**Webinar 24 - Best practices when moving into
Kentico and Save Time and Money**

**(Final Draft)**

**High everybody**

**Introduce Self and Role**

**What I want to do today is to convey some Best Practices that provide faster development, Lower costs and a better understanding of some of the features to take advantage of when converting… or developing a website in Kentico CMS.**

**Many consultations, Many ways to do things in Kentico.**

**If you have any questions, it would be better at the end of the presentation, but if you want to understand something more there is a question box in the chat window**

**Kentico Developer Styles:**

Portal vs ASPX vs Portal + ASPX (Hybrid)

* What is the difference?
	+ - Portal – Customize Webparts Not Templates
		- ASPX – Design in UI does not exist and Content areas are a “magic box”
		- Portal + ASPX – Design is Locked to Developers, Content to Editors
	+ When I use them?
		- Portal Development Style – I use the most as the clients have the most flexibility
			* Sites where the end-user will maintain everything after Go-Live
			* Sites are for Non-Technical end-users
			* Code customizations can be done with the UI/Custom Weparts etc…
			* Faster site delivery using a mostly out of box “common“ experience
			* Learning how Kentico “works” non-technically
			* Can still be complex and as technical as ASPX builds
		- ASPX Development Style – Used on a case by case basis
			* Slower Delivery time, unless the team “knows” the API and components
			* Completely Flexible as most of the Kentico code is accessible
			* Complex situations that can’t be done as custom Webparts
			* You don't like design modes or visual programming, but like writing your code by yourselves
			* The page templates are physical files must be completely implemented by you, including controls and the master.
			* Design, Data Structure changes are mostly done in Studio vs UI
		- Portal+ASPX Development Style
			* Combines the standard architecture and development process of ASPX templates with the flexibility and user‑friendliness of the portal engine
			* Allows Design Tab Access to end-users, but limited to exposed controls
			* Edit the page or webpart design via the UI vs code changes
			* More complex than Portal alone
			* Must maintain file based page templates
			* Must maintain up to 2 master pages (CMS root and File based)
	+ It's about the User, Stupid – Think of the content maintainers after Go-live
		- Always consider the end users staff, technical ability and access
		- While ASPX will make the client reliant on your dev team, do you want that?
		- If the client requested KenticoCMS, then unless there is some restriction, use the Portal method, because it’s more work to change after he fact
		- What is the end-user expecting? If you demo in Portal, build in Portal
		- Keep in mind that there really are no limits to either development method

**Content Storage Best Practices:**

Where to best store content in Kentico and why ?

* + (Settings Based) File system, Database or both
		- Store files in file system:
			* Best performance
			* Requires the ASP.NET User Modify rights to file system
			* Does not full-text search uploaded files
		- Store files in the Database
			* Worse performance
			* Allows you to use full-text search in uploaded files
			* No rights need to be adjusted on the file system
			* Allows you to easily backup the uploaded files as part of the DB
		- Store in both the DB and the File system
			* Combines the advantages of both options
			* Same performance as the file system-only option
			* Full-text search is available
	+ (Content Storage) Media library, Content Tree, App\_Themes, etc
		- Media Library
			* Pseudo File based – SQL still called for access
			* Better Performance
			* Can be Staged, but file level changes are not tracked
			* Can be used as content or set public
		- Content Tree
			* Database based – depends on settings
			* Good Performance – But, up to 3 DB queries to fetch content
			* Easier for Editors
			* Virtually no setup
		- App\_Themes
			* No queries
			* Best Performance
			* Difficult for non-technical types
			* Accessible in some cases via the UI
	+ How I usually use these
		- Store files in file system and database
		- App\_Themes for design assets not accessible to editors
		- Content Tree for Assets related to content, where not attached
		- Media Library for
			* Large amounts of PDF’s
			* Image Gallery (unless search is needed, then a custom doctype)
			* Video files
			* Common assets to many editors
	+ Versioning without workflow – Why enable it?
		- Easy ability to roll-back during the development stage
		- Easy for end user to roll back or compare in production
		- Has a small performance hit

**Master page, Home page and other types of content pages, Best Practices:**

1. How to modify Page templates, Webparts, Doctypes safely
* Understanding Template inheritance & Templating in Kentico
* Importing Packages into a working Kentico Site
* Multi-Site issues to be aware of
	+ When/Why to use doctypes to store the content
* Using custom tables to guide end users that add new content
	+ Saving time, Use the little Known Kentico Import Tool to Bulk Create Pages

**Photoshop to a Kentico Friendly format - 4 steps to save time, effort and money:**

1. You've finalized The Design... Now What? Simply break it down!
	1. Look at the design/wireframe knowledge etc…
	2. Narrow down the needed Templates vs the assumed Templates
		1. Example: 60 Page Template can be boiled to 7 Actual Templates
	3. Use Kentico to your advantage!
	4. Webparts and transformations can be more dynamic
	5. Use various Settings, C# Code, K# and Macros where possible
		1. Roles based
		2. Code Behind
		3. K# / Macros
2. Offload It! Save time and money using little know resource
	1. Designers / Design Firms are costly - Use them for their talents
	2. Chopshops are awesome: Good, Low Cost Photoshop Conversions Overnight
3. Whether, Photoshop conversion, a bought Template design or In-house build, Break it down to common design sections
	1. Post Wireframe
		1. Do you have page templates that are the same layout/structure?
		2. Can you combine similar content pages to use the same layout but different Webparts?
	2.
4. Add Data via Doctypes & Webparts, Content Pages, Assets

**Design Touchup – Simplify the process**

* Kentico CSS can be complex
	+ Stored in multiple files
	+ Stored in the file system
	+ Stored in transformations, Webparts etc..

Let’s make it easier to edit using 1 low cost tool – Stylizerapp.com

**IF TIME RUNS SHORT:**

**Common Pitfalls seen in the field after 2 years of working with partners and clients:**

* It's a .NET solution, just open in studio and edit it!!! RTFM - Read The Friendly Manual (please)
	+ Upgrades, Patches who needs them
* A CMS in the CMS!?! – A nightmare ASPX thing – no manual needed, I’m an expert
	+ Catal Example
* Fancy Redirection - Billing clients for confusion or Creating support headaches
	+ SEF Example
* Store Content in the ASPX Templates - Why have Kentico?
	+ SEF Example
* It's the root page, why you shouldn't store Home page content
	+ Inheritance and pathing hell
* Getting Started Consulting – Why it’s useful
	+ 5 Months to 14 Days delivered (Run example)

**NotesWhere to go from here:**

* MetaData on pages
* Taxonomy
* Tags
* Link Check Tool
* Validation Tool