Sitecore Worst Practices Blog Series

This document was compiled by Peter Prochazka from blog posts made by legendary John West.

Original blog posts can be accessed from this site.

Contents

Sitecore Worst Practices Blog Series 1
Project Management and Architecture 3
Configuration 4
Presentation 5
Information Architecture 6
Security 7

Project Management and Architecture

Suggestions to avoid worst practices in project management and solution architecture include:

- Don’t neglect to identify detailed functional requirements.
- Don’t neglect to read the Sitecore documentation and participate in the Sitecore developer online community.
- Don’t build excessive email notification or otherwise overcomplicate workflow.
- Don’t try to automate an existing manual process or a broken process.
- Don’t release a finished product and expect to get user acceptance without involving users early in the project and making project-specific training available.
- Don’t build features for or plan an architecture that requires content management and content delivery on the same system. Except in load-balanced content management environments, only one instance should write to the Master database.
- Don’t be afraid to extend Sitecore with validators, command templates, custom editors, and other features. No CMS can provide every feature you require, nor can any default implementation expose every possible parameter.

Configuration

Suggestions to avoid worst practices in configuration include:

- Don’t forget to configure the browser, such as to allow pop-up windows. See the Browser Configuration Reference.
- Don’t leave raw field values or standard fields showing.
- Get rid of the .aspx extension for content and the .ashx extension for media. Pick one of the techniques described in the guide to Dynamic Links in Sitecore to get IIS to use ASP.NET to process the required URLs. For media, set the Media.RequestExtension setting in the web.config file to an empty string.
- Avoid updates to system templates (/Sitecore/Templates/System//*). When necessary, implement your changes in custom templates, and specify those as base templates for the system templates.
- Don’t let media URLs begin with tilde (~). Media URLs that begin with the tilde character can result in multiple URLs for media items, sooner exceed IIS path length limits, and could result in performance issues. For more information and a potential solution to this issue, see the MediaURLTransformer Sitecore Shared Source project.
- In the web.config file, don't define custom /configuration/sitecore/sites/site elements after the default site named website; configure additional managed web sites before this default site.
- Update 5.May.2011: For elements within /configuration/sitecore, don't update the web.config file. Instead, add a config include file to /App_Config/Include.

Presentation

Suggestions to avoid worst presentation practices include:

- To maximize consistency and minimize maintenance, do not apply layout details to individual items; apply layout details to standard values.
- Avoid ASP.NET master pages by using the layout engine. Master pages can lead to challenges reusing content across multiple devices and reusing presentation across multiple languages.
- Don't use placeholders excessively. Use static binding when appropriate, and use placeholders for regions of the page that present different components at different times, such as on different pages that share a layout.
- Don't forget to enable caching of renderings and tune caching in general. For more information about caching, see the Presentation Component Reference and the Cache Configuration Reference.
- Don't access RTE field values directly. Use the <sc:html> XSL extension control or the sc:field() XSL extension method instead of the sc:fld() XSL extension method. In .NET renderings, use the FieldRenderer web control or the renderField pipeline.
- Don't expect Sitecore to pass the data source to sublayouts. For more information, see How to Apply Data Sources to Sitecore ASP.NET Presentation Components and Cascade the Data Source of a Sublayout down to Nested Controls.
- Don't resize images from the media library using simple HTML attributes. Use the corresponding features that resize the image on the server.
- Don't try to do everything with XSL. Use sublayouts and web controls where appropriate. Some solutions use sublayouts for everything, with no XSL or web controls. The last chapter of the Presentation Component Reference describes some considerations for choosing a presentation technology.
- Don't cache the same output twice. For instance, if a sublayout contains another rendering, don't configure caching for both the rendering and the sublayout. For maximum flexibility, configure caching for the renderings; for maximal reuse, configure caching for the parent sublayout.
- Avoid caching at the user level. There may be cases where you can cache at the user level, such as sufficient memory or a relatively small concurrent user population.

Information Architecture

Suggestions to avoid worst information architecture practices include:

- To provide better support for CMS operations such as move and rename, avoid referencing items by path, and instead reference items by ID. Present paths in the user interface rather than IDs.
- Do not allow a data template to contain multiple fields with a single key. If absolutely needed, you could still refer to the fields by GUID.
- Do not allow two children with the same key under a common parent.
- Avoid thousands or even hundreds of children under a single item, considering that the number may grow over time. The number of children beneath an item affects performance and usability similar to how a large number of subdirectories and files affects file system performance and usability.
- Avoid iterating large branches, such as with a Sitecore query, by using the descendant axis or through item recursion. When necessary, use a search index instead. Be sure to use the latest search APIs.
- Avoid branch templates for structures that can vary over time. Items created from a branch template do not reflect structural updates to that branch template.
- Minimize use of and features in the Rich Text Editor. Consistency leads to usability and usability leads to repeat visitors. Content authors should focus on content, not presentation and especially not markup.
- Don't store user-generated content and profile data in a content database. Store this information in the profile database or elsewhere. Store data in the CMS if it requires CMS versioning, security (other than user-level), workflow, publishing, or other CMS services.
- To avoid hard-coding, use The Sitecore ASP.NET CMS Configuration Factory and create classes such as Sitecore.ItemIDs.
- Don't create too many fields in a single data template or performance and usability will suffer. Use descendant items instead.

Security

Suggestions to avoid worst security practices include:

- To reduce maintenance, avoid applying access rights for users in favor of applying access rights for roles.
- To prevent denial from overriding allowance, avoid denying access rights in favor of breaking inheritance and granting access rights.
- Don’t forget to remove or change the password for any default users, but remember to create a new account with administrative rights before removing the default admin user.
- Encourage users to access the Page Editor, and avoid granting permission to the Content Editor and especially the Desktop.
- Don’t give users any more rights than are necessary, and especially limit use of the administrator role.
- Don’t create a large number of roles that each contain a single user. This might be one of the only cases where user-level access rights are appropriate, with consideration for managing those rights when users leave the system.
- Don’t log in as an administrator unless you must. Create a separate account for your general work.
- Don’t leave administrative pages such as those under /sitecore/admin unprotected.