

Deliver Now! Methodology

For Implementing Websites Using
Kentico CMS for ASP .NET

Deliver Now! Methodology

For Implementing Websites Using Kentico CMS for ASP .NET

Author: Kentico Software
Date: February, 2011
Release: Revision 1
Language: English

The contents of this document are property of Kentico Software. Copyright © 2004-2010 Kentico Software. All rights reserved. All other brand and product names are the property of their respective holders.



Kentico CMS
Unlimited website possibilities

www.kentico.com
sales@kentico.co

Kentico Software US
379 Amherst St, #375
Nashua, NH 03063, USA

Phone: +1-866-328-8998

Introduction

If you have ever been involved in web development project management or have implemented a Kentico CMS project, you probably know how important it is to keep in mind all those little details that draw the line between a successful project and an implementation bad egg. To help you avoid pitfalls and uncover potentially weak places in the project during the initial stage of the development process, we have prepared the Kentico Deliver Now! Methodology.

The main goal of the methodology is to provide you—Project Managers, System Architects, Integration Specialists, etc. with an easy-to-follow guide that describes the Kentico CMS project development process step-by-step before the actual implementation takes place. It should give you all the important information you need in order to finish your projects faster, using less resources, matching higher quality products while keeping your budget reasonable.

The methodology is neither attempting to serve as new documentation material nor as the Developer's guide, therefore, it doesn't include the same technical information you can find in other Kentico CMS documentation. Anytime a link to the documentation could be beneficial, it is included, so you can easily navigate to information about the more technical aspects of the particular subject.

Each chapter is introduced by a set of simple questions related to the particular subject. These questions help you think about areas you might forget about. Moreover, based on your answers, you should be able to select only information which is essential for you. In addition, chapters dealing with specific subjects also include some extra materials like checklists or sheet templates.

We hope you will find the methodology a great help, especially when going for your first project, but particular parts should also come in handy to experienced teams.

Kentico Software Team

Process Overview

The life-cycle of a Kentico CMS Project could be described by a simple scheme as pictured below.



Figure 1 Kentico CMS Project Life-cycle

Each phase expects some information, data, and actions as input, while providing a kind of product as the output. It is important to think about this concept as a complex element. If you do not invest enough effort to cover a particular phase, it might influence all the work done during the later stages of the project life-cycle.

Looking at the project from a time-line perspective, it might be interesting to consider how much overall time is spent in particular phases of the cycle. There are many elements included within each phase that influence the total time devoted to that phase to some extent; however, if we consider a standard-sized project, the graph could look like the following.

1 Requirements

The Requirements phase is where all the necessary information should be gathered. It usually follows an initial stage where the project proposal is provided by the customer. This phase should not be rushed in any way. Try to get as much information as you can from the customer on what the goal of the project is and what functionality is required. All the input data gained at this stage tends to be used during the phases that follow - you will refer to this piece of information over and over again. You should be really thorough during the Requirements phase.

A. What is the goal of the project?

Try to categorize your project based on the size of the website, its purpose, target audience, position on the market, etc. To understand whether it is going to be a simple web presentation, a complex portal, extranet or intranet solution or something like a community portal is crucial for getting familiar with the customer's needs. Correctly identifying the type of the project will help you to set the ideas you are going to read about into the right context.

Also, get the information on the motivation behind the project.

- What are the main reasons the customer decided for a new website?
- What problems, weaknesses or blind spots should the new website banish?

B. What is the expected number of users/ page views during the peak load, what is the expected number of documents on the website?

It is important to know the approximate number of users you can expect to visit your website per day as well as the number of page views your website is supposed to handle. Moreover, try to agree with the customer on the number of users able to login to the member areas of your website. It does not really matter if those users should be end users accessing a secured area on the live site or editors/designers with the access to the administration UI of Kentico CMS. In other words, you want to get information about the load your website would face during the peak-load period.

Ask the customer about the website content:

- Is it going to be static or dynamic content?

- How many documents do they plan to include within the website (hundreds, thousands, etc.)?
- Does your client plan to create multiple websites sharing the same (re-usable) content?

As you will see later, you are going to build the whole website structure around the information gained in this phase. In addition, plenty of the system settings are more or less dependent on the conclusions that you come to here.

C. What is the structure of the website and what types of content will be published?

As already mentioned above, you need to gain more details on what type of content will eventually be placed on the website and displayed to the visitors. The goal is to find out whether the built-in document types could serve you well or if custom ones would need to be created.

- Does your customer want to include some kind of document repository (for documents, video and audio files, images, etc.)?

Furthermore, ask about the customer's ideas and expectations on the website structure:

- What are the main sections the customer wants to divide the website in?
- What items should the navigation contain?
- Is there any requirement for implementing membership areas or restricted sections on the website?
- Do they plan to display personalized content (changing the layout, styles and look of the page based on the current user)?

D. Which web standards should be followed in terms of accessibility and coding?

If you care about accessibility of the website, you may want to try to comply with some specific accessibility standard or better yet a group of standards. It ensures your website is available for everyone including people with different kinds of disabilities. Thus, think about implementing the website to meet the requirements defined by the Web Content Accessibility Guidelines (WCAG), the User Agent Accessibility Guidelines (UAAG) or Section 508 (applies to the United States). Visit the

W3C [accessibility page](#) or [Section 508 home page](#) for more details on these standards.

You may also want to consider relying on coding standards that are available out there. In particular the HTML, XHTML, XML based standards, mobile client standards, web service standards, etc.

E. Which products and technologies will be used?

If your project requires integration with some 3rd party components (Telerik, ComponentArt, etc.), external software (ERP, CRM, custom .NET applications, etc.) or some other technologies (Adobe Flash, Microsoft Silverlight., etc.) you should keep track of these as well. It will play an important role when you start to work on the development plan. Based on the type and complexity of the integration, your development stage may shrink or expand respectively. Ask your customer what functionality they want to use.

F. What is the content life-cycle? Who is responsible for the content management?

Editors and Contributors

Think about the content life cycle for a while:

- What type of users would be responsible for creating, updating and deleting documents?
- Should website visitors get a chance to contribute to the website?
- If the content will be subject to workflow, what steps should it go through before being published?
- What roles would be in charge of approval in each particular step?

Integration

- When you integrate the project with some 3rd party CRM or ERP system, does it call for spreading the same change all the way to the external system?
- What task needs to be accomplished in the external system as a reaction to document changes?

Staging

In some scenarios the content gets entered and published on the staging server first and then pushed to the live site using synchronization.

- Is your customer interested in isolating the staging and production (live) environment?

Archiving

You have created and published the website content. Imagine the website is already running for some time now and you want to archive documents. You should therefore agree with the customer on what is going to happen with the content once it passes its life-time.

- Should you remove it from the website completely?
- Would it be suitable to archive it instead and keep it hidden within the website?

G. What languages will be used for the content?

The amount of work involved in developing the website obviously grows with the number of different language cultures your project needs to support.

- Talk with the customer about what countries they plan to expand to with the website,
- What would be the differences in the content for particular language versions?
- Do they want to simply translate the text or are more significant changes planned?
- Should the look and feel of the website be modified based on the selected culture?
- What about website structure and page hierarchy for different cultures?
- Do they require translation of the administration UI (CMS Desk and CMS Site Manager)?
- Is the required language for the UI ready yet (available for download at the [Localization Packs page](#) on Kentico.com)?

H. What is the required availability of the website?

As with other on-line solutions, your project might encounter some difficulties running on the live server due to various reasons. It does not need to be related to the Kentico CMS website implementation or your custom code at all. It might be an issue with a web server, SQL Server, network, data center availability, etc. Thus, talk about availability with your customer as well. If you are not familiar with the meaning of the word ‘availability’ in web terminology, take a look at the simple [introduction post](#) published by Kentico’s Product Manager. Basically, ‘availability’ expresses a percentage of uptime (site is running and serving user requests) and downtime (site is unavailable) in a given year.

High availability (99% to 99.9999%) could be achieved and managed by facilitating advanced scenarios within your solution. You can consider including [RAID fields](#) to provide a disk striping or disk mirroring extension to your storage environment. Furthermore, you may want to consider enhancing the live environment setup to support [Failover Clustering](#) in order to provide means for automatic detection of server node failure. It could cover both web servers as well as SQL server nodes. On top of failover clustering, the [Log Shipping](#) technique may take place to support failure resistance and log propagation between multiple nodes. Other useful methods for gaining better availability include [SQL Server Replication](#) and often neglected Backup & Recovery.

Please refer to [Appendix A – Requirements Template](#) to get a list of all questions mentioned in the text above.

2 Analysis & Design

The Analysis & Design phase takes all the information delivered as the output of the Requirements phase. By applying various techniques, it transforms the requirements into so-called design units—use case diagrams, wireframes ([Appendix C – Website Wireframe Example](#)), the website structure draft at the beginning and page templates, document types and physical file organization at the end.

2.1 Content tree

Website content structure is one of the most important items on the project to-do list. The structure of the website has an impact on several aspects of the final version of the project.

A. Content structure and URLs and SEO

The organization of documents within the [content tree](#) defines the URL format of your pages. The page URL is formed from the page [document alias](#) (unique name of the page) and the document aliases of all the parent pages. During the [request processing](#), the SEO friendly URL gets rewritten internally and resolved to a physical ASPX page rendering the content.

- Design content structure to get URLs in the required format,
- Utilize [wildcard URLs](#) to use a single page for displaying the content of multiple documents to comply with SEO standards.

B. Content structure and navigation and sitemap

There is a close [relationship](#) between the content tree structure, navigation elements and the sitemap. You can decide what content tree nodes will be displayed as navigation elements in the website [menu](#). The [sitemap](#) is generated based on the content structure, ready to feed the search engines.

- Define the desired navigation elements.

C. Content structure and membership and restricted areas

To create any [membership](#) or [restricted area](#) within your website, include the area parent page in the content tree first and then place all documents underneath.

- Force authentication when accessing the area root page and subpages,

- Do not forget to setup the document level permissions accordingly.

D. Content structure and website performance

Understanding the [way documents are obtained](#) and displayed from the database could help you catch on to the importance of designing the website structure in the right way. The more documents exist under a single parent page, the more resources are used when searching for the right content underneath.

- Keep content categorized/ grouped/ structured based on some common information (e.g. months concerning news items, etc.),

Furthermore, if you ever need to alter the default URL path of a page, use the custom document URL path as your first option. You can change the URL path through [custom document aliases](#) as well; however, it would cause greater overhead during the request processing compared to a custom URL path.

- Use custom document aliases only in case you need to specify more than one URL per page,
- Always enter the primary (most commonly used) URL using the [custom document URL path](#).

NOTE: If you want to know more about URL processing and how it can be influenced by content structure, take a look at the following [webinar](#).

E. Content structure and sharing the content

If you plan to use a Kentico CMS instance containing multiple websites sharing the same content, you might want to use [linked documents](#) to create a copy of the original document on the target website. Any changes in the settings of the original document (page template settings, document URL path, etc.) are reflected by all occurrences of the document on other sites.

If you want to share only the document content and not the document itself, you may use viewer web parts (repeaters, data lists, etc.) to pull the data out of the source website and display it on the target site.

- Define what documents need to exist on multiple websites,
- Identify the content displayed by multiple websites.

F. Content structure and multilingual sites

You can specify different content structure for each culture assigned to your website. You could either [translate documents](#) from the default culture or create completely different content instead. The metadata of a document is shared between all the cultures; however, document specific information is unique for the given culture. To meet your SEO plan, you can use [culture specific URLs](#) for the same document.

- Translate documents from the default culture or create new documents for a specific culture,
- Use different URLs for documents of a particular culture to keep up with SEO standards.

G. Content structure and media files

If you plan to upload a large number of files onto the website, try to avoid using the content tree as a storage location for files. Otherwise, the content tree could grow immensely resulting in overall performance degradation.

- Store files in the content tree only when able to control the number of files uploaded,

Read more about storing files in Kentico CMS in chapter [2.4](#) [Organizing media files](#).

H. Content structure and user contributions

To allow visitors to contribute to the website, you may want to setup the [User contributions \(Wiki\)](#) module.

- Specify the target location for newly created documents and the source path for the documents available for editing,
- Combine membership areas and the Wiki module to allow only certain users to manipulate the website content from the live site.

2.2 Page templates

Once you build a comprehensive website content structure, you can start working on [page templates](#). Each page (document) in the content tree is based on some specific page

template. A page template defines the look and feel of the individual pages that use it—their page layout. A template also contains elements ([web parts](#)) responsible for the displaying of content. The [way content is displayed](#) on the page is also affected by the content tree hierarchy.

A. Design template analysis

You have got the design template, the index HTML page and the related CSS style sheet. Process the template in the following way:

- Mark the parts of the design template that should be shared over the entire website—use them for the [master page](#) template,
- Try to find some general [layout](#) pattern by analyzing the position of the content elements—two columns, two columns and menu on the left, three columns with a top submenu, etc.,
- Split the design template into zones displaying some kind of content—document data, images, banners, navigation, etc.,
- Make sure the content zones from the previous step fit the layout identified at the beginning.

B. Page template and the website structure

As far as page template [visual inheritance](#) goes, you also need to think about the [way the parent pages affect](#) the look of the underlying pages.

- Pick zones from the previous example to be displayed on all pages (defining the root template) and zones that will be part of child pages (document templates).

C. Page template and web parts

Try to categorize the content displayed by the page template based on the content type.

- Go through the content zones outlined earlier and define the type of their displayed content—image, calendar, logon form, current user data, banner, rotating news, newsletter subscription, blog preview, forum, polls, etc.,

- Search through the Kentico CMS [out-of-box web parts](#) and [modules](#) to find out whether any of them could be used for displaying the content of a particular type,
- If the zone displays content of a specific type—content generated by a 3rd party control—mark it down. You might need to develop a [custom web part](#) to provide this kind of functionality,
- If you are not sure whether the built-in modules and web parts fit your needs, do not hesitate to consult your requirements with [Kentico](#).

Take a look at the sample result of design template processing in [Appendix B – Design Template Processing](#).

D. Page templates and personalized content

To display personalized content to the user, you do not need to design multiple page templates. You can setup web parts to [display content in personalized form](#) using current user information.

2.3 Document types

Any type of structured data is handled by [document types](#). All documents in the content tree are based on some document type. A wide range of pre-defined document types is available for you; however, you can also create [custom document types](#) according to your specific business needs to store complex pieces of information within the CMS.

Document content is displayed using [transformations](#) defined for the respective document type. They are a combination of HTML and ASP .NET code. Transformations are used by all viewer web parts. Document content is pulled out from the database and dynamically injected inside transformation text just before a page is rendered.

- Make a list of all different content types displayed on the website,
- Go through the existing document types—look for any that could fit your needs,
- Make sure using custom table would not be a better choice than a custom document type. Read more about custom document types and tables in [C. Document types and custom table](#),
- The document type defines how content appears on the page. Think about various forms of displayed content.

A. Document type and security

Each document type relies on its own [security settings](#). Document type permissions allow you to control access to documents.

- Make a list of roles authorized to manage the documents of a specific type—read, create, edit, delete, etc.

B. Document type and workflow

[Workflow](#) support allows you to manage the creation, updating and publishing of website content. [Versioning support](#) allows you to store, view (audit) and roll-back previous versions of the content. In addition, the archiving support can be used to archive documents. Archived documents are no longer available on the website, but are still kept in the content tree. You may re-publish content later.

The basic workflow document life cycle is depicted below.

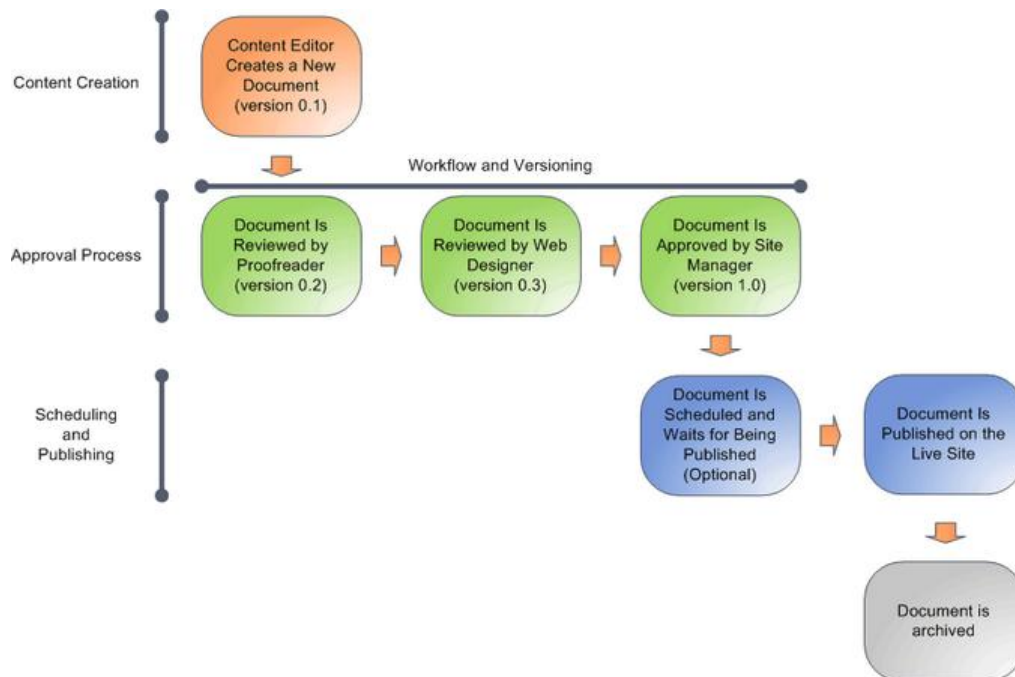


Figure 3 Example of a document life cycle when using workflow

Prepare custom workflow:

- Define a scheme for the content life-cycle,
- Make a list of steps that a document needs to pass through to get published,
- Define roles authorized to modify, approve or reject the document in each step,
- [Settings](#) allow you to apply the workflow only to documents in a specific content tree location,
- Consider enabling the [content locking](#) feature in order to prevent unintended overwriting of content changes,
- Decide whether un-published content should be archived or deleted completely instead.

C. Document types and custom tables

All documents are stored within the content tree. The tree structure may become inefficient for really large numbers of documents though. Thus, if your website is likely to contain hundreds of thousands of documents, you can use the [custom tables module](#) to store the data of a particular content type instead.

- Should the expected number of documents for the single website exceed 100 thousand consider using custom tables,
- Use custom tables to store content that does not rely on the tree hierarchy,
- You can [define custom tables](#) in the same way you define custom document types—by specifying custom fields, transformations, etc.,
- Custom tables can hold more than 100 million records.

2.4 Organizing media files

It is really important to choose the proper [way of storing](#) the media files on the website. The ideal storage type depends on the nature of the file (e.g. design image, media gallery image, media repository file, etc.) as well as the expected number of files.

You can also decide where your [files should be stored](#)—in a database, file system or even both. If possible you should store files in the file system. It gives you a performance boost; however, you must provide an adequate amount of free disk space.

You should consider the following:

A. Files in the content tree

Files are uploaded and stored within the content tree as CMS.File document type instances. Each file is represented by single record in the content tree. The size of the content tree expands with every file uploaded into the system. As mentioned earlier, if the content tree is unmaintained and gets too large, the CMS system may encounter a performance slowdown.

- Use the CMS.File document type for storing files in the content tree if you have control over the files uploaded by editors,
- The CMS.File type usually stores files used in multiple places on the website
 - **NOTE:** Best practice is to store all website design files in the appropriate App_Themes sub-folder,
- Retrieving a file stored as a document in the content tree requires additional resources,
- The maximum size of an uploaded file is limited by the maximum size supported by the SQL Server binary type,
- Do NOT use it for creating a file repository containing a large number of files,
- You can use the [bulk import tool](#) to upload multiple files at once.

B. Document attachments

Files can be stored as part of a structured document in the form of [attachments](#). This way, you can attach multiple files to a single document. Attachments are directly bound to their document and follow its life cycle.

- Files attached to a specific document are available only for this document,
- [Document level permissions](#) automatically apply to the attachments as well,
- You can specify multiple files for a single document with no impact on system performance,
- There are several types of attachments—[grouped](#), [unsorted](#) or [file field](#) attachments,
- The maximum size of an uploaded file is limited by the maximum size supported by the SQL Server binary type.

C. Media libraries

Whether you plan to create a file (asset) repository available to site visitors on the live site or are just expecting a large number of files to be uploaded to the website, you should consider using [media libraries](#).

- All files are physically stored in the file system,
- No limit on the number of files uploaded through media libraries—limited only by free disk space available,
- You can point a media library to an FTP location and upload new files this way,
- Files are organized in folders,
- Files are accessible using a direct path,
- Media libraries feature a rich [management UI](#),
- The files can be accessed from various parts of the system and used by multiple documents and pages,

D. Unmanaged files

Some files do not need to be managed by the CMS at all. Instead, you may store them in the web project theme folder as a part of the website design template (so files get exported along with the website) and accessed via a direct path. Unmanaged files can be various design files that won't be used for any purpose other than styling (including flash animations, etc.).

3 Development

Once you pass the Analysis & Design phase, it is time to get down to the actual development. This chapter discusses the most common actions and tasks concerning the development phase.

- First, you should reserve all resources required for project development-
- [3.1 Resources](#),
- To ensure a smooth and trouble-free development process, you may want to setup a team development environment as described in chapter
- [3.2 Team](#) development,
- A new CMS instance for the development environment may be installed using the [Web installer](#),
 - **NOTE:** The web installer creates a website project for you by default; however, you can [convert the project into a web application](#) if that is your preferred project type,
- You usually start the actual development by processing a design template delivered by a graphic designer: [3.3 Index template processing](#),
- Next, you process the static HTML and decide on the proper ways to display content, but now using the CMS: [3.4 Altering static HTML using web parts](#),
- You create templates for your pages afterwards: [3.5 Page template development](#),
- Structured, complex content data will be stored using document types. You can either use pre-defined document types or create custom ones: [3.6 Custom document types](#),
- If the out-of-box functionality does not cover all your needs, you may need to develop custom modules or web parts: [3.8 Customization options](#),
- When the content is ready, you may approach the website security setup: [3.10 Security](#),
- At the end you should double-check the website's configuration and settings to optimize and tune it up to get the best performance and stability.

3.1 Resources

There are several roles involved in developing a Kentico CMS project.

The Content Author, Content Editor and Content Publisher will be responsible for content publishing. Depending on organization, one person may cover multiple roles and perform all content related tasks.

The .NET Developer role is responsible for maintaining and extending existing site functionality. The Designer role on the other hand takes care of site-wide visual consistency. The Database Manager is in charge of the operational efficiency of the website, devoting time to optimization of the database.

The key roles are shown in the table below.

Role	Minimum Technical Expertise	Description	Required for Content Publishing
Content Editor	Word Processing, Web Browsing	Creates the Web content, determines what content is published, where it will be published and when it will be published	Yes
.NET Developer	ASP.NET	Maintains/extends and modifies WCMS application code	No
Designer	CSS, HTML, FLASH, Silverlight	Maintains/extends or modifies visual design elements	No
Database Manager	SQL Administration	Maintains/manages the WCMS repository	No

Table 1 Roles involved in the process of developing the website

3.2 Team development

Kentico CMS supports [team development](#) on a source code level using an external source control application (e.g. Microsoft Team Foundation Server- TFS). Setting up a team development environment for Kentico CMS requires more or less the same effort as for any other ASP .NET application.

If you need to allow multiple developers to work on the same project, you may setup the development environment as pictured below.

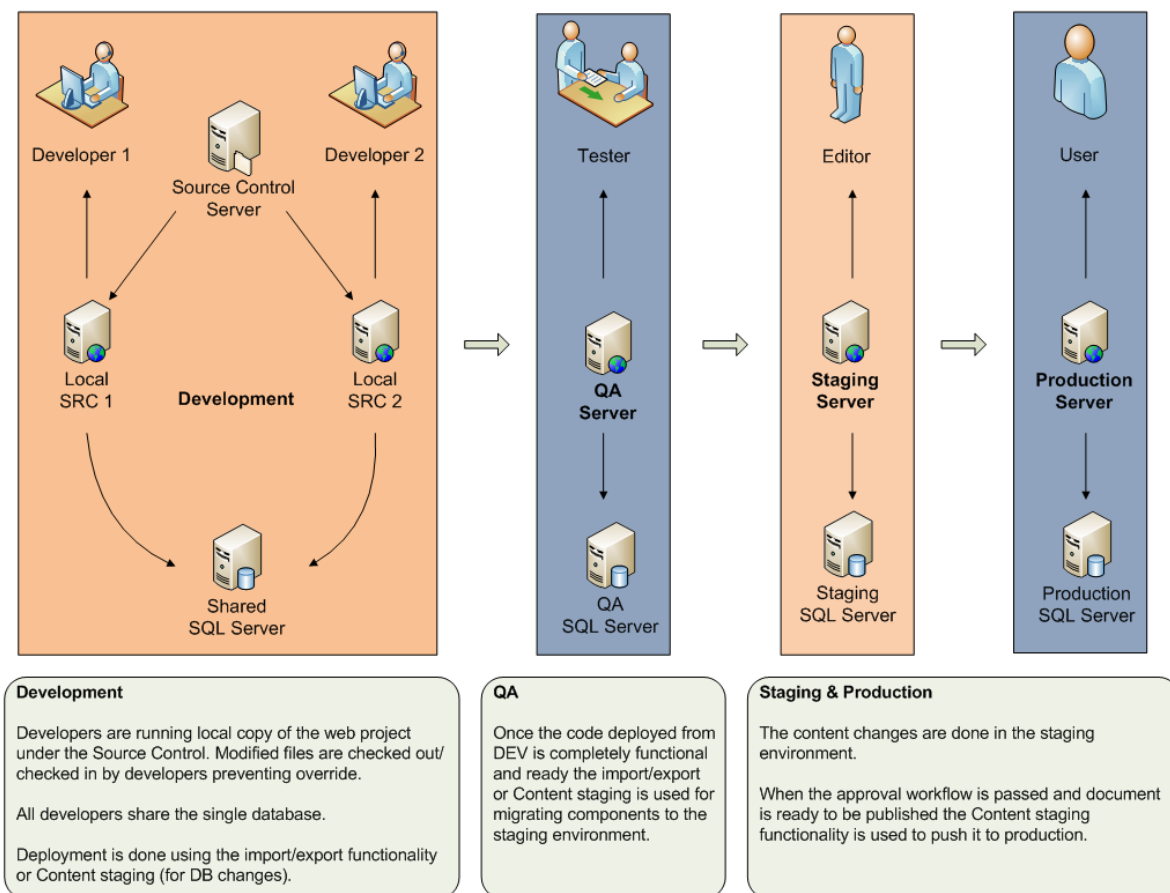


Figure 4 Team development environment setup

As you can see:

- Each developer has their own local copy of the project folder,
- The local instances (running on a local machine with properly configured IIS) point to the same shared database,
- New functionality is managed locally until the change set is ready to be deployed,
 - Modified code files are checked-out/ checked-in by developers in the source control application preventing unintended override,
 - Changes made to objects storing metadata in the DB are reflected for all developers,
 - [Web farm synchronization](#) may be used to notify all the development machines when a change requires the cache to be flushed,
- Once a developer feels confident about a set of changes, the code is submitted to the source control application (to spread over the entire development environment),

- Changes are then deployed to the QA environment (using [export/import](#))—the QA verifies the functionality,
- New features are transferred to the staging/production environment (using [available deployment options](#)),
 - If you want to manage all your content and new functionality in a staging environment and then push it to production, you may use [Content staging](#) functionality. It allows you to update content and make changes to the website on a staging server and then synchronize changes with the production environment. That way you do not perform any content oriented operations directly in production,
- For detailed information on deployment options, refer to section [5 Deployment](#).

Note that in the given scenario, developers use a shared database. This means changes done to the system metadata (or any object stored within the database in general) are not managed by the source control application. However, developers can use check-out/check-in functionality (available in the CMS UI) to avoid unintentional modifications.

Collision-free approach

Of course, to prevent the overwriting of changes completely, every developer may use a separate database. Although this approach ensures collision free development, changes made locally won't be visible by other team members till they are deployed to a shared database. You may find this approach useful, especially in scenarios where the new functionality being developed is independent on other parts of the system.

3.3 Index template processing

You kick-off the development by processing the design template coming from a graphic designer. Let's consider a design template to be a set of files (index HTML page, CSS stylesheet, design images, etc.) representing static version of the website home page (or any other page).

- Identify common content shared by all pages on the website (used for the [master page](#)) and parts of the page specific to certain sections,
- The goal is to specify the content and layout of [page templates](#) used for various website sections.

The actions involved in processing and transferring the index page to the CMS are:

1. **Populate master page code:** take the entire HTML code of the static index page and [place it](#) into the [master page](#),
2. **Apply design assets:** create a [new CSS stylesheet](#), copy the available styles into it and assign it to your website. Then upload all design files into the solution and update the HTML code/CSS style sheet references accordingly. Take a look at [2.4 _____ Organizing media files](#) for recommendations on where to store design files,
3. **Identify content used in the header and footer:** top logo, main navigation, search box, link directory, site map, etc. All of these will probably be shared by all pages on the website,
4. **Move the HTML code of the header and footer to the master page:** copy and paste HTML code common for the whole website and place it into the master page. Then add a new web part zone to the location previously occupied by content. Insert the Page placeholder web part into the zone to allow the content of sub-pages to be loaded inside the master page,
5. **Create the first page:** now you need to take care of page specific content (HTML code) used on the index page. First, [create a new page template](#). Grab the HTML code of the remainder of the index page (code that will be specific for this new page only) and copy it into the [page template layout](#) of the recently created page (let's call it home page),
6. **Review the index page in the CMS:** if you access the home page now you can see your index page displayed by the CMS (albeit still just static code). You should be ready to approach the next stage in the development process—replacing static code with web parts.

3.4 Altering static HTML using web parts

So far, you have defined the basic content for the master page and home page. However, the content of pages is still static. To make your website truly benefit from the CMS system, you need to alter the static code using web zones and web parts.

Do the same for all pages no matter whether it is a master page or a regular page:

- [Open the layout code](#) of the page,
- Search through the static code for enclosed HTML entities—lists, paragraphs, images, anchors, headings, selection lists, etc. For every entity, you need to decide whether you want to make it manageable using some web part. Otherwise you just leave it as part of hardcoded layout code,
- Each entity is usually part of a certain page element—main menu, logo, secondary navigation, titles, etc. You should check for out-of-box web parts that provide similar

functionality as the element that the entity belongs to. That way you move content management from the level of HTML code to a user friendly web part UI instead,

- At this time do not replace or manipulate with content (text, audio, video, etc.) data rendered dynamically (e.g. based on the incoming query string or some other condition)—you will most probably create a custom document type and use a viewer web part to output such content.

The process of transferring static content into the CMS is a routine task. It just requires you to know what you can or can't do with the pre-defined feature set. You can refer to the table below when transferring basic HTML entities/ elements into Kentico CMS. The table lists some of the most common elements and their CMS counterpart.

Static element	CMS equivalent
Links, headings	Text category web parts: editable text or static HTML, static text
Single image	Text category web parts: editable image or editable text, static HTML
Image gallery	Listing and viewers or Media library, Attachments category web parts (depends on the chosen file storage type, see 2.4 Organizing media files)
Editable content	Text category web parts: editable text
Structured content	Listing and viewers category web parts: repeater, datalist, grid
Form	BizForms module + BizForm web part/inline control
Logon form	Membership category web parts
Subscription	Newsletters category web parts

Table 2 Examples of static HTML elements and their equivalents in Kentico CMS

The information above represents just a really small fraction of available web parts. You can check a list of all web parts through the 'CMS Site Manager-> Development-> Web parts' section of the CMS UI. Moreover, you can see a complete list of web parts along with their descriptions in the [Kentico CMS Web Parts](#) reference.

NOTE: We recommend you to [contact our support team or consultant](#) before you make a final decision to develop a custom web part when you are missing some out-of-box functionality. Save unnecessary development time.

3.5 Page template development

Here are several important facts related to page template development that may save you some hard times figuring things out for yourself:

- Every page in Kentico CMS is based on [some page template](#),
- A page template consists of two parts: a [layout](#) plus [web parts](#) and their template specific configuration,
- Page template content (HTML markup, web part zones and web parts) is stored in the database (when using the [Portal Engine](#) development model). Web parts used on a template are stored in the database in XML form along with other template metadata,
- You may want to explicitly specify [which website sections allow the usage of a particular page template](#) (to limit the options of editors),
- Page templates can be defined as re-usable or ad-hoc,
 - A re-usable page template is stored under a unique name and is ready for use by other pages,
 - **NOTE:** Changes made to the template on one page influence all pages using the same re-usable page template. You can clone a template as ad-hoc for any particular page to avoid this behavior,
 - An ad-hoc template is used only by a single page [unless you save it](#) as a new template,
 - **NOTE:** An ad-hoc template used by a particular page is automatically removed when the page is deleted (unless you save the ad-hoc template as a new template),
- Despite having multiple different page templates, you may still share the [layouts](#) that templates use internally. Even for ad-hoc templates. Any modifications done to a shared layout are reflected by all templates using the same layout. Be aware of this fact,
- The template used by the root node of the content tree is always defined as a [master page template](#). Moreover, you can mark any other page template as a master page as well. That way, you will be able to reset the [visual inheritance](#) level for any nested page,
- If you mark a non-root page template as a master, all its child pages using the 'Inherit only master page' inheritance option will only display content from the nearest parent master page (no longer from the root node),
- The [hierarchy of pages](#) (using different page templates) influences the final look and feel of the displayed page,

- Do not create a new template from scratch everytime you need to make a minor change. You can [clone an existing template](#) instead and leverage its current settings and functionality,
- If a page is not displayed at all because of some serious error, you can alternatively remove the web part causing problems through the ‘Web parts’ tab of the template properties dialog,
- Take advantage of web part zone properties. You can easily hide a particular web zone on sub-pages, display zone content to a certain set of roles or only on specific document types, use [macros](#) to evaluate zone properties, disable view state, enable AJAX post backs, etc. This way you can to some extent manage all web parts in the given zone from a single place,
- You can [use multiple Page placeholder](#) web parts on your (master) page templates. Furthermore, you can display a placeholder only for specific document types and thereby include web parts defined by a child page on its parent page and more.

NOTE: Get familiar with macros by watching the following [webinar](#) and reading this [blog post](#).

3.6 Custom document types

The [concept of document types](#) in Kentico CMS allows you to store and organize structured data in the content tree hierarchy. When we were converting static HTML to web parts, we skipped content that changes under certain circumstances—dynamic content that is regularly added, updated and removed. The goal of this section is to point out important facts related to the development of custom document types.

You can [create custom document types](#) using the built-in wizard that guides you through the whole process. If you’re development oriented, take a look at how documents are stored within the CMS, interconnected with other parts of the system and what their [position within Kentico CMS](#) is.

The following points raise additional concerns related to (custom) document types:

General

- Once you define a custom document type, review all tabs available when editing it (General, Fields, Form, [Transformations](#), Queries, Child types, Sites, [Alternative forms](#), [Search fields](#)) to make sure you get the most out of your document types,

- A query may be defined both as query text or a stored procedure name (then you need to create a SQL procedure in the database) and use a transaction,
 - Use `##WHERE##`, `##TOPN##`, `##ORDERBY##`, `##COLUMNS##` macros to make certain parts of the query be resolved dynamically before execution. Macros are replaced by the actual values, usually passed from the property configuration of a web part,
 - **NOTE:** Use the 'Avoid automatic updates' option to explicitly specify that the system should ignore your query when updating queries after a field was added/removed for the document type,
 - **NOTE:** You should not modify the default queries unless you are pretty confident about the changes you are about to do. The default queries are used by the CMS to perform all tasks related to the document type. If you make an inappropriate change to a default query, you may break some functionality of the CMS.

You should also be aware of the following general facts about document types:

- **Document alias:** each document has its own unique name within the website. The alias path is generated when the document is created and does not implicitly change with the document name. The alias is used for generating the URL of the document (by default), thus any special (forbidden) characters are replaced,
 - You can specify the document alias source field through '*CMS Site Manager-> Development-> Document types-> <doc type>-> Edit-> Fields-> Document alias source field*' (at the bottom of the dialog),
 - **NOTE:** The document alias is shared by all language versions of a document,
- **Custom URL path:** if you want to use a custom URL for a document (different from what is generated based on the document alias) you may use the Document URL path field to assign an alternate URL,
 - **NOTE:** The URL path is culture-specific, so you can specify a different URL for different language versions of the same document,
- **Name path:** consists of the names of all documents on the specific path from the first parent down to the selected document. Unlike the document alias, the name path changes along with the document name. You can force the system to use the name path as the URL path by default,
 - **NOTE:** The name path is culture-specific,
- You can [configure IIS and Kentico](#) to support custom extensions or extension-less URLs and that way define the available extensions for a specific document,

- [Multiple document aliases](#): on top of a custom URL path, you can define multiple document aliases for each document. That way you make a document available through multiple URLs,
 - **NOTE:** Document aliases have lower priority than the custom URL path, thus make sure you remove a URL path that might cause conflicts with the assigned document aliases,
- [Wildcards in URLs](#): you may consider using wildcards in the URL path or document aliases in order to display content based on a specific part of the URL assigned to the document,
 - **NOTE:** Learn more about wildcards in the documentation. There are certain limitations on the use of wildcards if the document has multiple language versions,
- You can exclude certain parts of the website from CMS processing, setup URL prefixes, enable automatic creation of document aliases and manipulate various URL-related settings. [Read more here](#),
- Each instance of a document type allows you to specify its [metadata](#),
 - You can include system fields in document type fields (as described in the previous section) and allow editors to populate metadata without access to the 'Properties' tab,
 - You can also use [macro expressions](#) for metadata fields

Custom document types vs. custom tables

Please refer to section [C. Document types and custom table](#) for a comparison between document types and custom tables to find out whether custom tables might be better type of storage for your structured data.

3.7 Transformations

As mentioned earlier document content is rendered using viewer web parts applying templates we call transformations. But not only document types make use of transformations. The [Custom tables module](#) uses transformations to evaluate the content of particular custom tables as well.

Anyway, no matter whether your structured content is handled by document types or custom tables, [transformations are defined](#) the same way in both scenarios.

- You can choose whether the transformation should use HTML & ASP .NET code or [XSLT](#) instead,

- **NOTE:** You can't use ASP .NET calls in the XSLT transformations—neither pre-defined functions nor custom ones,
- You can generate default RSS, Atom and XML transformations when creating/editing a transformation,
- A set of universal predefined transformation can be found under the RSS or SharePoint document type,
- When creating a new transformation, you may use the '`<name>_<culture code>`' format to specify its name. That way you create a culture-specific transformation. The viewer web part then decides what transformation should be used based on the currently selected website culture,
- You can display a list of functions available in transformations using the link at the bottom of the transformation editing dialog,
 - **NOTE:** The complete list of pre-defined functions is also available in the Kentico CMS API Reference guide. Search for the `CMSAbstractTransformation` class. Its functions are called using the '`<%#[function] %>`' directive,
- ASCX transformations may contain calls to the Kentico API and (ASP) .NET as well,
 - **NOTE:** If you want to call some method (either Kentico API or .NET) in your transformation, you need to use a fully qualified name (including namespace) - e.g. '`<%= CMS.GlobalHelper.ValidationHelper.GetString() %>`',
- You may [create custom functions](#) available in transformations,
 - **NOTE:** Custom functions must be implemented in C#,
- If you want to [use a pre-compiled website](#) you need to deploy all virtual objects (including transformations) to the file system before publishing the pre-compiled website,
- The same applies to [Medium Trust](#) environment deployment,
- Team development may require using the 'Check out to file' button to avoid overwriting changes in transformations,
- When you display multiple document types with a single viewer web part, you may use [macro expressions](#) to specify the name of the transformation to be used, e.g. '`{%classname%}.default`',
- If you need to use macro expressions as the value of some document field, you need to explicitly call the macro resolver in the code of the transformation to evaluate the macro value,
 - The macro resolver is called from transformation code as follows: '`<%# CMS.CMSHelper.CMSContext.CurrentResolver.ResolveMacros(CMS.GlobalHelper.ValidationHelper.GetString(Eval("FieldName"), ""), true, true) %>`',

- Moreover, you can place controls into transformation when using ASCX transformations,
 - [Displaying content rating](#), [document attachments](#), [context menus](#), [abuse report control](#)—these are examples of using additional controls in transformations,
 - Any other control may be included the same way—even your custom ones,
- Search results are also [rendered using a customizable transformation](#),
- Document [attachments may be displayed in transformations](#) using attachment gallery controls,
- For information about advanced content transformations, take a look at the following [webinar](#).

3.8 Customization options

It does not really matter how hard we try to make Kentico the most flexible CMS solution on the market. You may still need to implement custom functionality or adjust existing features to meet your specific needs. Kentico CMS was designed in such a way so as to provide various possibilities [how to handle any customization challenge](#).

Review the list of customization options for various scenarios below. The list contains all basic tasks you may come across when developing a CMS project. For more details on customization and API examples for specific module, please take a look at the [modules section](#) of our developer’s guide.

Task	Customization options
Hiding certain parts of the UI	Use the UI Personalization module to decide what parts of the UI should be hidden/displayed for a particular group of users.
CMS Desk & Site Manager UI customization	You have the full source code of the CMS UI available in the web project folder. You may modify the UI of pages directly or use the upgrade-proof approach .
Execute custom code as part of application/ request/ session/ page/ control life cycle	You can make use of system events fired by different application objects. Read more here .
Extend system object	System objects can be extended by custom data and fields using built-in and native support for easy-to-use customization.

Touch queries before/ after execution	If you need to handle a query before it gets executed or need to manipulate the result set returned, you can easily pre-process or post-process queries .
Modify the layout of out-of-box web parts	You may either modify the ASCX code file related to a web part directly or use a custom web part layout for an upgrade-proof solution.
Modify web part default behavior	The default behavior of web parts may be altered in the following ways .
Work with a custom class without a physical assembly	Follow the steps for customizing providers from the App_Code folder.
Perform additional actions when manipulating with system objects	Use a custom Data event handler to react to changes to system objects. It is useful in case you integrate the CMS with an external application like ERP, CRM, etc.
Handle exceptions your own way	Use the Exception event handler to perform additional actions when exceptions occur within the CMS.
Switch from Forms authentication to other authentication method	You can choose several methods to authenticate users .
Modify authentication and authorization process	Use the Security event handler to integrate external user data storage and modify the default authentication and authorization process.
Execute custom actions when a document is manipulated	You may use a custom TreeNode event handler to perform extra actions, synchronize changes made to documents in external systems, etc. You may handle workflow events through a custom Workflow handler .
Need to use custom database connector, search provider, e-mail engine or e-commerce provider	Develop an alternative provider using Custom providers .
Extend the default WYSIWYG editor toolbar	Follow the steps on how to customize the editor toolbar .
Ship the same information with all documents (no matter the document type)	You can add custom data to all documents from one place using the tree node custom data property.
Customize the registration form	You may create an alternative form to be displayed to users using a customized registration form .

Perform custom actions when a BizForm is submitted	You may run your own custom code when a form is submitted by a user.
Integrate custom/ 3rd party external modules	You can add your own module into Kentico CMS so that it behaves as a native part of the CMS system.
Integrate a custom ASP .NET application	Read more on how to integrate the CMS with an existing ASP .NET application .
Execute custom code in regular intervals	You may use the Scheduler module to schedule custom code .
Include custom functionality/ control on a page	Develop a custom web part to be used on your pages (read more on web part development in the next chapter). The other option is to include your control in the page template directly (no web part created).
Extend an existing page template	To create a custom page template you can modify or clone an existing page template.
Customize default workflow	Simply define custom steps and a scope to alter default workflow configuration.
Support custom media types	Define a process for handling custom media types .
Extend the system object metadata table	You may modify the definitions of system objects related to the metadata table using the System tables module .

Table 3 Overview of customization options for certain parts of the Kentico CMS system.

As you can see in the previous examples of implementing a custom module or web part, one important step is to inherit from the right base class. This represents a general idea that can be applied to all CMS elements. Take a look at the available base classes you are going to use when creating custom CMS elements.

Element	Base class	Description
Filter	<code>CMSAbstractBaseFilterControl</code>	Filters are user controls that are connected to other web parts and allow you to filter out specific data.

Data source	CMSBaseDataSource	Data source web parts retrieve data from the database and expose it for use by other web parts. Find more info about filter development here .
Web part	CMSAbstractWebPart	Custom web parts allow you to use your user controls (displaying content, providing specific functionality) from within CMS Desk.
Widgets	(no base class)	Every widget is based on some web part . Widgets allow live site visitors to define a personalized page template layout.
Form control	FormEngineUserControl	Form controls allow you to create custom field type used in CMS forms running on CMS FormEngine.
Inline control	InlineUserControl	Inline controls are controls placed into text in the form of a macro. When the text is rendered, the macro is replaced by a dynamically loaded control.
CMS Desk UI page	CMSDeskPage	Base class for custom UI pages integrated into the CMS Desk interface.
CMS Site Manager UI page	SiteManagerPage	Base class for custom UI pages integrated into the CMS Site Manager interface.
Master page	CMSMasterPage	To keep a consistent look and feel of your custom UI pages, you should consider creating a custom master page inheriting this base class.
Modal page	CMSModalPage	If you plan to enable modal popup dialogs in your custom UI, use this class for dialog pages.

Table 4 CMS elements and related base classes.

If you are concerned about some specific functionality that you feel requires customization, try to search our on-line [Knowledge Base](#). All articles in the knowledge base reflect real world needs of our customers. It is probable that someone has already come across the same functionality you need, we helped deliver it and shared it on our [DevNet portal](#).

There is also a [Marketplace](#) section on the DevNet portal where you can find custom modules developed and submitted by other clients using Kentico CMS. It is worth reviewing what is out there, as you can save development time if you avoid implementing something that is already available.

NOTE: If you plan to perform complex operations with documents (and any system objects in general) that consist of multiple tasks, you may want to execute all actions in a single transaction. The CMS API provides you with an [easy-to-use transaction mechanism](#) you can take advantage of.

NOTE: You should keep a track of all customized system code files that come as a default part of the CMS project. You will benefit from such a list later during the maintenance stage where you are going to apply a hot fix package or upgrade your website. Needless to say, you will need to overwrite your current versions whenever we update some of the CMS system files. That means that all your customizations would be overwritten, thus you could use a list to identify the customized files and merge those rather than give up on your changes.

NOTE: If you built custom a UI, make sure it complies with Kentico CMS standards. Use the **UI check-list** that can be found within the Kentico Deliver Now! Methodology package.

3.9 Custom web part development

[Web parts](#) are the basic building blocks of your page templates and pages. A web part is nothing more than just an ASCX user control that inherits from a specific base class.

- [Web part development](#) is the same process as creating a standard ASCX control,
- Your custom non-Kentico control can be turned into a web part just by changing its inherited base class,
- If your web part works with date and time in any way, consider using [proper time zones](#) in its code,
- If you need to achieve a consistent look for all web parts on a page, you may consider using pre-defined [web part containers](#) or even [develop custom](#) ones,
- In certain situations you may not need to create a web part from scratch. If you only wish to change the default behavior of an existing web part using different initial property values, [web part inheritance](#) is the best fit for you,

- You can setup [web part properties dynamically](#) in your code,
 - NOTE:** You can also use [macro expressions](#) in the web part configuration dialog to evaluate values for the web part properties. Also, you can take advantage of the macro selector when specifying a macro expression (open it using the black arrow button displayed next to the property field),
- Get yourself familiar with the [common web part properties](#). Due to the inheritance from an abstract base class, you automatically include certain functionality controlled by these properties (displaying the web part only for a certain group of users or on specific document types, enabling AJAX or disabling ViewState for the web part, etc.).

When developing a custom web part, you should also be aware of the relationship between the page and the web part's life cycle.

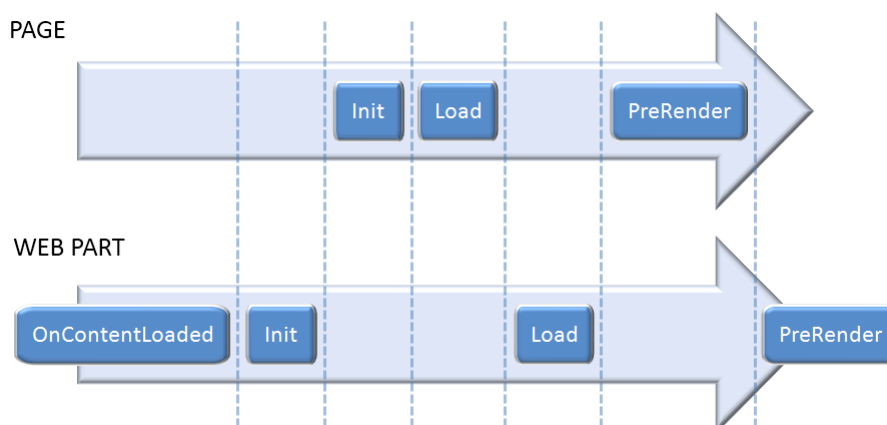


Figure 5 Page and web part life cycle flow.

Events fired during the page/ web part life cycle:

- OnContentLoaded (web part),
 - The event is fired for all controls inheriting from the `CMSAbstractWebPart` base class,
 - The event is fired before the `Init` event,
 - Use this event to initialize the properties of any inner [Kentico CMS Control](#) used in your web part to make sure that the server control has all its properties initialized prior to its `Init` event. Read more in the 'Initializing Kentico CMS controls in your custom web parts' note (at the bottom) of the [web part development](#) section,
- Init (web part),

3. `Init` (page),
 - At this stage all the web parts are loaded and initialized,
4. `Load` (page),
 - All the controls on the page are loaded; `ViewState` and `ControlState` are restored on the `PostBack`,
5. `Load` (web part),
 - As defined by .NET architecture, the `Load` event is fired for all the web parts on the page right after the `Load` of the parent page itself,
 - Register for control events in the `OnLoad` handler, e.g. `Click`, `SelectedIndexChanged`, etc.,
6. `PreRender` (page),
 - This event allows you to make final changes to the content of the page or its controls before the rendering phase,
 - Called first for the parent page and then recursively for all child controls,
7. `PreRender` (web part)

The outlined life cycle matches the default page and ASCX control life cycle defined by ASP .NET architecture. If you are not familiar with the page life cycle read more in the following [MSDN article](#).

NOTE: To make sure a created web part complies with Kentico CMS standards, verify that it is using the **web part check-list** which can be found within the Kentico Deliver Now! Methodology package.

NOTE: To see a hands-on example of how to create a website from scratch following what was mentioned above, take a look at the following [blog post](#).

3.10 Security

Before we start discussing various facts and concerns about security and membership, you may check out the [minimum security requirements for Kentico CMS](#) first. It should give you an idea of what permissions are important for the CMS application to behave correctly. You can overcome some difficult times following the advice given here.

Kentico CMS [security and membership](#) general concerns:

General

- The Kentico security approach is very similar to Windows ACLs,
- The security model consists of [users](#), [roles](#), module permissions, [document type permissions](#), [document-level permissions](#) and extensive [UI personalization](#) features,
- All security related information is stored in the [database in the respective tables](#),
- Roles are website-specific, users are shared between websites,
- Roles are fully-customizable, you may create as many roles as you need,

- Users are members of roles (single user assigned to multiple roles),
- Document-level (per page) permissions can be defined for both users and roles,
- Module permissions are assigned to roles (not to users directly),
- Some modules provide additional security settings (available through the module UI) on top of the global module permissions,
 - **NOTE:** Module-specific security settings are described under particular modules in the [Modules section](#) of the Developer's Guide,
- If a user belongs to multiple roles, the overall permission set is calculated as the sum of all role-specific permissions,
- Permissions for documents are calculated as the sum of permissions assigned to all roles the user belongs to and user-specific permissions for given documents,
- The DENY permission always takes precedence,

Authentication

- You can switch from the default Forms authentication to a different [authentication method](#) according your project needs,
 - Integrate CMS with your [Active Directory](#) (AD),
 - **NOTE:** Enabling [Windows Authentication](#) does not immediately import users from AD. However, a user and his AD groups are imported when a domain authenticated user accesses the website for the first time,
 - Also, you may use the [AD Import Utility](#) included in the Kentico CMS installation to import all users in advance,
 - [Configure mixed mode authentication](#) to allow users to log in using both Forms and AD credentials,
 - If you run multiple applications under a single domain, you may consider enabling [single sign-on authentication](#),
 - To allow users to register and log in using their Facebook, Windows Live, Google or other credentials, you can take advantage of [3rd party authentication service](#) support,
- Use the [Security event handler](#) to integrate external user data storage and the modify default authentication and authorization process,
- You can continuously track on-line users by enabling the [On-line users](#) module,

Registration

- You can easily alter the default registration process by modifying the [available registration web parts](#),
 - Use out-of-box [registration approval and double opt-in capabilities](#),
 - **NOTE:** All users registered on the website can login to all websites running in the same CMS instance by default. Consider locking users only for the sites they registered for by updating [shared accounts configuration](#),
- If you are implementing a community portal, you are most probably going to display user profiles to visitors at some point. To ensure users' confidence, you may allow them to control the [visibility of particular profile fields](#),

Securing the live site and UI

You have several other options on how to increase website security besides document and module permissions,

- You can force the system to require user authentication when accessing certain website areas by [creating secured website areas](#),
 - **NOTE:** By default, only pages within a secured area check user permissions. If you want to check page permissions for public pages, you need to enable 'Check permissions' (for specific pages on the 'Properties-> Security' tab) or to enable the 'Check page permissions' for the whole website (more info at the bottom of the page [here](#)),
- If you deal with sensitive data, you may consider forcing a secured SSL (HTTPS) transfer between the client and web server by [enabling SSL support](#),
 - If you need to enable secure transfers for the CMS UI, go to '*CMS Site Manager-> Settings-> Security*' and enable the 'Use SSL for administration interface' property,
 - **NOTE:** Make sure that SSL is configured on your IIS server and working properly before you switch on the setting. Otherwise, you may receive an error complaining about SSL settings which then prevents you from accessing the CMS UI,
 - **NOTE:** Kentico CMS does not configure SSL for the website, you still need to [setup it up manually on the IIS level](#),
- To deny access to visitors coming from certain IP addresses, you can additionally setup the [Banned IPs module](#),
- To prevent robots from submitting multiple actions on your website, take advantage of the flood protection feature,
 - You enable flood protection at '*CMS Site Manager-> Settings-> Security*' using the 'Enable flood protection' option,

- **NOTE:** You can specify the minimum time interval required between user actions,
- You can [authenticate users using the API](#) whenever necessary,
- If you implement a custom web part, module or other object used in the CMS, you should [check user permissions using the Kentico API](#),
 - **NOTE:** As mentioned earlier, you can create permissions for a custom module to get control over its usage,
- If you need to achieve extra secure configuration that would even check GET and POST requests for any non-standard parameters, you may [explicitly specify accepted parameters](#).

Security threat handling in Kentico CMS

Today's overcrowded internet environment is fertile soil for various groups planning and launching attacks on public facing websites. These groups usually take advantage of various known exploits. To eliminate possible back doors and allow your website to fend off attacks, review the following list of the most common security threats. Along with definitions of the issues, you get our recommendations on how to protect against each of them. In addition, you will see what means the Kentico CMS API provides to allow you to achieve high security of your custom code.

A. Password theft

Issue	If you store and display passwords in the UI in readable form, it is a security threat. You can never say for sure that nobody is watching your screen.
Solution	<p>Protect passwords in the database as well as the presentation layer.</p> <ul style="list-style-type: none"> • Make sure you do not display passwords in readable form anywhere in your custom UI—not even administrators should be able to see the passwords of other users, • Consider using SHA-1 hashing to encrypt passwords before they are stored in the database, <ul style="list-style-type: none"> ○ NOTE: If you switch the password format from default plain text to SHA-1, you need to reset all existing passwords, ○ NOTE: When using SHA-1 hashing, users are not able to retrieve forgotten passwords; you need to generate new ones.

B. Permission theft

Issue	<p>Let's assume there is a user that is only allowed to add new content to the website. Due to a hypothetical security hole, this kind of user could access the section of the website where permissions are managed. If the user can gain higher permissions through this interface, you may have a serious issue. With a new permission set, the user is able to perform unwanted changes to the website configuration or cause some serious information leaks.</p>
Solution	<p>NOTE: All security management areas built into the Kentico CMS UI perform an authentication and authorization check by default.</p> <ul style="list-style-type: none"> • You should always check a user's permissions in custom sections of the UI, especially when you also integrate custom security management areas, • If you define custom permissions through the CMS, you can then use the Kentico API to check whether the user is authorized for specific resources, • Make sure users are not allowed to assign higher permissions than they already have to themselves or to other people, • If you develop your custom UI integrated into the CMS UI, use the correct page base classes (a list of available UI page base classes can be found in Table 4 CMS elements and related base classes).

C. [Cross-site scripting \(XSS\)](#)

Issue	<p>A page displaying an input field makes XSS possible if input is not treated in the proper way.</p> <p>Imagine the following situation. You display an unhandled text box field on the product page. Users enter comments for the products. When a comment is submitted, it is listed on the page. A user enters JavaScript code as a comment and submits the form. The code gets rendered and executed. The attack was successful.</p> <p>The worst part is that an attacker may cause you many sleepless nights by using very simple JavaScript code.</p>
--------------	---

<p>Solution</p>	<p>Always expect that users will enter harmful code. You must handle content entered via input fields and escape potentially harmful code.</p> <ul style="list-style-type: none"> • Encode content before it is rendered on pages, • Encode all strings coming from an external source (user input, database, context/environment, URL query string parameters, method parameters, etc.), • For strings from external sources, use the <code>HTMLHelper.Encode()</code> method, • For URL parameters, use <code>QueryHelper.GetText()</code>, • Values coming from an external source rendered as part of JavaScript code must be encoded with <code>ScriptHelper.GetString()</code>.
------------------------	--

D. [Cross-site request forgery \(XSRF\)](#)

<p>Issue</p>	<p>XSRF is the ability to perform certain actions on the website just by redirecting a user to a page using specific parameters in the target URL. XSRF takes advantage of authentication information stored in cookies. When such a URL is submitted by an authenticated user, a certain action is executed without the user's approval.</p>
<p>Solution</p>	<ul style="list-style-type: none"> • Do not perform any actions on GET requests, always use POST, • Never turn ViewState validation off on a page, • Encode ViewState using machine key, • Do not set the <code>CMSUseViewStateUserKey</code> key to <code>false</code>. It enables ViewState encryption using a user specific validation key (<code>SessionID</code>), • When you create a custom UI page, always inherit from the base page classes listed above (Table 4 CMS elements and related base classes.) • If you create a new page base class for custom UI pages, make sure it inherits from <code>CMSAbstractPage</code> (directly or indirectly), • You may also consider securing URLs using a hash code.

E. [SQL Injection](#)

Issue	<p>Similar to XSS attacks, SQL injection also takes advantage of unhandled input fields. When you use the value of some input field in the text of a SQL query, you give users the possibility to change the text of the original query and execute possibly harmful code. It allows users to execute any database command and gain access to the system, change or delete the data and so on.</p>
Solution	<ul style="list-style-type: none"> • Use SQL parameters for dynamic parts of the <code>INSERT</code>, <code>UPDATE</code> or <code>DELETE</code> queries, • Avoid using the <code>exec()</code> function in SQL code, • When you build <code>SELECT</code> queries in the code, replace single quotes with double quotes—use <code>.Replace("'",'"')</code> for every input coming from an external source, • Do not rely on JavaScript validation. JavaScript is executed on the client side. An attacker can disable such validation easily, • Make sure the input used in SQL queries represents a valid value of the expected data type. For example, use the <code>ValidationHelper.GetInteger()</code> method to validate incoming input if a query expects an integer value.

The list given above obviously does not include all vulnerabilities known today. We therefore recommend that you check community groups for the latest information on revealed exploits and how to leverage your code security accordingly.

3.11 Website optimization

The last step in the development process should be performance optimization. Several people worked on the development, which may result in varying quality of the code and built solution. To ensure performance is not hurt by inefficient code or overlooked settings, you should devote some additional time to reviewing the available performance indicators and verifying website configuration.

A. Built-in debugging tools

A kind of best practice would definitely be to use the [CMS debug capabilities](#) to review how the website stands and performs from an efficiency and optimization point of view. Before you get your hands on professional optimization heavy artillery, give the [out-of-box debugging functionality](#) of the CMS a shot.

You can use debug to review loaded [System objects](#) and [Cache items](#), [Cache access](#), [SQL query](#) efficiency, [ViewState](#) usage, [Output](#) validity, [Security](#) evaluation, [Request](#) processing, [Macro](#) evaluation, [Worker thread](#) performance or [Web farm task](#) synchronization. Information displayed on the debug tab is very useful, not only for identifying the cause of potential issues, but extremely handy when you are optimizing a website.

- You can enable all debug tabs at once using [bulk keys](#),
- If you want to get notified when a system error occurs, setup [website error notifications](#),

System objects

- All system objects listed on the System objects tab are loaded into hash tables for higher performance,
- Whenever you make a change to a system object in a web farm environment, the [Web farm support](#) should take care of reloading objects in system hash tables for you. Otherwise you may encounter problems with outdated data displayed to other developers in a team development environment,

Worker threads

- Check the duration of active threads,
- Any operation running suspiciously long may require your attention,
- Use context information to locate the source of the call causing an overloaded thread,

SQL queries

- Check the queries log for query duplicity—if the same query is executed multiple times from the same context, it is most probably an optimization issue,
- Keep an eye on data returned by queries
 - Check the number of columns returned by a query—make sure the query returns only necessary columns,

- **NOTE:** Most of the web parts feature the 'Columns' property that you can use to specify what columns should be returned,
- **NOTE:** If you are using the Kentico API to execute queries, always use the override accepting the columns parameter,
- Also check the duration of query execution, if you find a query running longer than others, verify its efficiency,
- Make sure you use the proper indexes for your data tables. We are not able to create default indexes to suit every project and every scenario (double check for custom tables),
- If you are still not satisfied with the performance of your web site, run SQL Management Studio Profiler or Data Tuning Advisor to investigate the efficiency of your query execution plan, index usage statistics, etc.
 - Watch out for the [index seek versus index scan](#) ratio,
 - Eliminate as many ad-hoc execution plans as possible,
 - If you are running SQL 2008 or higher, use forced parameterization for queries (parameterization is enforced only for queries without ORDER BY 1 statements),
 - Get yourself familiar with [AWE](#) and [PAE](#) settings to decide whether they can help your scenario,
 - Consider modifying [MaxDOP](#) used by the SQL server,
 - **NOTE:** It is recommended to set MaxDOP to match the number of physical processors available on the SQL server machine,
- You can also setup a multi-server database environment to handle the enormous load your website may possibly get,
 - More information about supported architectures is given in section [5.1 Production environment configuration](#),

ViewState

- ViewState serializes and renders additional content on pages causing the page size to grow accordingly,
 - The advice here is pretty simple: disable ViewState for all controls that do not need it to work properly (typically labels, textboxes, literals, drop-down lists, etc.),
 - Especially try to eliminate ViewState items with the 'Is dirty' flag set to 'Yes' (in red),

Output

- Check the size of the page output (HTML code). If it gets too big (above 80kB), review the code for any parts that could be optimized in some way,

Security

- You should check the security debug tab for any duplicity in security operations,

Request

- On this tab, you may easily check what specific part of the requests caused a delay in response time,
- Review any part of the request processing that took significantly longer than other parts.

NOTE: All debug records are stored within the database. This means that enabling debug facilities puts more pressure on database and CPU resources. The higher the load you get with debugging enabled, the greater the overhead your server will face.

- Turn off debugging as soon as you finish optimizing/ troubleshooting your website.

B. Caching options

Caching is the most powerful weapon when it comes to website performance. You should definitely understand how caching works in Kentico CMS in order to be aware of all the [different caching possibilities](#) offered by the CMS (read even more in the following [blog post](#)).

- If you are developing a custom control that manipulates data in any way, you should consider implementing caching for it,
 - Kentico CMS provides [API to help you maintain cache](#) for custom controls,
 - You can get more examples of API usage in [this webinar](#),
- When you cache content that depends on some other system object and want such content to be updated on the client side correspondingly, take advantage of [cache dependencies](#),

- If you notice issues with the application using too much memory, it could be related to the way [caching is setup for web parts and controls](#),
- If you encounter any performance degradation in the production environment under a heavy load, make sure caching is not overused,
 - If you do not have enough memory to cache all content, you may end up with a situation where items are removed and added from/to the cache too often. In such a case, you should consider turning off caching for some parts of the system or better yet expanding the available memory level,

C. Output filter and other settings influencing performance

Output filters perform additional changes to the HTML output before it is sent to the client. There are several [types of output filters](#) you may use. As you can guess, such HTML output manipulation uses resources and increases the overhead of request processing.

An output filter (depending on its purpose) basically searches the output code and tries to fix any problems. In an ideal scenario, we should not need any output filters as long as our code is squeaky-clean. Unfortunately, that is not always the case and that is why you should [understand each of available output filters](#). Then decide whether you can afford to turn off a certain filter or not.

- You can extend the collection of pre-defined output filters and [create a custom one](#),
- Another group of functionality that influences website performance to a certain extent is related to images and files. You may boost website performance by [speeding up the retrieval of your files and images](#),

D. Event log

Whether you realize it or not, the event log (*CMS Site Manager-> Administration*) may cause performance issues under certain circumstances.

Event log data is stored in the database. The length of the log history is defined by the '*CMS Site Manager-> Settings-> System-> Log size*' setting. You can partially control what events are recorded in the log using the '*CMS Site Manager-> Settings-> System-> Log metadata changes*' option.

The website slow down may be caused by the event log being under a heavy load when an error on the website is thrown too often (e.g. references to a missing image). In such a case, new records of the exception are inserted over and over again and once the log meets its max length, it starts removing old records. As you can see, there are many I/O operations invoked by the CMS that the database may not be able to handle.

- Always set a reasonable history length for the event log,
- Try to resolve all errors you get in the event log.

NOTE: If you followed all recommendations and are still facing performance issues, try the advice on [troubleshooting performance issues](#).

E. SEO optimization

There is no doubt [SEO](#) is an important part of the development and optimization process. The same SEO rules apply to your Kentico CMS websites as for any other websites. Unlike ad-hoc websites, Kentico CMS provides easy-to-manage SEO support that you can benefit from.

There are a [few basic rules](#) related to SEO that you should definitely comply with. If you are interested in SEO topics, you can download [Google SEO Starter Guide](#) to help you understand what it is all about. You will eventually see that SEO is not as complex as one might think.

Kentico CMS manages SEO in several parts of the system:

- You should be concerned about [canonical link elements](#) on your website. Even though the CMS does not support this feature out-of-box, you can [easily achieve this functionality](#) by using built-in functionality,
- You can [specify page metadata](#) (title, description, keywords) through the 'CMS Desk-> Content-> <selected page>-> Edit-> Properties-> Metadata' tab,
- You can use external tools to check how your website follows SEO standards,
 - You can even use SEO analysis applications available for free. [Web CEO](#) is a free suite combining several SEO tools that help you uncover weak spots in website search engine readiness,
- Let the CMS system handle the [robots.txt](#) file for your website. You can also easily [handle robots.txt files when running multiple websites](#) under a single Kentico CMS instance,
- You should include keywords in links wherever possible,

4 Testing

The testing phase is crucial for ensuring the overall quality of deliverables. Besides validating specific functionality, you should go through some general quality assurance process of functional testing, site validation and load testing to make sure you deliver a stable, safe and defect-free solution to your client.

4.1 Functional testing

At this stage you should make sure any document type fields, BizForm fields, [custom form controls](#), custom controls, web parts and basically all input fields across the website apply validation rules. Also verify the functionality of custom developed features and modules.

- For any input field defined through the CMS UI (through the field editor) that requires validation, you should specify validation parameters in the 'Validation' section of the field detail form,
 - Do not forget to set max length for input fields to avoid exceptions thrown by the SQL server due to data length exceeding column specifications,
 - **NOTE:** For special types of fields (e.g. e-mail, phone, ZIP, etc.), it is not always necessary to define a custom regular expression. You can use some of the pre-defined field types designed to provide the desired validation functionality out-of-box,
 - **NOTE:** To implement validation in your custom controls or web parts, you may take advantage of the `CMS.GlobalHelper.Validator` class which features pre-defined methods providing different kinds of validation (e.g. code name format validation, e-mail format, various data type validation and so on),
 - You need to instantiate the `Validator` class to get access to all its non-static classes,
- Sooner or later you are going to develop/ customize certain part of the CMS system. Any enhancement done to the CMS should be bundled with an appropriate monitoring mechanism. You should use the `CMS.EventLog.EventLogProvider` class to log any events system administrators should be aware of,
 - **NOTE:** Logging exceptions through the `EventLogProvider` ensures that notification e-mails are sent when an error occurs. You therefore need to specify the '*CMS Site manager-> Settings-> System-> Error notification e-mail address*' setting,
- Make sure any custom UI sections comply with Kentico CMS standards. Use the **UI check-list** which is part of the Kentico Deliver Now! Methodology package,

- Review ‘*CMS Site manager-> Administration-> Event log*’ for any exceptions that occurred during functional testing performed recently. You should not see any of those before moving live,
- Perform functional testing to verify use case scenarios.

4.2 Site validation

Time devoted to site validation can eventually be turned into a nice website performance boost and tightened security.

- Make sure the (X)HTML output produced by the website complies with W3C standards allowing you to turn off various output filter options as mentioned in [C Output filter and other settings influencing performance](#),
- Make sure that the website is attack-proof against various types of security threats as mentioned in section [3.10 Security](#) above,
- Check whether the website output matches all the requirements of various web standards as mentioned in section [D Which web standards should be followed in terms of accessibility and coding?](#),
- Make sure permissions are applied and validated correctly if you configure advanced security for the CMS users (e.g. for limited access to the modules UI, secured website areas, etc.),
 - **NOTE:** Should you encounter any difficulties identifying the causes of permission related issues, take advantage of the built-in [Security debug](#) capabilities.

4.3 Load testing

In chapter [3.11 Website optimization](#), we talked about optimizing the website by maintaining the database indexes and statistics. Part of the testing stage should therefore also be validating the results of such optimization with your own load test preceding the deployment to the live server.

With every major version, Kentico releases a performance report (the [latest report](#) was published for Kentico CMS 5.5), which provides you with the results of load tests performed for the specific version of Kentico CMS using one of our default sample websites.

Besides advanced load testing applications, you can use [Visual Studio Enterprise Architect – Application Center Test](#) (ACT) to perform a simple load test on your website to verify that any changes and optimizations you have done serve a purpose. This tool is part of MS Visual Studio Enterprise Edition.

5 Deployment

5.1 Production environment configuration

Before you go live, it is important to consider production environment setup. You can take advantage of different network architectures to support websites ranging from small to enterprise level. A properly chosen architecture allows a website to perform well and ensures it serves its purpose.

A. Web farm support

A web farm is a collection of computer servers intended to accommodate server needs far beyond the capability of a single machine. It is configured on the IIS level and basically allows an increased ability to serve incoming requests.

- If you expect heavy traffic to hit your website, configure a web farm environment on the IIS level and enable [web farm support](#) in Kentico CMS,
- The number of web farm servers (WS) supported by a single Kentico instance depends on the type of purchased license—contact your Account Manager for details about your license if you are not sure,
- Web farm support ensures the synchronization of all changes in the site settings, cache or file system between all servers in the farm.

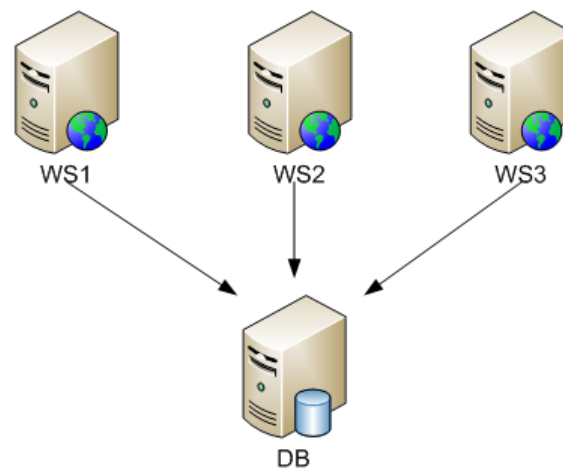


Figure 6 Basic web farm setup

B. Web farm support and a clustered database

Another option is to run the website in a web farm environment and enhance the database level configuration. The database does not need to be just a single physical machine. Utilizing multiple machines and [running the SQL Server instances in a cluster](#) could multiply database performance. The higher the load the database can handle, the faster the web site reacts to the users' requests.

- A database cluster acts as a single SQL Server instance for the website. It means web farm support can eventually be applied the same way as in the previous example.

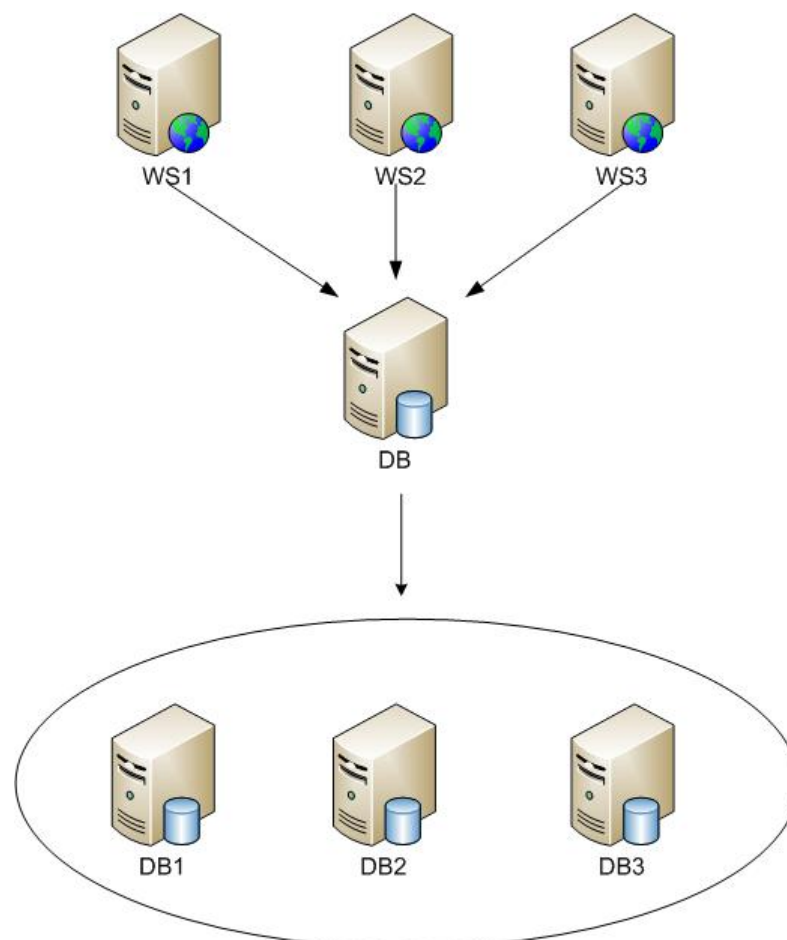


Figure 7 Web farm running on a clustered database

C. Web farm support and SQL replication

The most advanced scenarios may incorporate replication of the data changes on a database level using native support provided by the SQL Server. The website may be a part of a content delivery network (CDN)—a network where web servers and database servers containing copies of the data are located at various points. The purpose is to maximize bandwidth when accessing data. The data are always accessed from the server nearest to the client.

- Each web server makes use of a single database server,
- All database servers establish a [replication relationship](#) and propagate changes to each other,
- Web farm support is [enabled](#) the standard way.

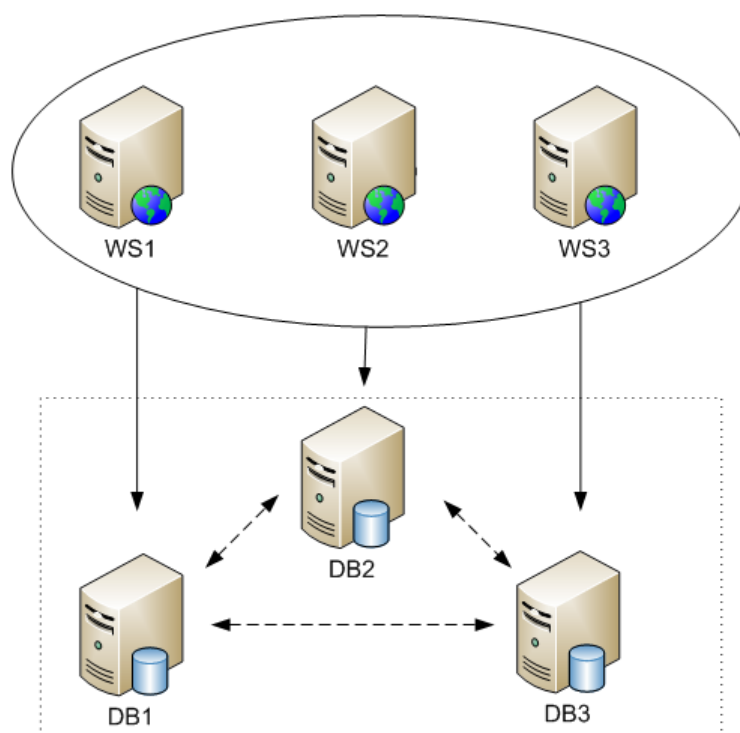


Figure 8 Web farm support and replication

NOTE: If you are interested in more details on Kentico CMS setup for load-balanced, highly available redundant web farms, watch this [webinar](#).

5.2 Deployment options

Before you start deployment actions, you should verify that the production server meets the [minimum system requirements](#). Hardware-wise, it is recommended to have at least 1 GB RAM and a Pentium 4 or Pentium Core 2 Duo (or similar) processor. Obviously, the more memory and computing power you supply, the higher the load the website can handle. Although, sometimes it is better to spend additional time optimizing the website rather than spending money on more powerful machines.

A. Deployment in general

Concerning deployment, you have basically [two options how to deploy a website](#) to the production server:

- Using the [Export/ Import](#) functionality of the CMS,
- Restoring the website using a backup project folder backup and DB,

Export/ Import

- Export/ Import allows you to [export a website](#) along with all site-specific data and global objects and [import](#) it into the production instance of the CMS,
- You need to install a production instance before deployment,
 - You can use the web installer to [install Kentico CMS on a remote server](#),
- If you implemented any custom functionality, you should be familiar with how the [project folder structure](#) influences what physical files are included in the export package,
 - If you want to exclude specific files or folders from the export, you may [use web.config keys](#),
- There is no limit on size of the export package—it may contain files of any size,
- You need to gain access to the website project folder to be able to copy the export package to the target website,
 - **NOTE:** If you use the publish function of Visual Studio to deploy a pre-compiled website, you will not be able to use export/ import—the website is compiled into an assembly, so the project does not keep the standard folder structure required by export/ import,

Restore from backup

- The backup-restore option allows you to restore a production environment identical to what you have in your local environment,
- There is no limit of the maximum size of objects related to the website,

- You simply copy the project folder to the production environment and setup the production IIS to point to this new location. Then you restore the DB backup file and update the web.config file to point to the new database,
 - **NOTE:** You need to gain access to the SQL server to be able to restore a DB backup file,
 - **NOTE:** Ideally the development and production SQL servers should use the same settings,
- Because you restore the DB from a backup file, all metadata including website settings are configured for the development environment. Thus, you need to make sure that environment specific settings (in '*CMS Site Manager > Settings*') reflect the new location (site domain and domain alias settings, SMTP server details, etc.),
- You cannot use this option for incremental deployment as there is no easy way to select individual objects to be deployed,

Incremental deployment

- You can use export/ import to perform incremental deployment of new functionality into a website already running in the production environment,
 - Incremental deployment is possible using the [export objects](#) or [single object export](#) functionality,
 - The other option is to setup [Content staging](#) between the development/ staging and production environment,
 - You can synchronize individual objects specifically—you can make a selection on the object level,
 - You cannot synchronize all system objects using staging functionality (e.g. web part code files, BizForm data, forum posts, ACLs [document permissions], etc.). A complete list of supported objects can be found on the [module's overview page](#),
 - Content staging uses HTTP POST requests to transmit data between the staging and production server. Due to this fact, the maximum size of any synchronized file is dependent on the maximum size specified for a request.

B. Deployment to Medium trust environment

If you are going for shared hosting as a production environment, you will most probably end up with a medium trust security policy. In such a scenario be aware of several limitations that require additional configuration of your CMS website.

- If you are planning to deploy a website using export/ import functionality, you will need to [install a Kentico CMS instance on the shared hosting](#) in a slightly different way than in a full-trust environment,
 - Actually, the only difference is that you need to update the web.config file to use a managed directory provider,
- As medium trust policy is about restricting application access to resources that can cause issues for other websites in shared environment, you need to perform [additional configuration tasks](#) for the website.

C. Pre-compiled website

[Website pre-compilation](#) allows you to publish the website in the form of an assembly (instead of hundreds of project files) and that way make deployment easier and the website faster. However, there are drawbacks to this approach in the form of several limitations related to the way how pre-compiled websites handle virtual objects (e.g. transformations, templates, etc.) that do not exist physically.

- Like when using a medium trust environment, a pre-compiled website requires you to perform [additional configuration](#) before deployment.

D. Nested websites

Sometimes you may need to [setup nested websites](#) in a production environment. Nested websites allow you to run another non-Kentico web application under the same domain as the CMS website and that way create the consistent look and feel of a single website (even though there are two different web applications running on a separate code base and database).

5.3 Deployment actions

Once you have a website deployed to a production environment, you should check the integrity of the website by reviewing the functionality of the live site, especially:

- Check the website for broken links,
 - Links may get broken after deployment to a new environment if you have not used relative paths (using '~' in the URL) for links,
 - **NOTE:** If you use CMS selection dialogs for inserting references to files or documents, the system automatically uses relative links. You should therefore primarily focus on custom code and links you have hardcoded into website content,
- Perform load tests on the production environment,
 - Check the CMS event log for any errors occurring in the production environment,
 - You may temporarily enable the CMS debugging feature to check for bottle necks in the data access layer,
- Verify the SEO accessibility of your production website.

NOTE: You can use the [Link Checker and Validation Module](#) available on the Kentico Marketplace to validate (X)HTML markup, broken links, accessibility and mobile device friendliness.

5.4 Post-deployment actions

To make sure a website in production is continuously performing well, you should regularly carry out several routine maintenance tasks:

- First of all you should keep an eye on the CMS event log. Any error in the event log indicates a possible issue that may be the result of a content update, new deployed functionality, broken service, content staging malfunction, etc.,
- Kentico delivers a 7-day bug fixing policy, releasing a new hot fix package every week. The package provides fixes for all bugs and issues reported in the past 7 days. You should therefore always apply the [latest hot fix package](#) available to make sure you keep your CMS instance in the best shape,
 - The hot fix package usually contains updated files for you to copy over to your website project folder and SQL scripts to execute against the website DB. The package always includes a PDF document with detailed information on how to apply the specific hot fix—read the instructions carefully,

6 Website evaluation

When you have the project live and running, it is time to stop for a while and evaluate your efforts. Answering a few questions should help you get the big picture of how the project evolved, what issues you encountered, how the customer feels about the results, etc. It should help you to answer the most important question of all: was my project a success and what can I learn from it?

- You should establish whether all the goals of the website were met,
- Ask your client if they are happy with the website, which part of website they like or dislike, if there is anything they would change after they got a chance to see the website live, etc.,
- Determine if the website response time is reasonable based on the used hardware, hosting, etc.,
- Summarize the issues you encountered during project development,
 - Based on our experience gained during [consulting sessions](#), it is a huge help for upcoming projects if you compile a list of issues, misunderstandings or harsh times (if any) that you went through during the development,
 - You can use such a list for future reference to avoid the same issues.
- Evaluate the SEO accessibility of your website,
- It might be useful to collect feedback on the functionality from website visitors to get the opinion of real users. You can prepare a simple questionnaire to be handed off to visitors or even create a custom feedback BizForm displayed on the website.

Appendix A – Requirements Template

	Note
A. What is the goal of the project?	
B. What is the expected number of users/ page views during the peak load, what is the expected number of documents on the website?	
Is it going to be static or dynamic content?	
How many documents do they plan to include within the website (hundreds, thousands, etc.)?	
Does your client plan to create multiple websites sharing the same (re-usable) content?	
C. What is the structure of the website and what types of content will be published?	
Does your customer want to include some kind of document repository (for documents, video and audio files, images, etc.)?	
What are the main sections the customer wants to divide the website in?	
What items should the navigation contain?	
Is there any requirement for implementing membership areas or restricted sections on the website?	
Do they plan to display personalized content (changing the layout, styles and look of the page based on the current user)?	
D. Which web standards should be followed in terms of accessibility and coding?	
E. Which products and technologies will be used?	
F. What is the content life-cycle? Who is responsible for the content management?	
What type of users would be responsible for creating, updating and deleting documents?	
Should website visitors get a chance to contribute to the website?	

Appendix B – Design Template Processing

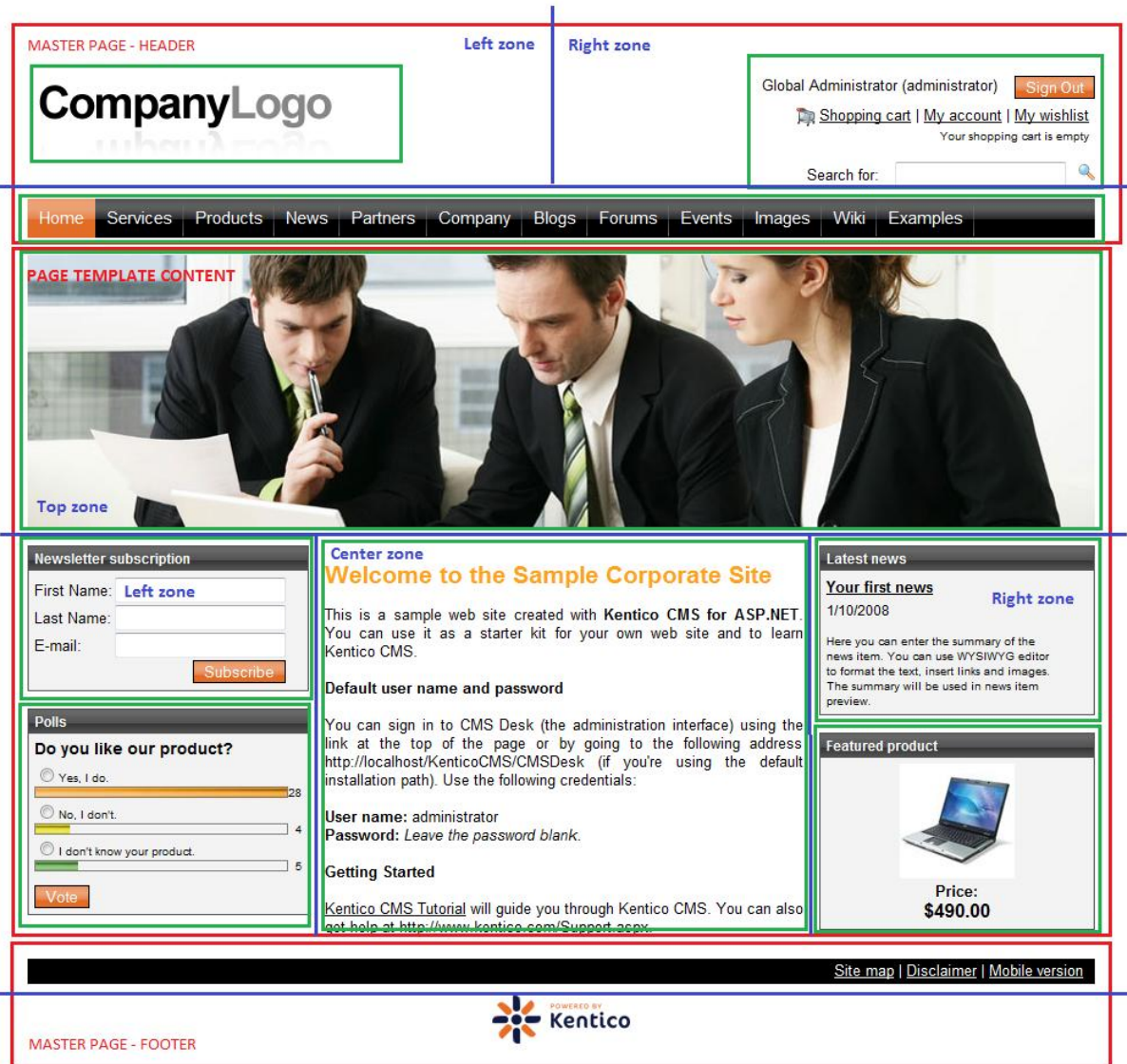


Figure 9 The red box at the top and bottom represents the master page template content while the red box in the center is the content of a page template. Blue lines split each part into zones. Furthermore, each zone is divided into several smaller parts—web parts

Appendix C – Website Wireframe Example

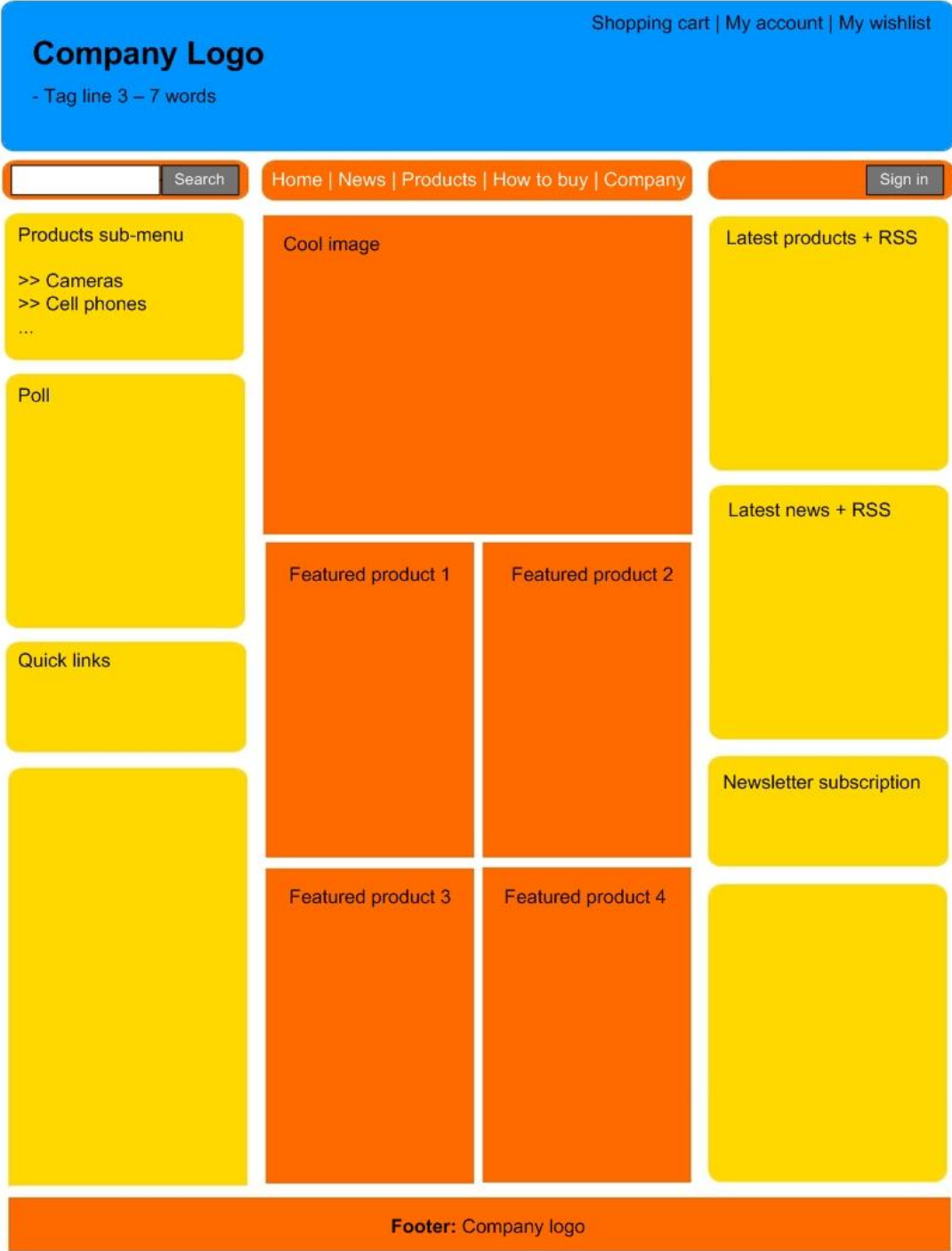


Figure 10 Wireframe example