-type: none"> c. User level: <ul style="list-style-type: none"> xi. IsUserInRole Cache xii. UserProfile Cache xiii. AccessResult Cache 	
6	Sitecore provides recommended values for Data cache, Items cache to be around 150MB on Delivery Servers.	These are general guidelines and you should start with these and fine tune for your application requirements based on monitoring the caches.	Performance
7	In case of usage of custom caching solution being used in your application, use the publish:begin or publish:end events of Sitecore to handle eviction strategies, based on updates to Sitecore item that is cached.	Cache eviction strategy will facilitate updating the site with the latest changes.	Implementation



sitecore®



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

Workflow Best Practices

#	Practice	Rationale	Practice Impact
1	Allow the approver the option to publish on demand if there is a needed situation, but general business process flow should be to publish on a scheduled nightly task.	Publish operation clears the html cache of the site on the servers, which would impact the load times and performance of the site. As such, it is better to not do auto publish on approval, but rather have a scheduled nightly publish to avoid any caches being refreshed frequently. The ability to publish on demand would still be possible in demanding situations.	Implementation, Performance
2	Avoid setting email notifications per item publish.	If set, it will generate lot of emails when items are being published and will flood the corresponding email inbox.	Implementation, Content Editor Experience
3	Identify the item types that would require undergoing workflow process and having those item type templates inherit the created workflow template.	This would provide for workflow for those items.	Implementation
4	Work with functional roles rather than specific users in organization, such as Business Unit Content Authors role, Business Unit Content Approvers Role rather than assigning specific user of the business group for approving.	Users may join or leave the organization making it difficult to maintain the system, when the workflow process uses specific users instead of roles of the organization.	Implementation
5	Have a preview environment to preview the content, as it would appear on publish.	This would help the approvers.	Implementation, Content Editor Experience

Development Best Practices

#	Practice	Rationale	Practice Impact
1	Set up continuous integration environment process flow for the project in order to have scripted compilation, coding style checks, run through set of unit tests, merge\replace different environment specific config tokens, and deploy to respective DEV, QA and PROD environments.	Continuous integration scripted environment provides for multiple releases to multiple environments at the click of the button and is standard development practices.	Implementation
2	Content Editing should be done on the master database, and then pushed to Web database.	This would provide for data flow from master to web database always and one point of content editing and smooth flow of data changes on publish from master to web.	Implementation
3	Avoid any content editing happening on the Delivery Servers.	In case of requirement to have workflow attached to website visitor generated content (ex: comments), post that content back to the authoring server and have it follow the workflow process.	Implementation
4	Be aware of the context, database, language, security restrictions of the items that are being programmatically modified.	Use UserSwitcher instead of SecurityDisabler object when modifying items programmatically, in order to be aware of the context and to restrict the editing options based on the roles of the user who is editing.	Implementation
5	Site context should be considered when programmatically creating links in a multi-site or multi-device scenario.	In multi-site environment without this setting, there would be conflict in the urls generated. Set the Rendering.SiteResolving in web.config to true in such scenarios.	Implementation
6	Disable View State for pages which do not require having one.	This reduces the size of the page, thereby improving the load times and the performance of the site.	Performance
7	Use the general best practices of development as applicable to C# with tools like Resharper, and StyleCop.	This would provide for standardized coding practices across the team as well as with .NET coding standards.	Implementation



sitecore®

oshyn
POWERING INNOVATION™

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

8	Identify the controls that can be cached and appropriately assign the values for caching the markups.	This would impact the load times and caching of those controls.	Implementation, Performance
9	Log debug, error statements to log file, or to Sitecore log file, to be able to check these logs for errors.	Logging level should be set at ERROR level on the servers in order to avoid huge logs. Also backup and delete the logs on the server on a monthly\timely basis.	Implementation
10	Use GUID's instead of Sitecore paths where possible. If paths, then consider the differences between Fast query results versus normal query results.	This would impact performance of the site. It is acceptable to have set of GUID's stored in a static class to be available for use within your solution. However, make sure that these GUID's are propagated between environments by using Sitecore packages for transfer of items between environments.	Performance
11	Encapsulate the Content types or templates using one of object mapper modules available at shared source, to be able to address as Product.Name instead of productItem.Fields["Name"] all over the code.	This encapsulation will help provide for having validations and accessing fields at one place and centralize the mapping layer at one place. However, when using such object mapper modules, make sure it is doing lazy loading of objects or simplistic loading as needed and not aggressively loading objects, as it would impact performance.	Performance
12	Use Sitecore Rocks plug-in in Visual studio for development.	It provides for a lot of features and easier development environment with Sitecore within Visual Studio IDE.	Implementation
13	Place the item in editing mode before it is edited programmatically.	This would provide for locking the item facilitating the edit and avoiding inconsistencies.	Implementation
14	Use Hooks, Event Handlers, Pipelines and Processors appropriately.	Hooks are used for minor initialization tasks when application starts up, Event handlers for registering to specific events for performing custom logic to be invoked on event occurrence, Pipelines and Processors as to intercept processing and perform custom logic as needed.	Implementation
15	Separate the configurations of Sitecore from the custom configurations being added as part of	Any new configurations can be added as include configuration file in App_Config\Include directory, providing for separation of Sitecore provided	Implementation



sitecore®

oshyn
POWERING INNOVATION™

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

	the solution.	configurations from solution specific configurations. Creating Patches or additional include configuration files is the way to implement.	
16	Remove the .aspx extension for the page URL and .ashx extension for the media items.	This will help SEO.	Implementation



sitecore®

oshyn
POWERING INNOVATION™

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

Acknowledgements

I would like to take this opportunity to thank my colleague [Christian Burne](#) for initiating this idea to document the practices and share with the rest of the world the way we work with Sitecore and the things to consider when developing with Sitecore. Thanks for his feedback on various parts of this document and improving it a lot.

References

- http://oshyn.com/_blog/Web_Content_Management/post/Hudson-NAnt-NUnit/
- http://oshyn.com/_blog/Web_Content_Management/post/Hudson-Parameterized-Builds-with-NAnt/
- http://oshyn.com/_blog/Web_Content_Management/post/Enterprise-Sitecore-Continuous-Integration-Configuration-Management-Part-2/
- http://oshyn.com/_blog/Web_Content_Management/post/NET_Distributed_Caching_Solution_-_Part_4_-_Configuring_Sitecore_with_Windows_Server_AppFabric_Distributed_Caching_Solution_and_Cache_Eviction_using_Sitecore_Events/
- Documentation provided at <http://sdn.sitecore.net/Reference/Sitecore%206.aspx>
- John West Blog - <http://www.Sitecore.net/Community/Technical-Blogs/John-West-Sitecore-Blog/>

About Oshyn

[Oshyn](#) is a technology services and product provider, with a reputation for delivering innovative solutions for the web, mobile devices and enterprise technology platforms. Oshyn is known for its expertise in implementing websites and creating products for the most popular Web Content Management Systems and platforms in the marketplace including [Sitecore](#), [EPiServer](#), [Jahia](#) and [OpenText](#).

Headquartered in the Los Angeles metropolitan area, Oshyn's [growing client list](#) includes Coca-Cola, Electronic Arts, Epson Electronics, Fordham University, Lexus, Mars, the



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

National Education Association, Sapient, Jamba Juice, California Teachers Association, Southern California Edison, and Volkswagen.

About Sitecore

Sitecore redefines how organizations engage with audiences, powering compelling experiences that sense and adapt to visitors both online and in-person.

Sitecore 's leading Content Management System software is the first to cohesively integrate with marketing automation, intranet portal, e-commerce, web optimization, social media and campaign management technologies. This broad choice of capabilities enable marketing professionals, business stakeholders and information technology teams to rapidly implement, measure and manage a successful website and digital business strategy. Businesses can now easily identify, serve and convert new customers with Sitecore's Digital Marketing System, part of its encompassing Customer Engagement Platform.

Thousands of public and private organizations have created and now manage more than 32,000 websites and digital experiences with Sitecore including ATP World Tour, CA Technologies, General Mills, ESPN Rise, Heineken, ISS, Lloyd's of London, Microsoft, Omni Hotels, Siemens, The Knot, Thomas Cook and Visa Europe.

About Prasanth Nittala

During his seven years in enterprise technology, Prasanth Nittala has implemented various Content Management Systems into the websites of clients in shipping, Customs Brokerage, HR Processing, Higher Education, and supply chain industries. His experience is a blend of strong academic background with solid industrial experience in many areas. Since 2009, he has been implementing websites using Sitecore WCM. He has management experience in leading big development teams and has successfully completed numerous projects on time. Prasanth adds value to the projects by understanding the business requirements of client, assisting them in analyzing business processes, and by providing and implementing technological solutions to support the evolving or existing processes.



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