

Kentico CMS 6.0 Tutorial ASPX

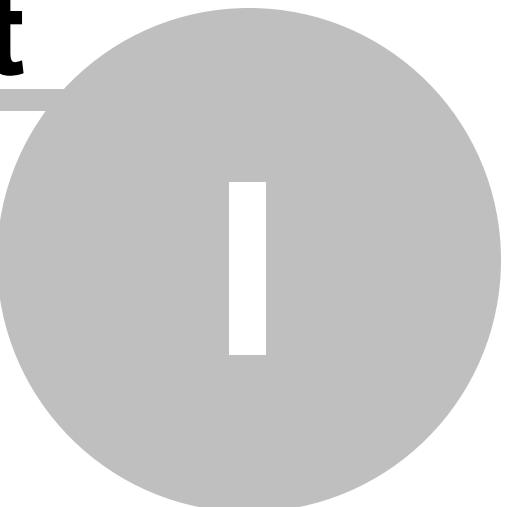


Table of Contents

Introduction	5
Kentico CMS Overview.....	5
Installation	7
Prerequisites.....	7
Setup installation.....	8
Web application installation.....	8
Database setup and Corporate Site.....	12
Managing content	19
User interface overview.....	19
Editing home page content.....	21
Creating a new page.....	23
Inserting an image.....	27
Creating a link.....	28
Creating a news item.....	31
Site Development Overview	35
Site Development Overview.....	35
Creating pages using ASPX templates	40
ASPX page templates.....	40
Creating a simple ASPX page template.....	42
Using master pages.....	48
Adding portal engine functionality to ASPX templates.....	50
Managing styles and design	58
CSS styles.....	58
App themes.....	60
Menu design.....	61
Creating a new site using ASPX templates	64
Overview.....	64
Creating a new web site using the wizard.....	64
Creating the CSS stylesheet.....	69
Opening and configuring the web project.....	71
Master page.....	72

Main menu.....	76
Home page.....	77
News page.....	85
Services page.....	89
Products page.....	93
Overview	93
New document type	93
Transformations	99
Page template	101
Search page.....	105
Secured section for partners.....	108
Further steps	114
Further steps.....	114

Part



|

Introduction

1 Introduction

1.1 Kentico CMS Overview

Kentico CMS for ASP.NET helps you create powerful dynamic websites with minimum effort. This document will guide you through the most important features of the system step-by-step, so that you can start creating your own websites.

This document was written for evaluators and new users. It's intended for developers who create the websites. It's not intended for end-users without programming knowledge.

If you need a more detailed documentation of some features, please see one of the following documents:

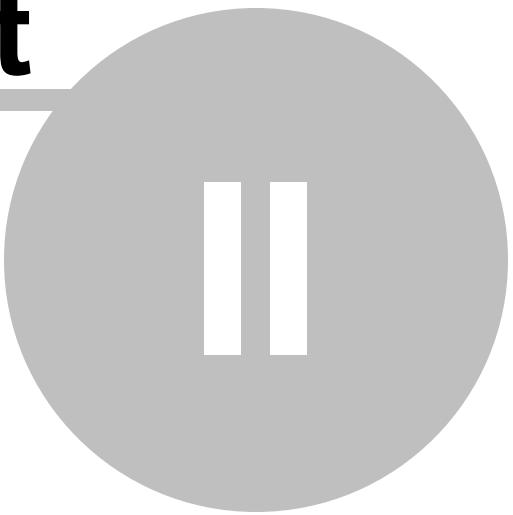
- Developer's Guide
- Controls Reference
- Web parts or Widgets Reference
- API Reference
- Database Reference



Kentico CMS Support

You get free technical support during your evaluation period. If you need any help, please visit <http://www.kentico.com/Support.aspx>.

Part



Installation

2 Installation

2.1 Prerequisites

Before you start the installation, please make sure you have the following software installed. Other configurations may work too, but the system was not tested on them.

Server-side Requirements

- Windows XP, 2003, 2008, 2008 R2, Windows Vista Home Premium/Business/Enterprise/Ultimate or Windows 7 (both 32bit and 64bit)
- Microsoft .NET Framework 3.5 SP1 or higher
- Microsoft Internet Information Services (IIS) or Visual Studio/Visual Web Developer 2005/2008 built-in web server
- Microsoft SQL Server 2005, 2008 (including free SQL Server Express Edition)

Hosting Requirements

- ASP.NET 3.5 SP1 (or higher) and Microsoft SQL Server 2005/2008 support
- Medium-trust or full-trust permissions for the ASP.NET application
- If the server uses medium trust, ASP.NET AJAX 1.0 must be installed on the server.
- If the application uses .NET Framework 3.5 SP1 and is hosted in a medium trust environment, it is necessary to have [Microsoft Chart Controls](#) installed on the server.
- It's recommended that your hosting plan comes with 125 MB or more memory and 100+ MB database.

You can use your favorite hosting provider or choose from our [hosting partners](#).

Development Tools

If you want to create custom web parts or integrate custom code, you need **Visual Studio 2005/2008/2010** or **Visual Web Developer 2005/2008/2010 Express Edition**.

Supported Client Browsers for Content Editors

- Internet Explorer 7, 8, 9
- Firefox 3.6, 4.0, 5.0
- Chrome 12
- Safari 4.0, 5.0, 5.1

Supported Client Browsers for Site Visitors

- Internet Explorer 6.0+
- Firefox 3.6+
- Chrome 12+
- Safari 4.0+
- Opera 10.50+

(the visitor browser requirements depend on functionality used on the website)

Required experience

Although Kentico CMS allows you to create dynamic websites without programming, you may need to create custom web parts or add custom code when developing a more complex website. You should be able to create a simple application in ASP.NET 3.5 using Visual Studio 2005 and have some experience with relational databases and SQL, so that you can leverage the flexibility of Kentico CMS.

2.2 Setup installation



Troubleshooting installation issues

If you encounter any problems during the installation, please see **Kentico CMS Developer's Guide -> Installation and deployment -> Troubleshooting installation issues** or contact our support at <http://www.kentico.com/Support.aspx>

Run KenticoCMS.exe and follow the installation wizard:

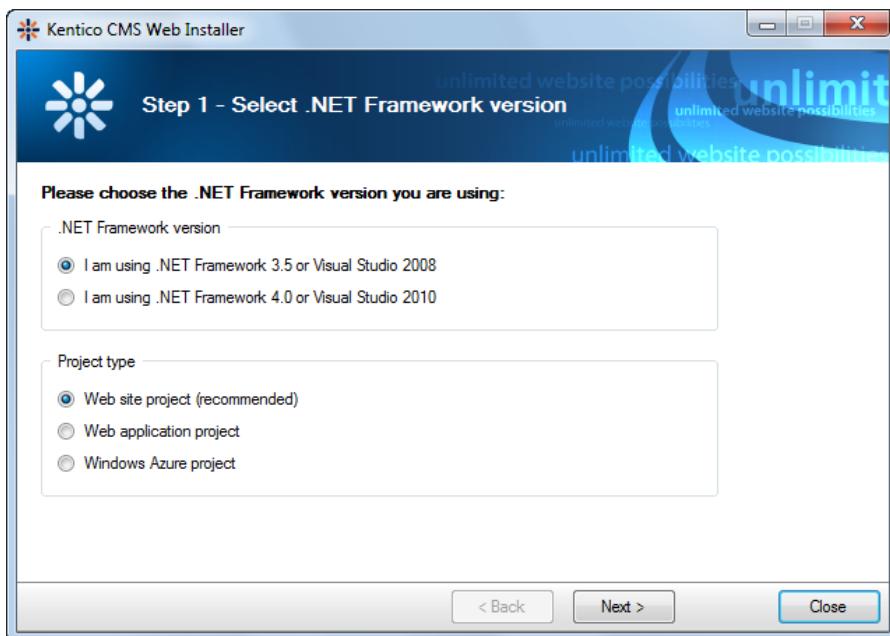


Read and accept the license agreement and click **Next**. Choose the installation location of the binary files and documentation on your disk. After the setup completes the installation, check **Launch Kentico Web Installer** and click **Finish**.

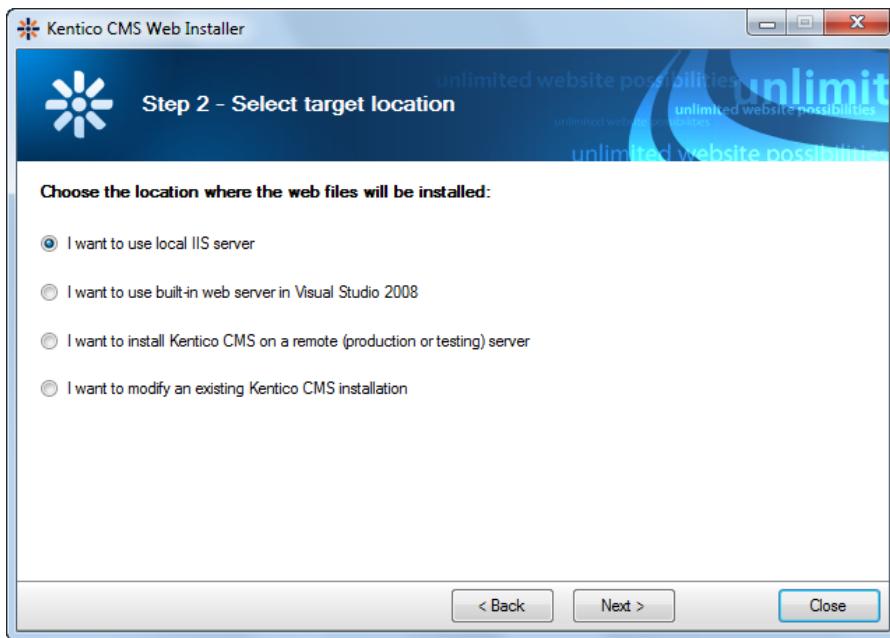
2.3 Web application installation

Now you should see the Kentico Web Installer. If it has not started automatically, you can always run it from **Start menu -> All Programs -> Kentico CMS -> Kentico Web Installer**.

First, you need to choose the version of .NET Framework and Visual Studio and the project type that you wish to use. Click **Next**.



Now choose to use either IIS server or the Visual Studio built-in web server (if you do not have IIS installed). Click **Next**.

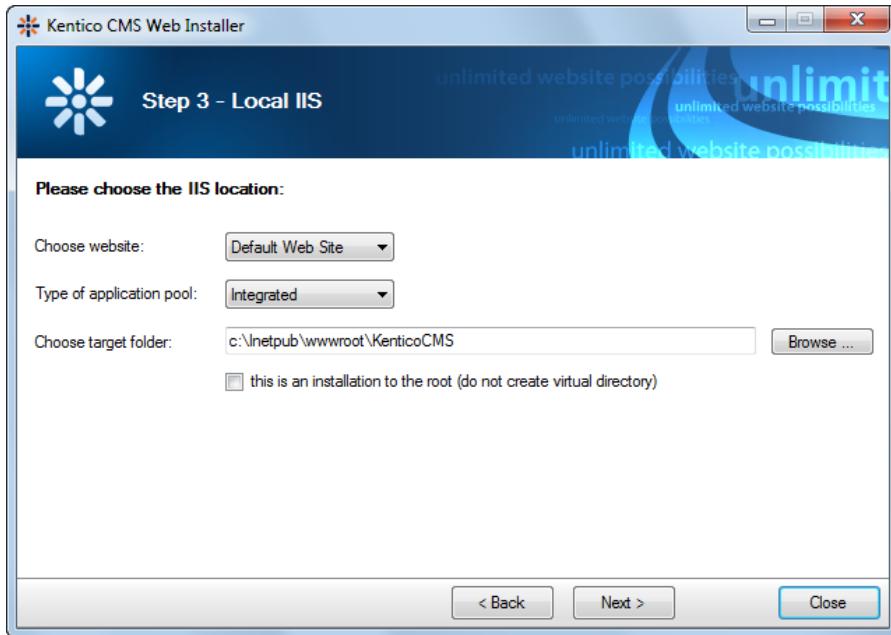


IIS installation

If you choose the IIS server, you can choose the website where the virtual directory will be created and the folder on your local disk where project files will be deployed. The installer will create a new virtual directory on your server and configure it for ASP.NET. Click **Next**.

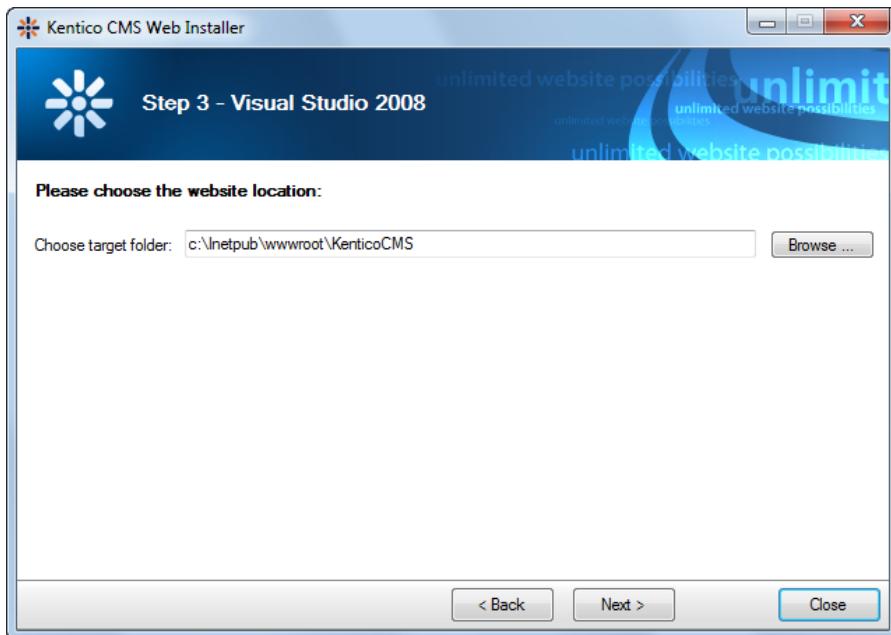
Please note: if you are installing Kentico CMS into the root of your website (such as <http://www.domain.com>) and do not wish to create a virtual directory (such as <http://www.domain.com/cms>), please

check the **This is an installation to the root (do not create virtual directory)** check-box.



Visual Studio installation

If you chose to use the built-in server in Visual Studio, you only need to specify the local disk where the project files will be deployed. Click **Finish**.

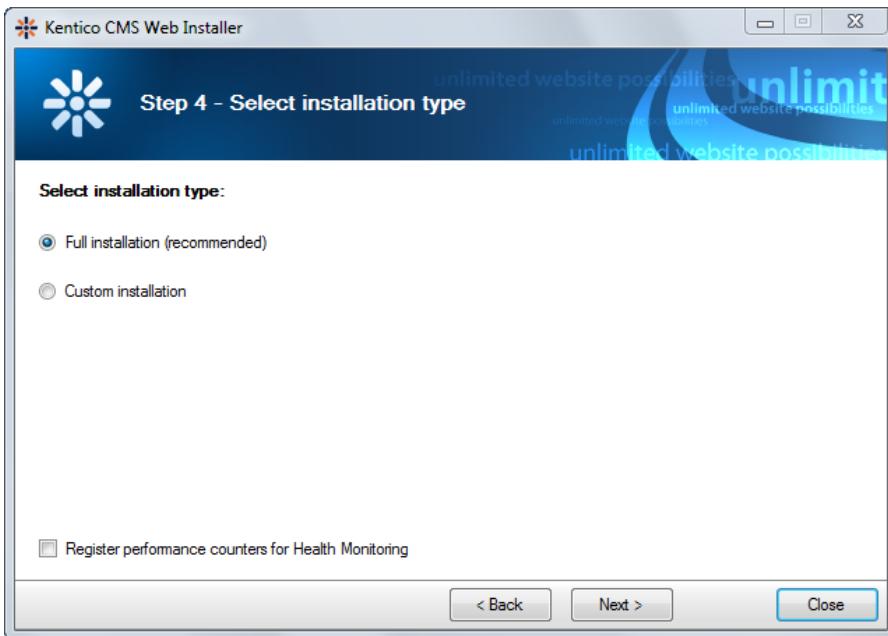


Type of installation

No matter if you chose the IIS or VS installation, the next step after specifying the target folder is the selection of the installation type. The following two types of installation are available:

- **Full installation** - this is the recommended option for the purposes of this tutorial; in this type of installation, all components of the CMS will be installed.
- **Custom installation** - in this type of installation, one extra step will be displayed, letting you choose which components you want to include in the installation.

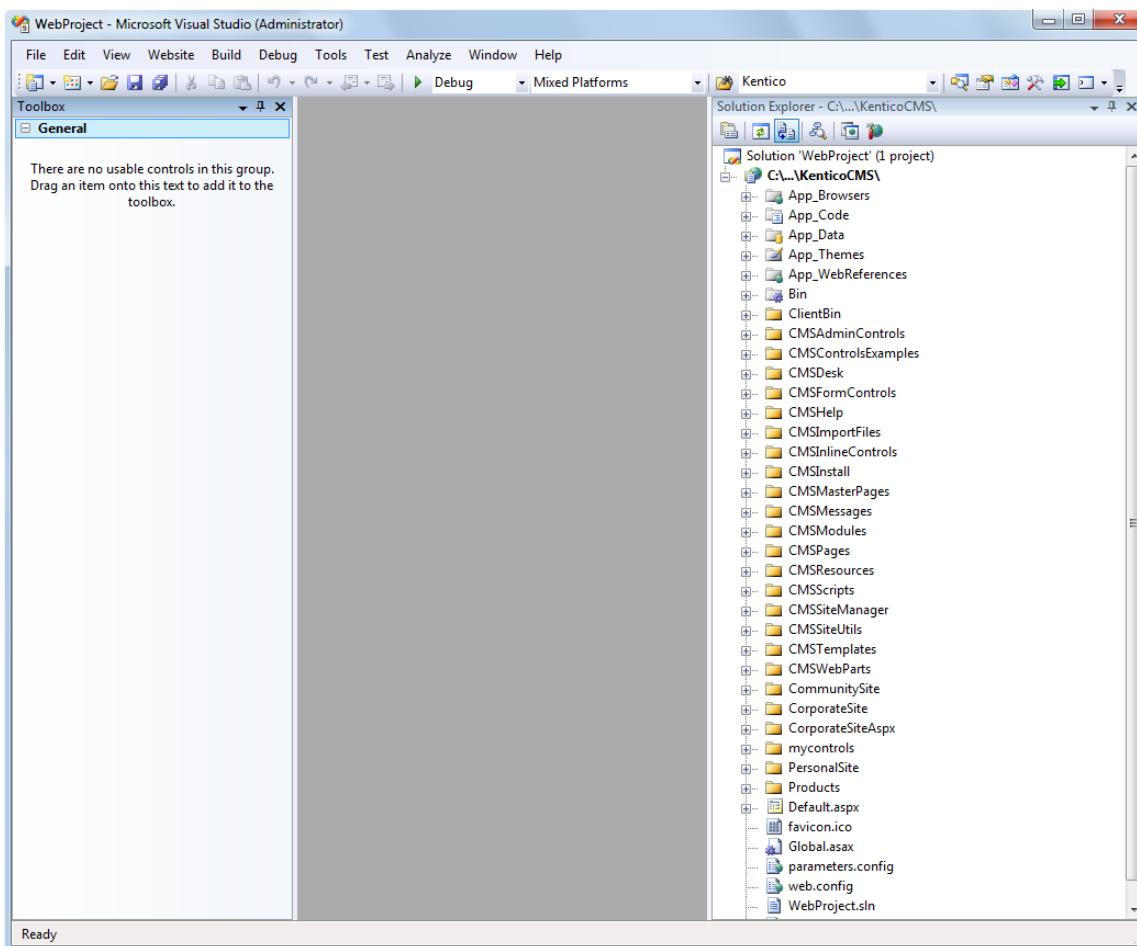
Click **Next** to start the installation process.



After the setup copies all files, you will see a link that opens the web application in your browser or the web project in Visual Studio. **Click the link.**

Opening the website in Visual Studio

If you chose Visual Studio installation, the project is opened in Visual Studio:



Choose **Debug -> Start without debugging** from the main menu. The site will be displayed in a new browser window using the built-in web server.



If you cannot open the website in Visual Studio

If the link for opening the project in Visual Studio doesn't work, you may need to start Visual Studio manually and choose **File -> Open -> Web Site...** and locate the project folder on your disk manually.

2.4 Database setup and Corporate Site

Now you should see the **Database setup** in your web browser.

In the first step, choose the SQL Server name or IP address. If you are using SQL Server 2005 Express Edition, the default server name is `.SQLEXPRESS` or `(local)\SQLEXPRESS`.

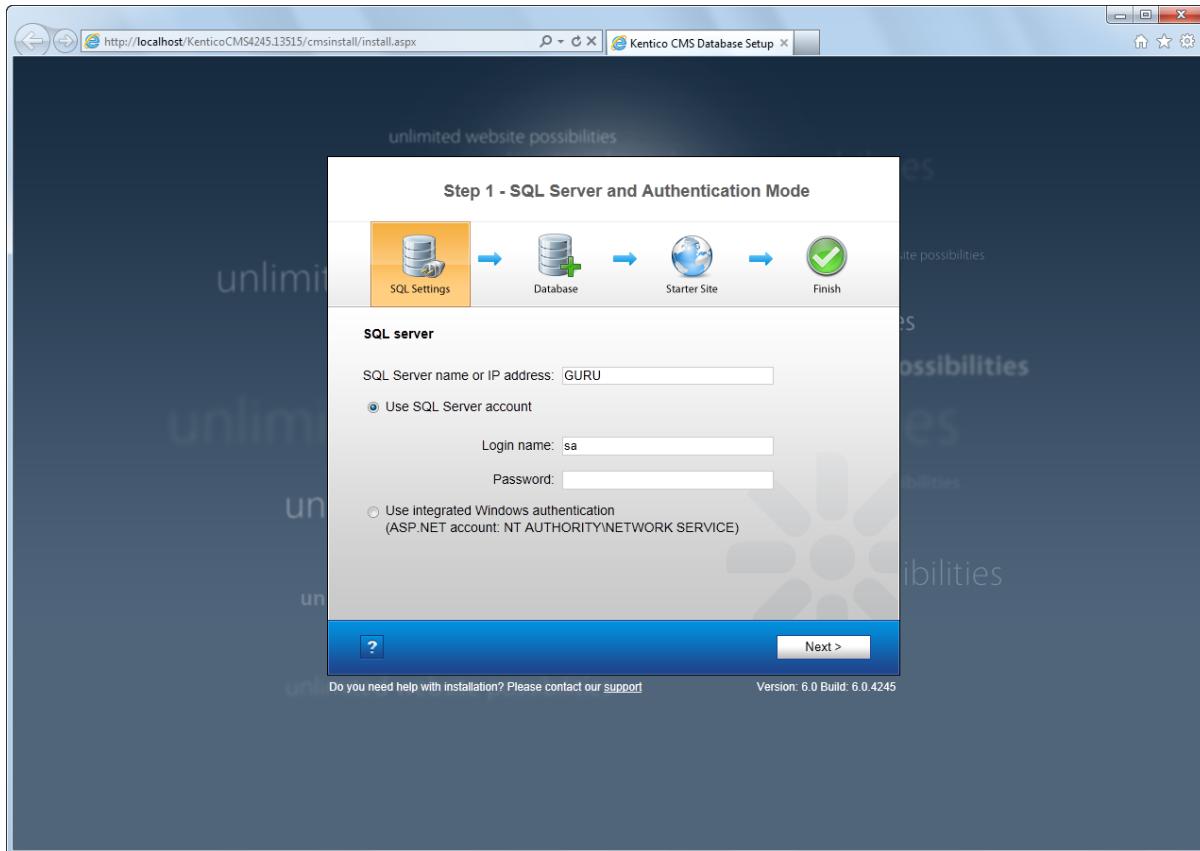
You can use either SQL Server authentication (recommended) or integrated Windows authentication.

- In case you use **SQL Server account**, you need to enter the user name (such as `sa`) and password.

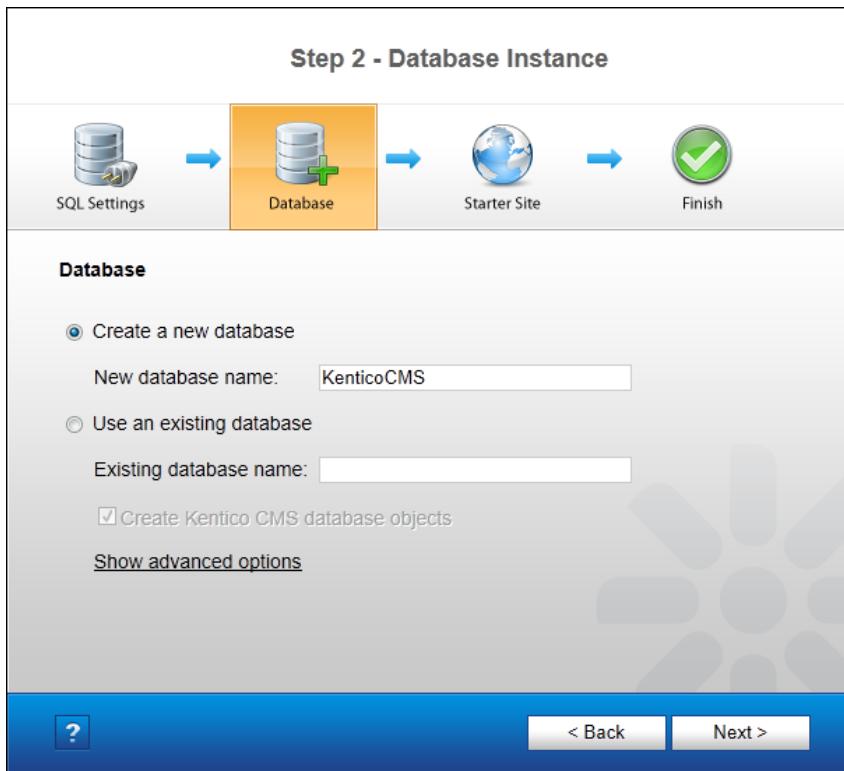
- In case you use **Windows authentication**, you need to ensure that the ASP.NET account of the name displayed in the brackets has an appropriate login name in your SQL Server.

Permissions for creating a new databases or for creating database objects in an existing database must be granted to the account.

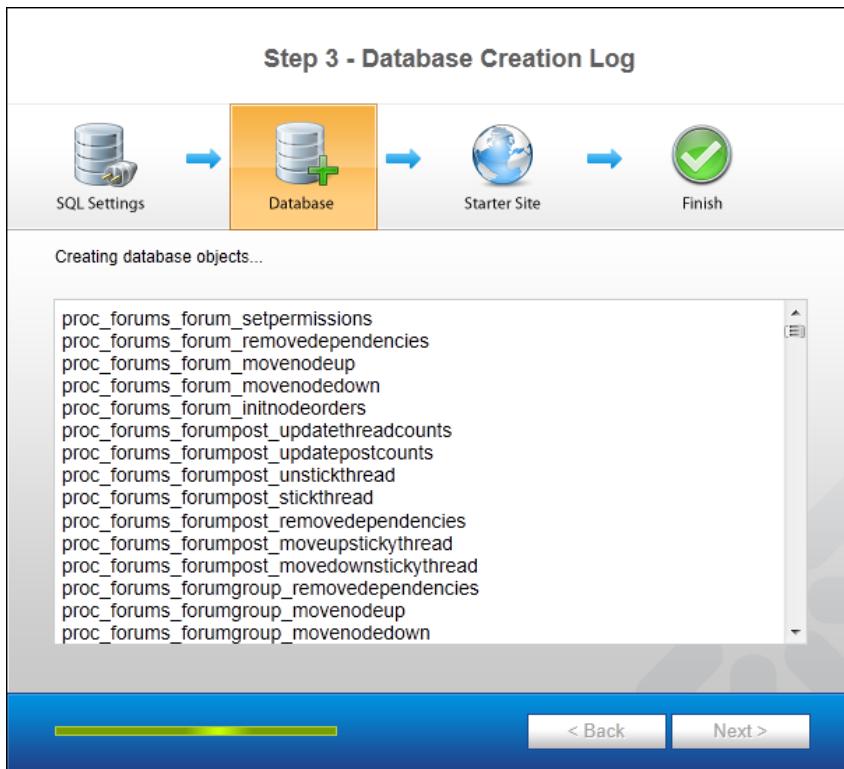
Click **Next**.



In the **Database Instance** step, choose **Create a new database**, enter the name of the new database into the **New database name** field and click **Next**.

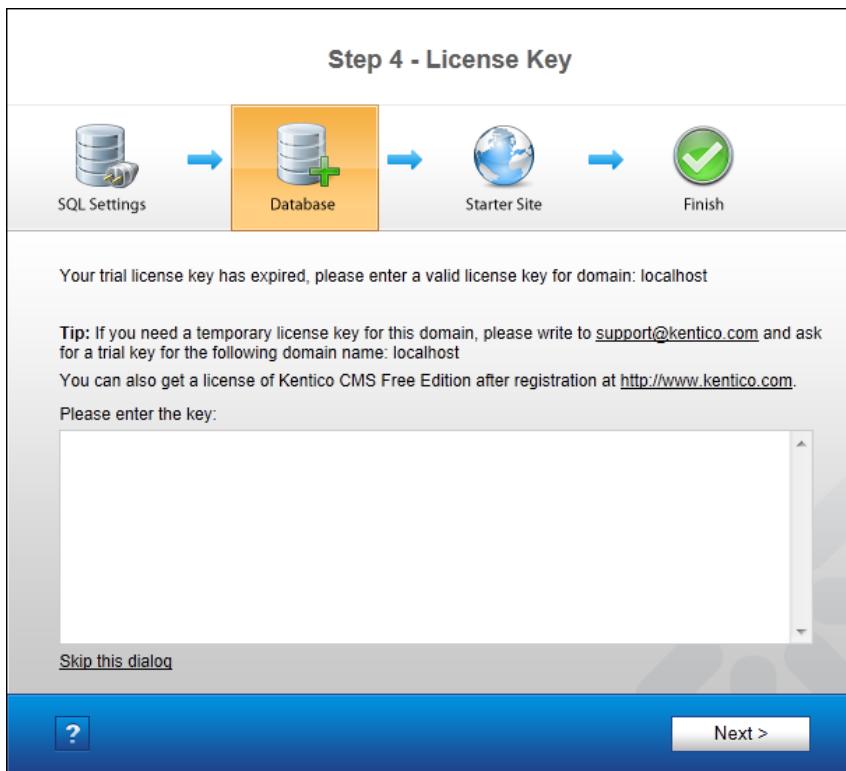


The database creation log will be displayed.



When the database is created, you will be asked to enter your license key. If you do not have a license

key yet, click **Next** to continue in trial mode. The functionality of the trial mode is the same as the full version.



In the **Starter Site** step, you can choose from the following options:

- **Choose a starter site:**
 - **Corporate Site (portal engine)** - this option installs the sample corporate site. This is **recommended** for most users, especially **for evaluators**.
 - **E-commerce Site** - this sample site can be used as a starting point for creating your own e-shop and shows the possibilities of Kentico CMS's E-commerce module.
 - **Personal Site** - this is a web template suitable for a simple personal site.
 - **Community Site** - complex web template suitable for community webs, showing Kentico CMS's social networking features in practice.
 - **Intranet Portal** - ready-to-use solution for company intranets with support of departments, workgroups, project management, etc.
 - **Blank Site** - this is a blank site without any content; you will use it to create a new site from scratch.
 - **Blank Site ASPX** - the same as above, but for ASPX page templates.
- **Continue to the New site wizard** - this option is recommended if you are starting a new site from scratch.
- **Import an existing Kentico CMS website** - use this option if you already created a website with Kentico CMS and need to import it into the new installation (e.g. on the production server).

For the purposes of this guide, please select the sample **Corporate Site** and click **Next**. You will see the confirmation and a link to your new website:

Step 5 - Starter Site

Choose starter site

Corporate Site

This is a web template for a general corporate site. It's used as a showcase of Kentico CMS capabilities and it can be used as a starting site that you modify as needed. It uses the portal engine and it's the recommended choice for developers who are new to Kentico CMS.

E-commerce Site

This is a web template for a simple E-commerce site. It's used as a showcase of Kentico CMS E-commerce module capabilities and it can be used as a starting site that you modify as needed. It uses the portal engine and it's the recommended choice for developers who are new to Kentico CMS.

Continue to the New site wizard

Import existing Kentico CMS website

[Next >](#)

A log showing the creation of the site will be displayed. When it's finished, you will be shown the final step that you can see in the screenshot below. Click the **Continue to the new website** link.

Step 7 - Finished

The site has been created successfully.

[Continue to the new website](#)

You will be redirected to the title page of the sample Corporate Site:

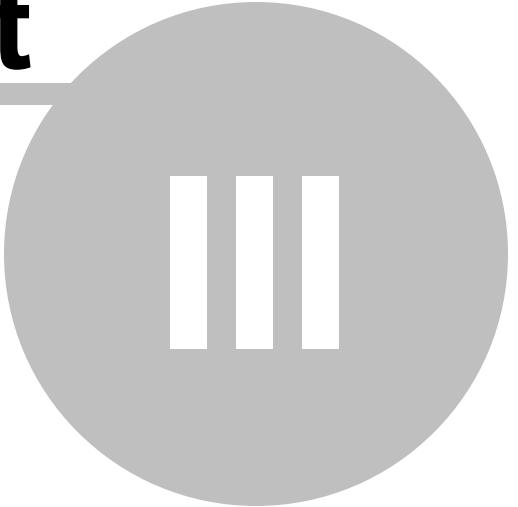
The screenshot shows a web browser window displaying the 'Corporate site - Home' page. The URL in the address bar is <http://localhost/KenticoCMS4246.16572/default.aspx>. The page has a blue header with the 'IT Company' logo and navigation links for Home, Services, Products, News, Community, Company, and Media. A 'Log on' button is in the top right. The main content area features a banner with the text 'Discover Unlimited Website Possibilities!' and a large yellow arrow graphic. Below the banner is a 'Learn more' button. To the left, there's a newsletter sign-up form with fields for First Name, Last Name, E-mail, and a 'Subscribe' button. To the right, there's a 'Latest news' section with a link to 'Community Website Section' and a poll section asking 'How do you like our new website?' with an 'Excellent' option selected. The footer contains a 'Where to learn more?' link.



Sample website

The Corporate Site website is only an example of a website you can create with Kentico CMS. You have full control over the site structure, design, page layout and functionality as you will see in the following chapters.

Part



III

Managing content

3 Managing content

3.1 User interface overview

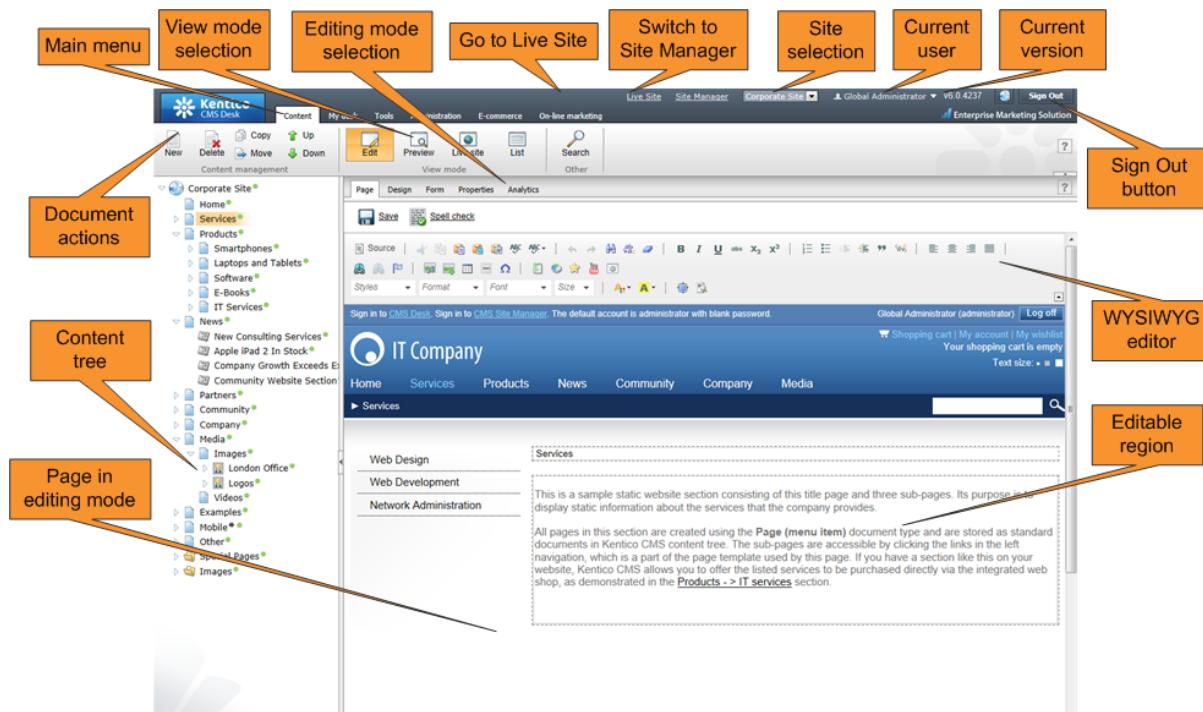
Click the **Sign in to CMS Desk** link at the top of the website or go to <http://<domain>/<virtualdirectory>/cmsdesk>. You will be asked for a user name and password.


Default user name and password

The default user name is **administrator**, the default password is **blank (no password)**

It's highly recommended that you change the password before you publish the website on the live server.

Once you sign in, you will see a splash screen, giving you some basic information. Click the **Continue** button, you will be redirected to the following page:



The user interface consists of the following main sections and features:

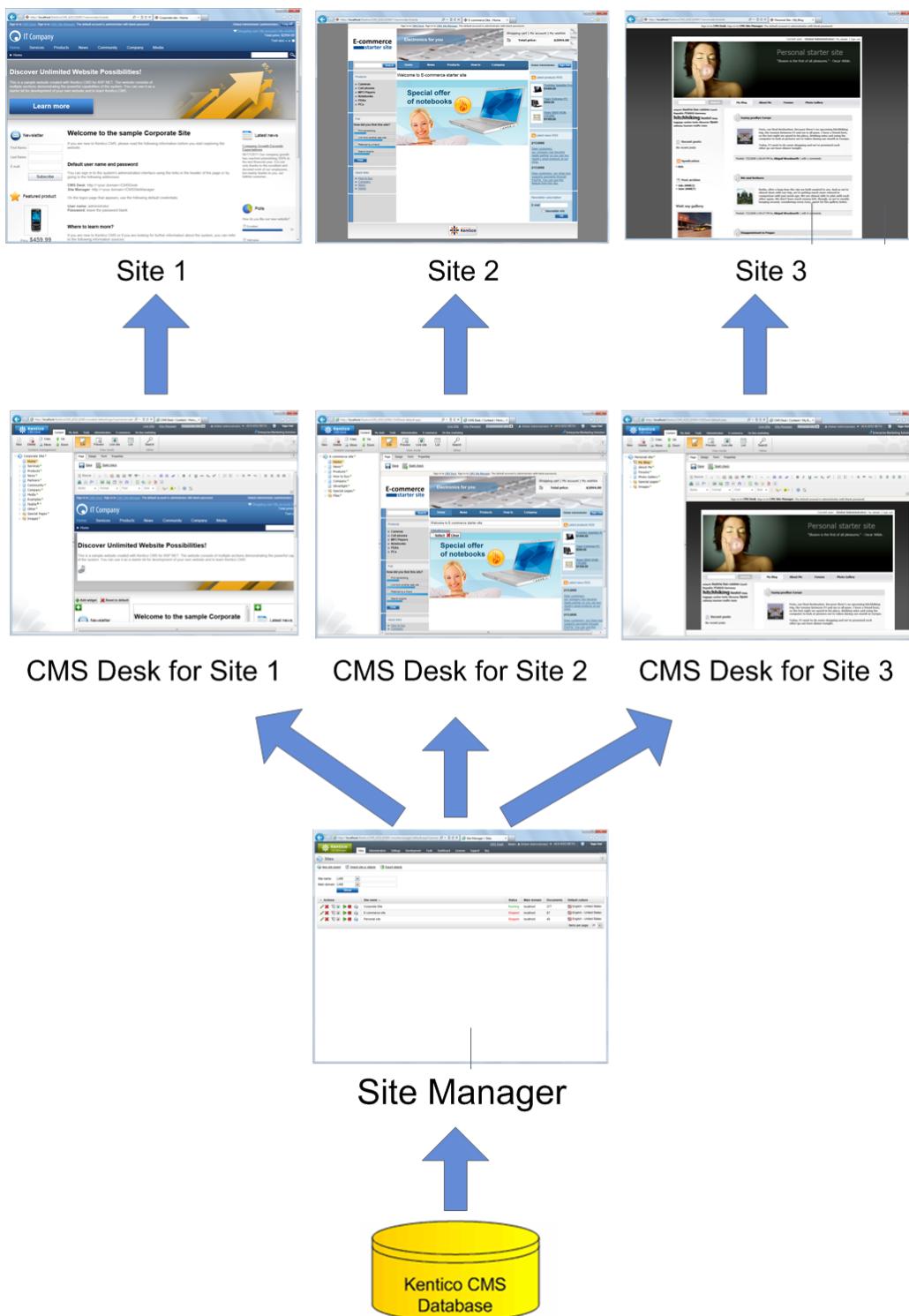
- **Main menu** with Content, My Desk, Tools and Administration tabs.
- **Content tree** that represents the site map of the website and allows you to organize the structure of documents and choose document that appears on the right side of the screen.
- **Document actions** toolbar with buttons for creating new documents, deleting, copying, moving and sorting documents.

- **View mode selection** - allows you to choose between editing, preview, live site view and list view.
- **Editing mode selection** - you can choose to edit page content, design the page template, edit the document fields, product properties or document properties.
- **Live Site** - this action redirects you to the title page of the currently edited website, logged under the same user account that you used to log into the user interface. This is a more convenient way than using the **Sign out** button and logging in on the live site afterwards.
- **Site Manager** - redirects you to Site Manager, the other part of the system's user interface. This option is only available to global administrators.
- **Site selection** - this drop-down list is used to select the currently edited website. Only those websites that the current user can edit are available in the drop-down list.
- **Current user** - user name of the current user.
- **Current version** - version of Kentico CMS.
- **Sign Out button** - clicking this button log you off the user interface and redirects you to the title page of the live site. This button is only displayed if *Forms authentication* is used. When using *Windows authentication*, the link is not displayed.
- **WYSIWYG editor** - allows you to edit text stored in **Editable regions**, change text formatting or insert graphics into the text.
- **Page in editing mode** - this is where you can view and edit the document selected in the content tree, in the mode selected in the view mode and editing mode toolbar.

CMS Desk and Site Manager

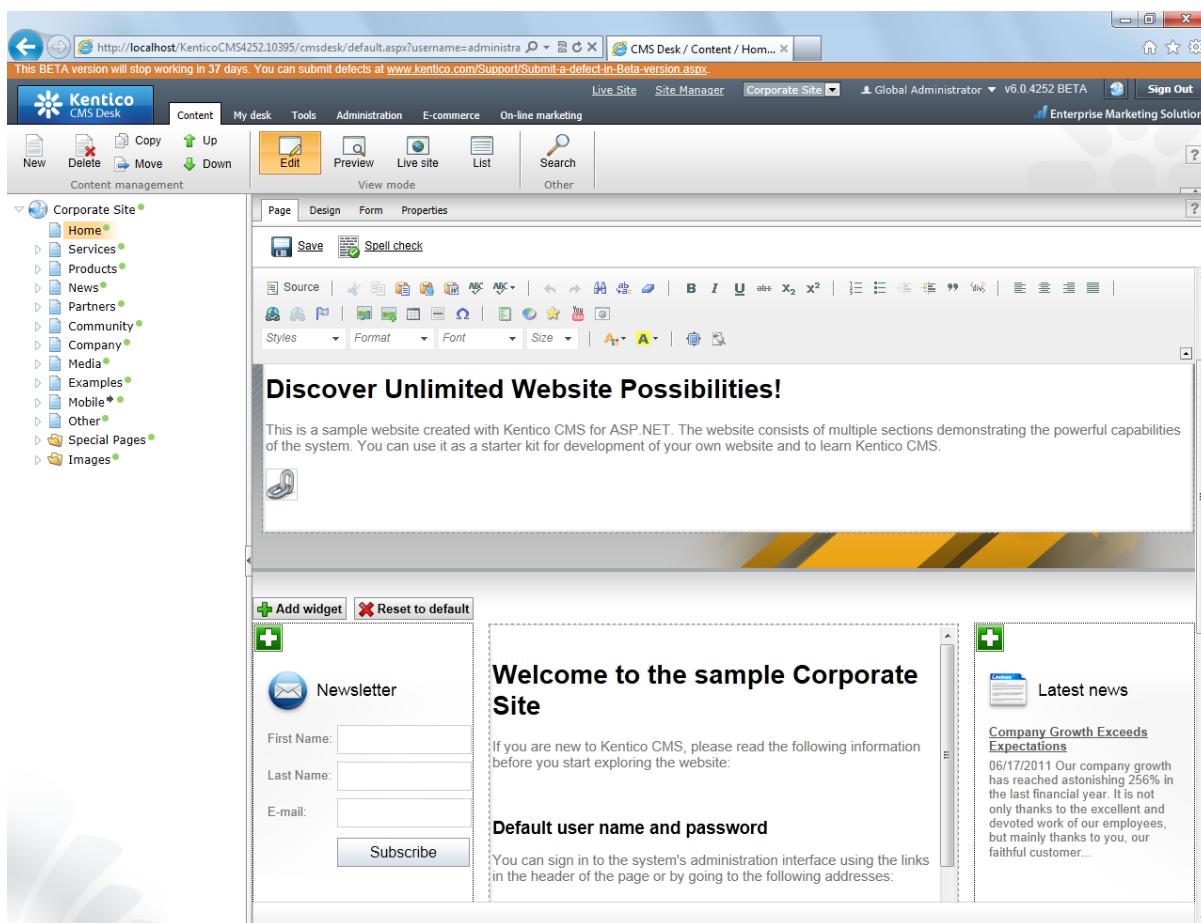
CMS Desk allows content editors to edit content of a single website. Developers and site administrators who need to manage settings, code and configuration of all websites, can also use the **Site Manager** interface.

The Site Manager interface is accessible either through the **http://<domain>/<virtualdirectory>/KenticoCMS/CMSSiteManager** URL, by clicking the **Site Manager** link at the top of the CMS Desk user interface or the **Sign in to Site Manager** link at the top of the live site. The following figure shows how the database, Site Manager, CMS Desk and websites are related:



3.2 Editing home page content

Now we will modify the home page content. Click **Home** in the content tree. You will see a page like this on the right side:



The page is now displayed in editing mode with an editable region. Delete all content from the editable region and enter the following text:

This is my first text.

You can then use the WYSIWYG editor toolbar at the top of the page to change the formatting of the text like this:

*This is my **first** text.*

Click the **Save** button at the top of the page or press **CTRL+S** to save the changes.

Now click the **Live site** button in the main toolbar. You will see the modified version of the home page as it's displayed to the site visitors.

Preview mode



If you click the **Preview** mode now, it will display the same content as the **Live site** mode. It works as a preview mode only if you set up workflow. Then, you can preview the latest modifications before they are published.

3.3 Creating a new page

Now we will create a new page under the Services section. Click **Edit** in the main toolbar to switch back to the editing mode. Click **Services** in the content tree. Click **New** in the main toolbar. You will see the following dialog that allows you to select the type of document you want to create under the currently selected document:

 **New document**

Please select new document type:

Document type name : LIKE

 [Page \(menu item\)](#)

 [Article](#)

 [Blog](#)

 [Cell phone](#)

 [E-book](#)

 [Event](#)

 [Event \(booking system\)](#)

 [FAQ](#)

 [File](#)

 [Folder](#)

 [Image gallery](#)

 [IT Service](#)

 [Job opening](#)

 [Knowledge base article](#)

 [Laptop](#)

 [News](#)

 [Office](#)

 [PDA](#)

 [Press release](#)

 [Product](#)

 [Product - Cell phone](#)

 [Product - Laptop](#)

 [Simple article](#)

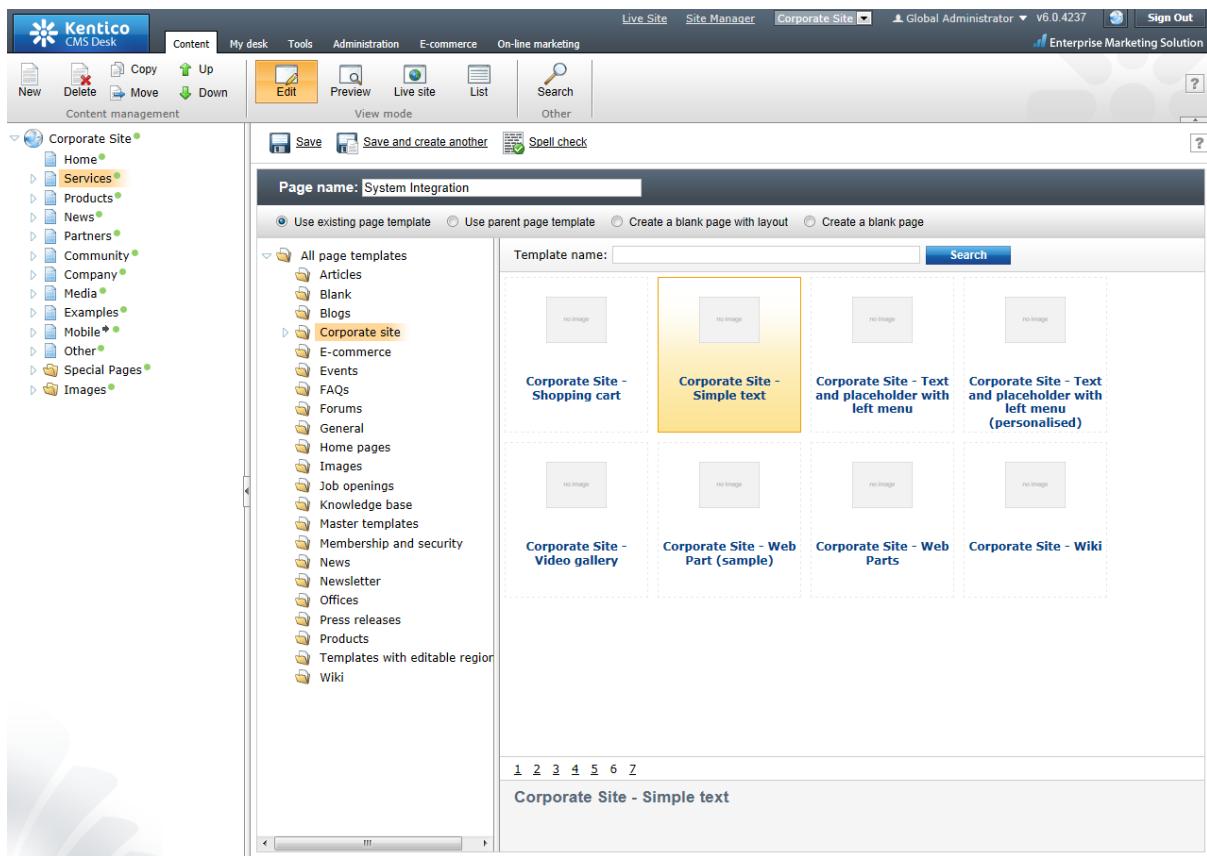
 [Smartphone](#)

 [Software](#)

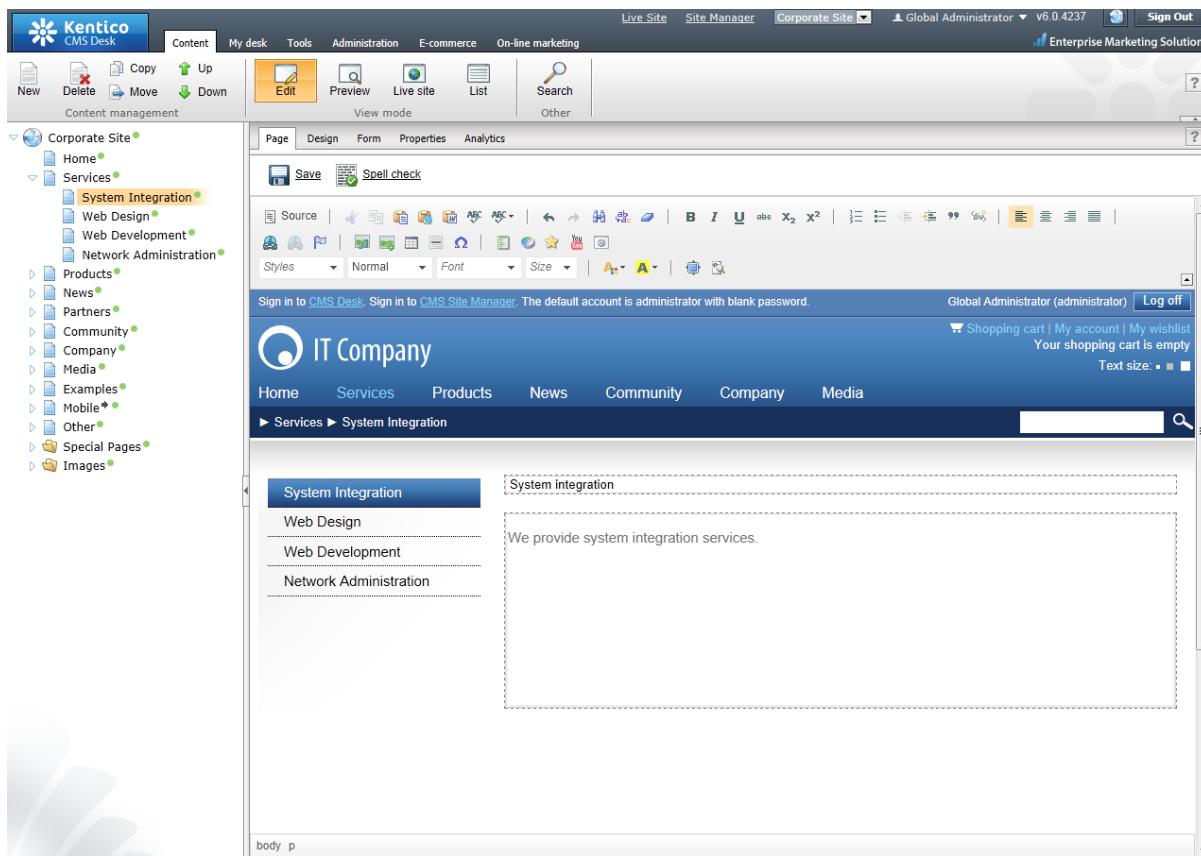
     

 [Link an existing document](#)

Click the **Page (menu item)** button. You will be redirected to the new page properties dialog. Enter *System Integration* in the **Page name** field and choose the **Corporate Site -> Corporate Site - Simple text** template:



Click **Save** to create the new page. The page is now created in the content tree and you can edit page content on the right. Enter some text in the editable regions and click **Save** again.



Now you may want to change the order of the items in the content tree on the left. Click the **Down** button in the main toolbar three times. The **System Integration** item is moved to the bottom of the section:



Click **Live site** in the main toolbar. You will see your new page as it is displayed to site visitors. Please note that the **System Integration** item is placed at the end of the left menu as you specified:

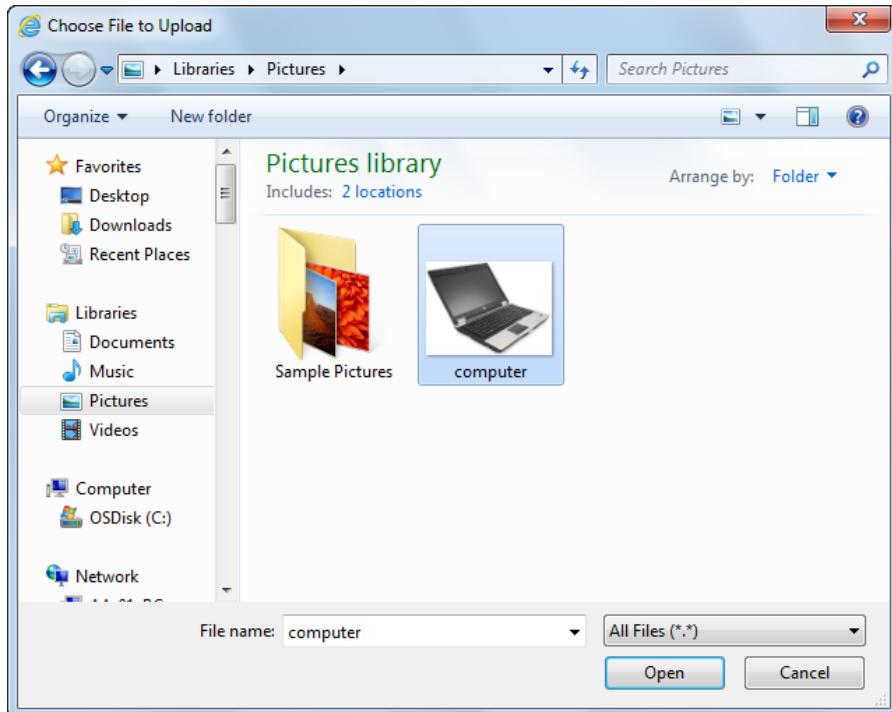
The screenshot shows a website for 'IT Company'. The header includes a logo, navigation links for Home, Services, Products, News, Community, Company, and Media, and a breadcrumb trail indicating the current location is Services > System Integration. On the left, there's a sidebar with links for Web Design, Web Development, Network Administration, and System Integration (which is highlighted). The main content area has a title 'System integration' and a subtext 'We provide system integration services.'

You have learned how to create a new page based on a pre-defined page template.

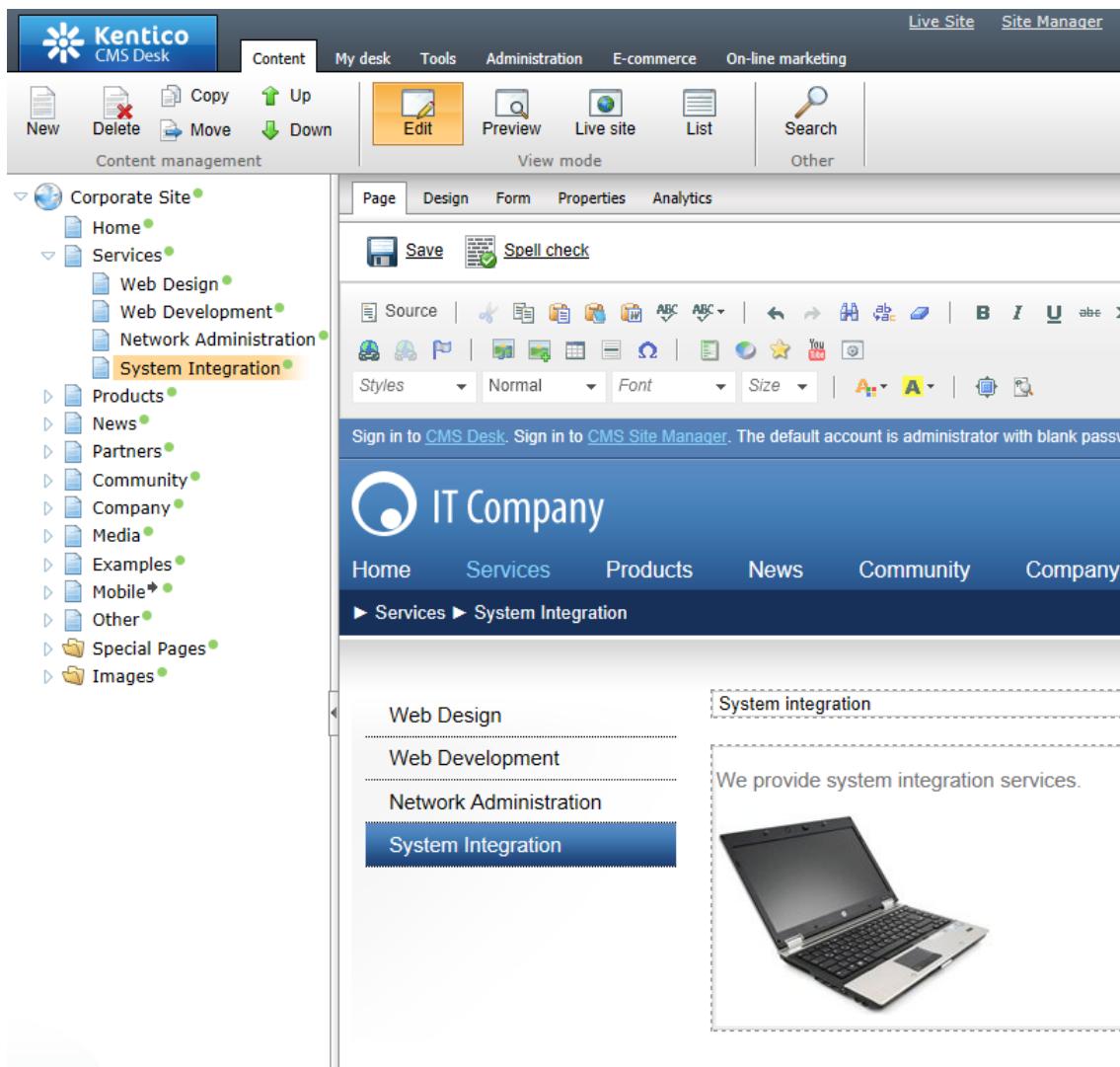
3.4 Inserting an image

Now we will upload and insert a new image to our new page. Click **Services -> System Integration** in the content tree. Switch to the **Edit -> Page** mode. Place the cursor into the main editable region, just below the text, and click the **Quickly insert image** (in the WYSIWYG editor toolbar.

The browser's **Choose file** dialog opens. Locate some suitable image file and click **Open**.



The image will be pasted to the editable region so that the page looks like this:



Click **Save** to save the changes. Click **Live site** to see the new version of your page.

You have learned how to upload an image and insert it into the text.

Allowing pop-ups for the website

If you are using a pop-up blocker, you may need to allow pop-up windows in your browser so that the Web part properties dialog as well as some other dialogs work correctly. This applies only to the administration interface, so the site visitors are not affected by this.

3.5 Creating a link

Now we will create a link between the **Services** page and our **System Integration** page. Click **Services** in the content tree and make sure you have the **Edit -> Page** mode selected.

Add a new item in the bulleted list called **System integration**:

The screenshot shows the Kentico CMS content tree on the left with a node for 'Services'. The main content area displays a page titled 'Services' with the following content:

We provide professional services in the following areas:

- [Web development](#)
- [Web design](#)
- [Network administration](#)
- [System integration](#)

This is a sample static website section consisting of this title page and three sub-pages. Its purpose is to display static information about the services that the company provides.

All pages in this section are created using the **Page (menu item)** document type and are stored as standard documents in Kentico CMS content tree. The sub-pages are accessible by clicking the links in the left sidebar.

Select the whole line:

The screenshot shows the Kentico CMS content tree on the left with a node for 'Services'. The main content area displays a page titled 'Services' with the following content:

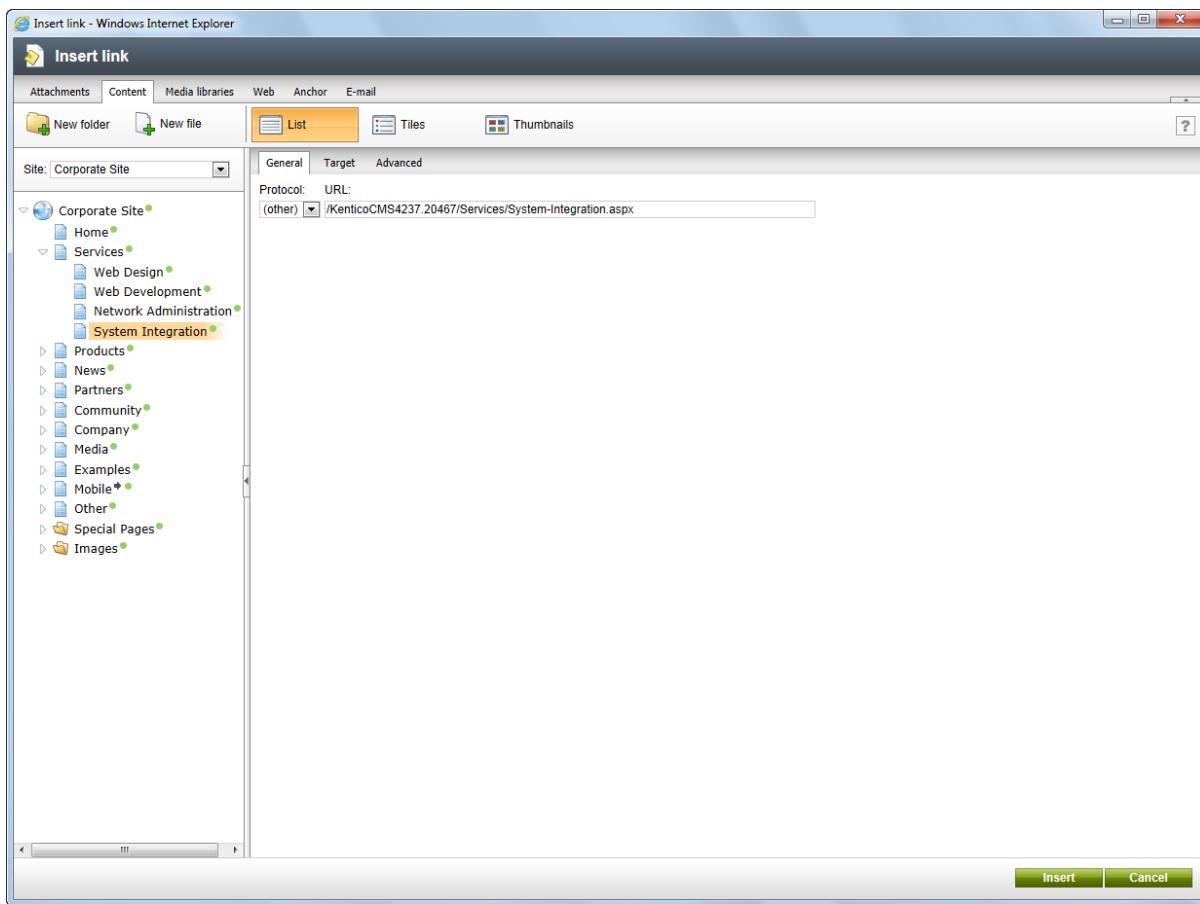
We provide professional services in the following areas:

- [Web development](#)
- [Web design](#)
- [Network administration](#)
- [System integration](#)

This is a sample static website section consisting of this title page and three sub-pages. Its purpose is to display static information about the services that the company provides.

All pages in this section are created using the **Page (menu item)** document type and are stored as standard documents in Kentico CMS content tree. The sub-pages are accessible by clicking the links in the left sidebar.

... and click the **Insert/Edit Link** (link icon) button in the WYSIWYG editor toolbar. The **Insert link** dialog opens. Switch to the **Content** tab, select the **Services -> System Integration** page from the content tree and click **Insert**.



The text is now marked as a link:

A screenshot of the Kentico CMS 6.0 website editor. The top navigation bar has a search bar and a magnifying glass icon. The main content area has a sidebar with "Services" and a "Services" section. The sidebar contains links for "Web Design", "Web Development", "Network Administration", and "System Integration". The main content area has a heading "Services" and a paragraph "We provide professional services in the following areas:" followed by a bulleted list: "• Web development", "• Web design", "• Network administration", and "• System integration". Below that is a note: "This is a sample static website section consisting of this title page and three sub-pages. Its purpose is to display static information about the services that the company provides." At the bottom, there is a note: "All pages in this section are created using the Page (menu item) document type and are stored as standard documents in Kentico CMS content tree. The sub-pages are accessible by clicking the links in the left sidebar." A vertical scroll bar is visible on the right side of the content area.

Click **Save** and choose the **Live site** mode. When you click the **System integration** link now, you are redirected to the new page.

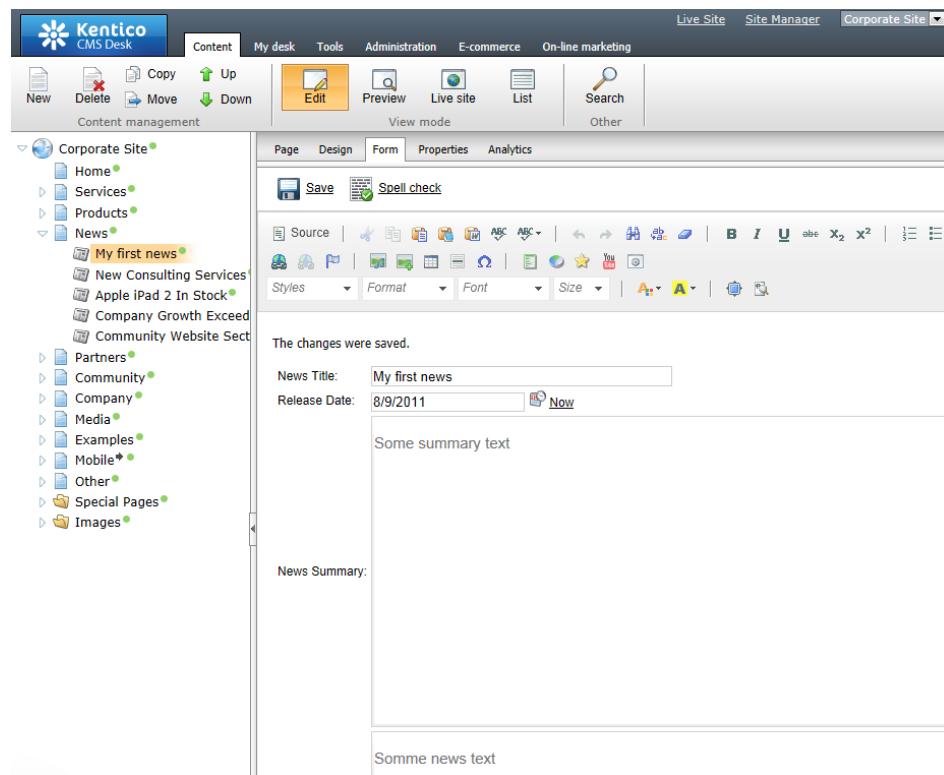
You have learned how to create a link between pages.

3.6 Creating a news item

Now you will learn how to create a news item. Click **Edit** in the main toolbar. Click **News** in the content tree and click **New**. Choose to create a new document of type **News**. You are redirected to the form that allows you to define news item sections: title, summary, full text and release date. Enter the following text:

- **News title:** My first news
- **Release date:** click Now
- **News summary:** Some summary text.
- **News text:** Some news text.

Click  **Save** to save the new document.



As you can see, the editing mode is now set to **Form** instead of **Page**. It means you do not edit editable regions on the page, but rather the structured data related to the document. The **Form** tab is used for editing the **structured data related to the document**. The document fields are fully customizable for every document type.

When you click **Live site**, you will see the news item displayed using a pre-defined transformation on both the **News** and **News -> My first news** page:

The screenshot shows a Kentico CMS 6.0 news item page. At the top is a blue header bar with the logo 'IT Company' and navigation links for Home, Services, Products, News, and Community. Below the header, a dark blue breadcrumb bar shows 'News > My first news'. The main content area has a light gray background. It features a white box containing a graphic of a newspaper titled 'Daily News' with the subtitle 'Today's Headlines'. To the right of the graphic is a text box with the heading 'Some summary text'. Below the graphic and text box is another text area with the heading 'Some news text'.

You have learned how to add a news item and how to use the editing form for structured documents.

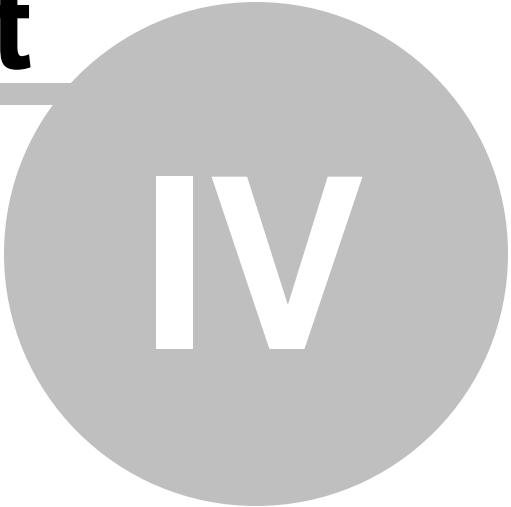
Page versus Form

There are two aspects of a document: content stored in editable regions on the page and data stored in form fields. The following table compares both approaches:

	Editable regions on the page	Form
Content structure	Simple content structure, only text content.	Complex content structure, typed data, such as text, date-time, numbers, etc.
Validation	Only basic validation rules for minimum and maximum length.	Complex validation rules, including regular expressions and custom form controls with custom validation code.
Display	The content is displayed in the context of the page as it is displayed in the editing mode.	The content is displayed using XSLT or ASCX transformations using special controls or web parts.
Storage	The content is stored in a single XML document in the document properties.	The content is stored in a separate database table. Each field has its own column. The data can be easily modified using SQL queries or API.
Examples of use.	Home page, contact page. Generally: pages with	News, product specification, event details, job openings, etc.

	<p>simply structured or unstructured text-based content.</p> <p>The editable regions are usually used for documents of type Page (menu item).</p>	<p>Generally: pages with structured content where you need to separate content from design and keep the content in its original data type.</p> <p>Form-based content is usually used for documents of type News, Product, Article, etc.</p>
--	--	--

Part



IV

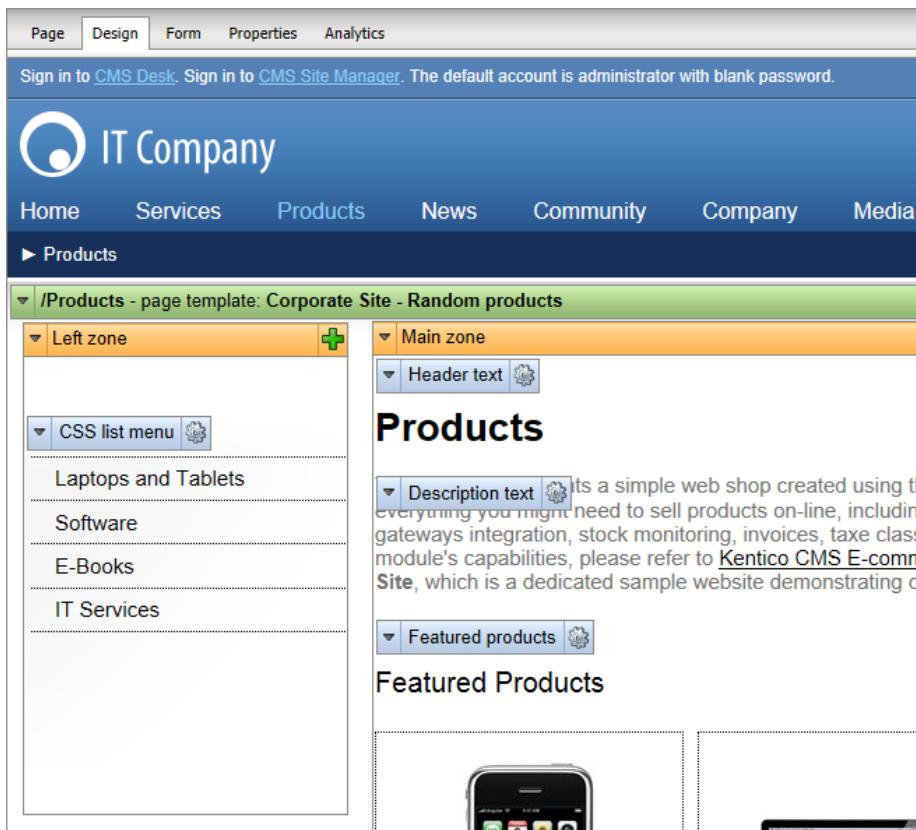
Site Development Overview

4 Site Development Overview

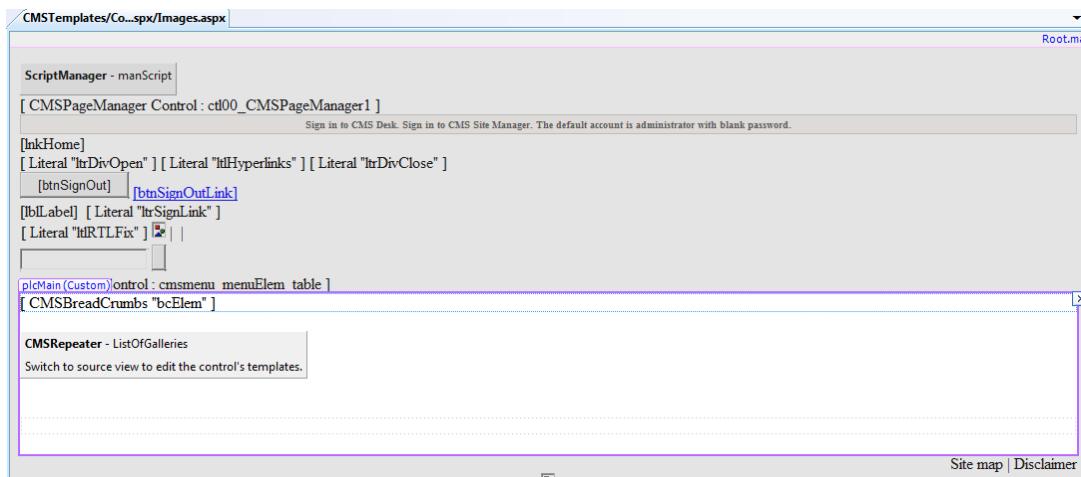
4.1 Site Development Overview

Kentico CMS provides two development models and you can choose which one suits you better:

- **Portal Engine** - this model allows you to build websites using a portal engine. It's the recommended way for most developers since it doesn't require programming and using Visual Studio. You can easily build websites using web parts in the **browser-based** user interface.



- **ASPX Templates** - this model can be chosen by advanced ASP.NET developers who prefer to create the website using standard ASP.NET architecture and using standard development tools, such as **Visual Studio**. You need to be familiar with ASP.NET development and have at least basic programming knowledge of C# or VB.NET.



Both approaches are fully supported and they provide the same level of flexibility and extensibility. We recommend that you use the portal engine model, but if you are a hard-core .NET developer and do not trust portal engines, you may want to use ASPX templates.

If required, both models can be combined on a single website. For example, you can integrate ASPX templates into a portal engine website, or even custom ASPX pages implementing your own applications. On the other hand, special areas can be defined on ASPX templates where editing through the portal engine is possible.

The following table compares the portal engine and ASPX templates:

	Portal Engine	ASPX Template
How you work	You build websites using the browser-based interface. No programming knowledge is required for common tasks.	You build ASPX page templates that are used to display content from Kentico CMS. At least basic programming knowledge of ASP.NET and either C# or VB.NET is required.
How you assemble pages	You use built-in or custom web parts that you place into customizable page layouts.	You use built-in or custom ASP.NET server controls that are placed on the ASPX pages. These are standard ASPX pages and they are part of the website project that you can open in Visual Studio. You can also place web parts (which are actually standard ASCX user controls) on page templates if the functionality is not available as a server control.
Master pages and visual inheritance	Sub-pages inherit the content from the parent pages by default (so called "visual inheritance"). The inheritance can be optionally broken if you want to create a page without parent content.	All page templates (.ASPX pages) may use a master page, which is a standard ASP.NET master page (.master file).

		The pages do not inherit content from their parents, they only inherit content from the master page (if it's used).
Custom code integration and extensibility	<p>You can create your own user controls and web parts if you need to integrate specific functionality.</p> <p>You can add any custom controls and code to the web parts or user controls that you use on your website.</p> <p>You can also use standard ASPX pages within your portal engine-based website.</p>	You build standard ASPX pages with code-behind which means you can use any custom controls and code on the page in Visual Studio.
Advantages	<ul style="list-style-type: none"> • Easier and faster way of building a website. • ASP.NET programming knowledge is not required for common tasks. • You can build the whole website very quickly, using only the web browser. 	<ul style="list-style-type: none"> • Standard ASP.NET architecture. • You can use your favorite development tools, such as Visual Studio for all changes.
Disadvantages	<ul style="list-style-type: none"> • Proprietary architecture and development. 	<ul style="list-style-type: none"> • Requires ASP.NET programming knowledge.



Is Kentico CMS just another portal engine?

Now you may ask what's the difference between Kentico CMS and DotNetNuke or SharePoint.

Well, the main difference is the **flexibility**. Kentico CMS gives you full control over:

- site structure
- site navigation
- page layout
- design
- content structure

Also, it's important to explain that Kentico CMS is a **content management system**, not only a portal engine. It provides features of advanced CMS systems, such as:

- content repository with a logical tree hierarchy of documents
- content/design separation
- custom document types with custom fields
- workflow and versioning
- content locking (check-out, check-in)
- multilingual content
- content preview and content staging
- document-level permissions with permission inheritance
- full-text search in all content

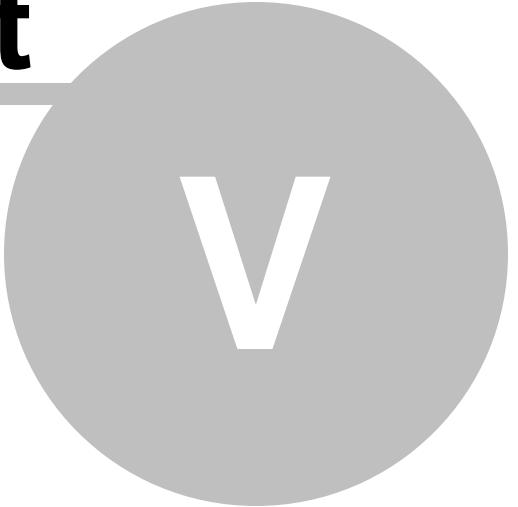
- document management features for uploaded files

Moreover, Kentico CMS comes with many **professional and flexible built-in modules out-of-the-box**, including Newsletters, On-line forms, Forums, E-commerce, Staging, Image gallery, Event calendar, Events, Blogs, Polls and others

It means you do not need to purchase third-party modules with inconsistent user and programming interface, but you get everything from a single source, with complete documentation.

The rest of this tutorial explains the ASPX template approach. If you want to use ASPX templates, please read the Tutorial for ASPX page templates.

Part



V

Creating pages using ASPX templates

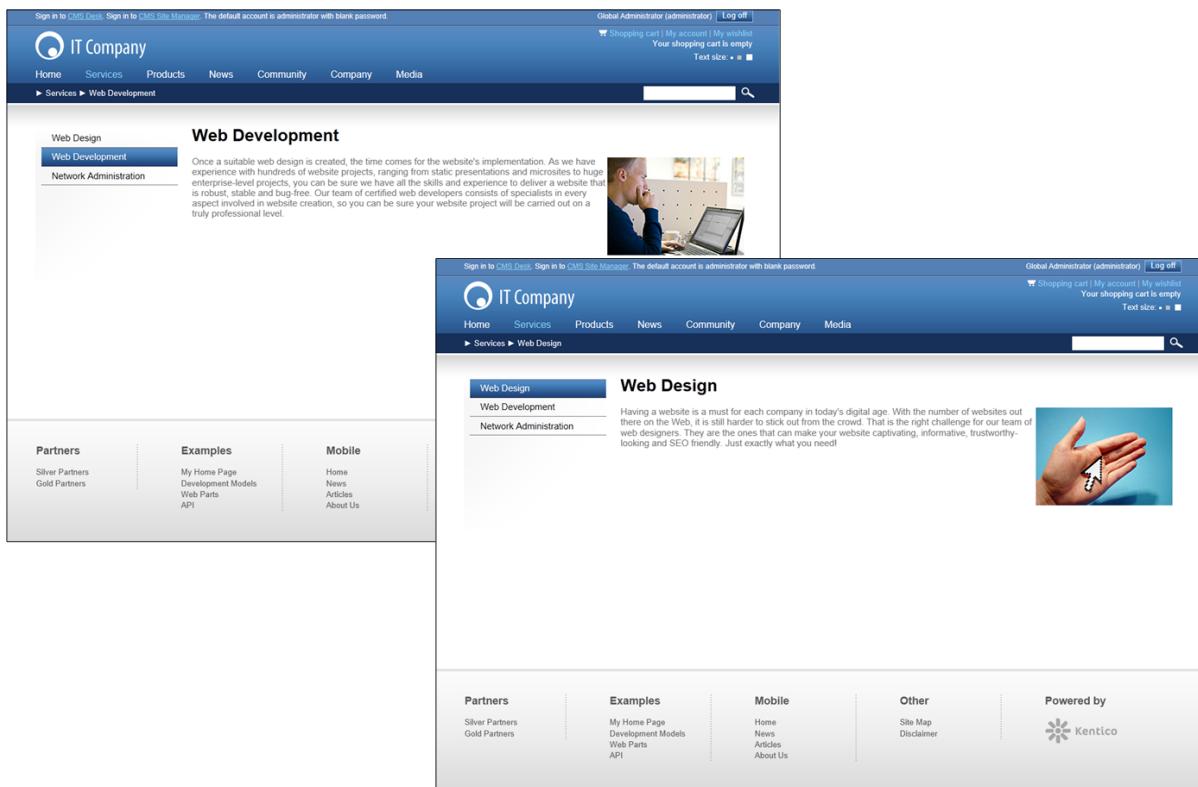
5 Creating pages using ASPX templates

5.1 ASPX page templates

If you are familiar with ASP.NET development in Visual Studio, you may choose to develop websites using standard ASPX page templates. ASPX page templates in Kentico CMS are standard ASP.NET pages that display content from Kentico CMS. They receive the **aliasPath** URL parameter that tells the page template which page should be displayed.

What is a page template?

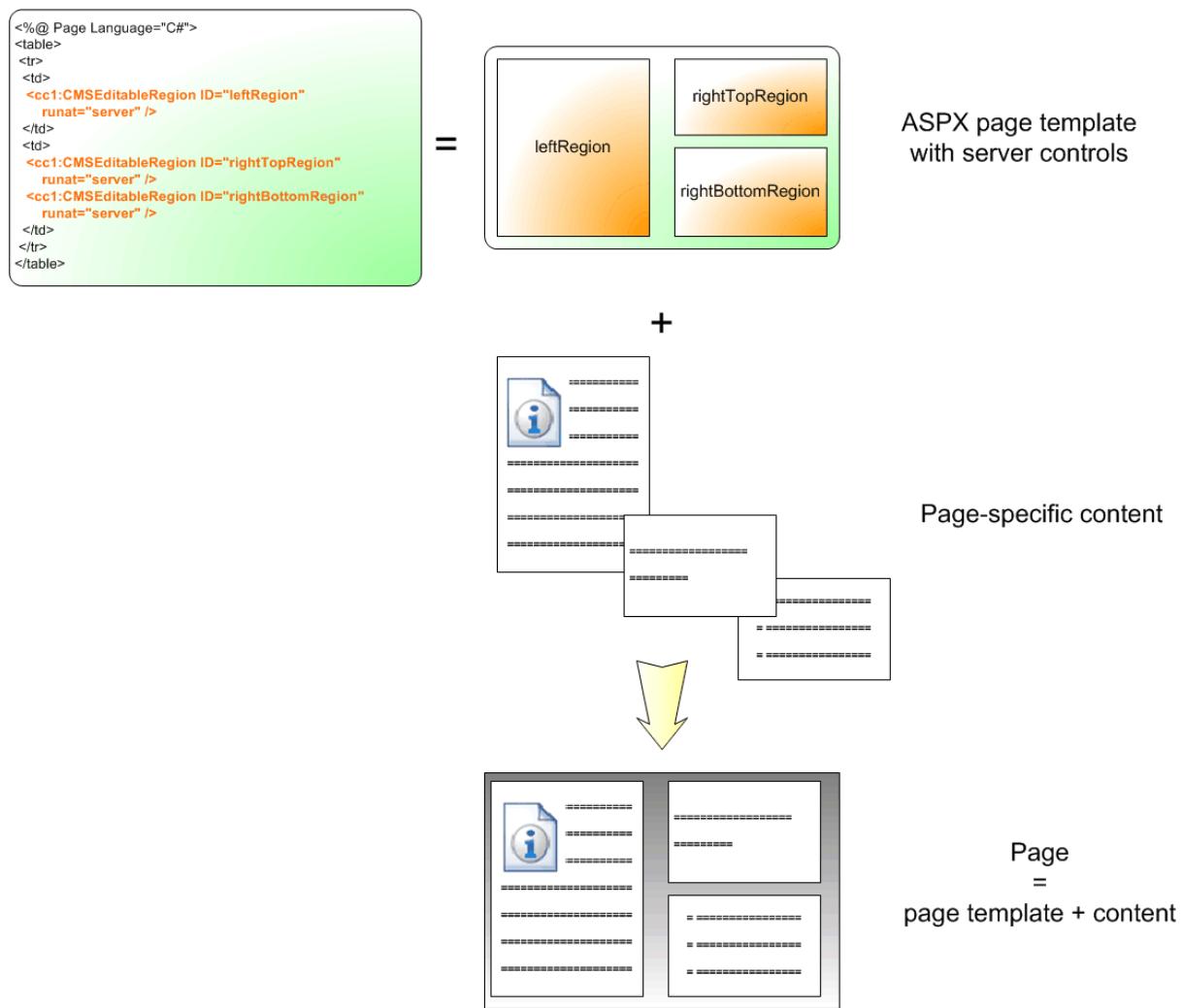
Every web page is based on some page template. The page template can be specific for a single web page ("ad hoc" page template) or it can be re-used for several pages. The following picture shows an example of two pages that use the same page template:



As you can see both of them use the same header, main menu, sub-menu, content structure and footer - they are based on the same **page template**. In this way, you can create multiple pages using the same design.

What does the ASPX page template consist of?

The page template is a combination of static HTML code and ASP.NET server controls (or user controls) that render dynamic content. The following figure illustrates how an ASPX page template and page content are combined to display a page:



As you can see, the ASPX page template is a standard page that may contain HTML code, CMS server controls or any other controls. You can also use code behind (in both VB.NET and C#) to modify page behavior and add custom functionality.

How is the ASPX page template processed?

When a user requests some page, such as `/services/web-development.aspx`, the system calls the assigned page template with the **aliasPath** URL parameter, which specifies what content (which page from the content tree) should be displayed using the given template:

User requests page
[http://localhost/CorporateSite/Our Company.aspx](http://localhost/CorporateSite/Our%20Company.aspx)



The request is processed by Kentico CMS URL rewriting engine.

The engine recognizes that page /Our Company should be displayed using the SingleArticle.aspx template.

The path is rewritten to:

[http://localhost/CorporateSite/SingleArticle.aspx?aliasPath=/Our Company](http://localhost/CorporateSite/SingleArticle.aspx?aliasPath=/Our%20Company)



The SingleArticle.aspx page is executed.

The CMSPageManager control gets the aliasPath parameter from the URL and loads content for page /Our Company. It then displays the text using appropriate CMSEditableRegion controls. Your custom code is executed as usual.



The output is (optionally) processed by output filter that fixes some form settings and improves XHTML-compatibility.



Final page is displayed to the visitor.

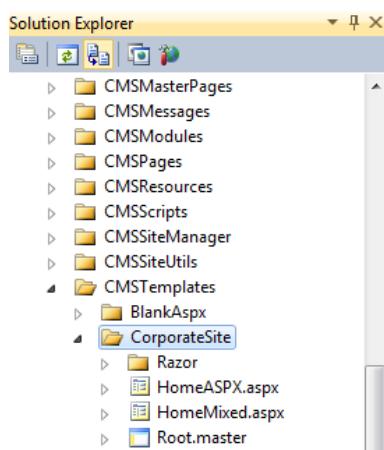
The built-in Kentico CMS controls understand the **aliasPath** parameter in the URL and render the appropriate content automatically.

As you can see, the system uses standard ASP.NET architecture. If you developed the website without Kentico CMS, you would most likely use URLs like this: `/news.aspx?newsid=127` which is similar to `/news.aspx?aliaspath=/news/november news.aspx` as used in Kentico CMS. Kentico CMS also uses friendly URLs in format `/news/your-first-news.aspx` that are better for search engine optimization.

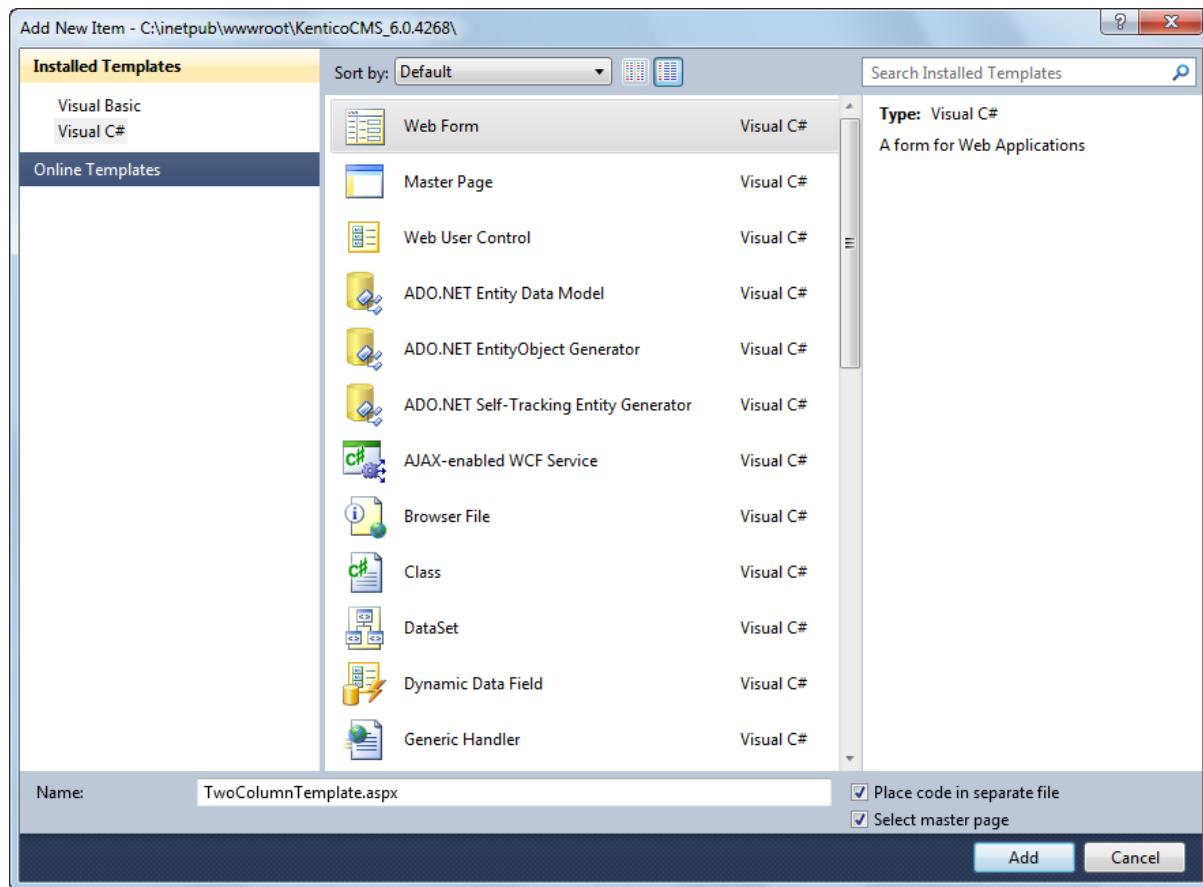
5.2 Creating a simple ASPX page template

Now you will learn how to create a new ASPX page template. We will create a new page with two columns that will contain editable regions.

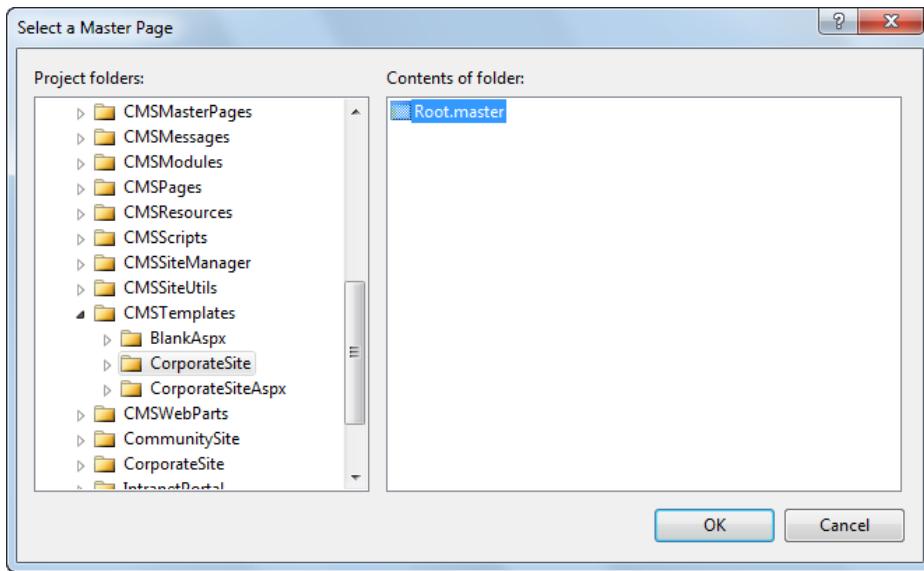
1. Open the web project in Visual Studio. You can open it either using the **WebProject.sln** file or using **File > Open > Web site** in the menu.
2. Now right-click the **CMS Templates -> CorporateSite** folder in the Solution Explorer and select **Add New Item**:



3. Choose to create a new **Web form** and call it *TwoColumnTemplate.aspx*, check the **Select master page** box and click **Add**.



4. The **Select a Master Page** dialog appears. Choose the folder **CMSTemplates/CorporateSite**, select the **Root.master** file and click **OK**.



Writing the ASPX code

5. Open the **Source** view of the newly created ASPX page and add the following code inside the **<asp:Content>** control:

```
<table width="100%">
<tr>
<td width="50%">
<cms:CMSEditableRegion ID="txtLeft" runat="server" DialogHeight="400"
    RegionType="HtmlEditor" RegionTitle="Left column" />
</td>
<td width="50%">
<cms:CMSEditableRegion ID="txtText" runat="server" DialogHeight="400"
    RegionType="HtmlEditor" RegionTitle="Right column" />
</td>
</tr>
</table>
```

The **<asp:Content>** control specifies that this content will be loaded into the master page (as defined in the Root.master file). As you can see, you can use the standard ASP.NET concept of master pages.

The **CMSEditableRegion** control defines an editable region that will be displayed as an HTML editor on the **Content -> Edit -> Page** tab of **CMS Desk**. On the live site, it ensures that the entered content is displayed on the page.

Please note: this example uses a table layout. If you prefer a CSS layout, you can simply replace the surrounding HTML code with **<DIV>** elements. As you can see, you have full control over the content.

6. Now we need to modify the code behind. Switch to the code behind file and add the following namespace:

[C#]

```
using CMS.UIControls;
```

7. The last step is to modify the class from which our page is inherited. Change the following code:

[C#]

```
public partial class CMSTemplates_CorporateSite_TwoColumnTemplate : System.Web.UI.  
Page
```

to this:

[C#]

```
public partial class CMSTemplates_CorporateSite_TwoColumnTemplate : TemplatePage
```

so that the page can be used as a page template in Kentico CMS.

Please keep in mind that the name of the class must be identical to the value of the **Inherits** attribute of the `<%@ Page %>` directive on the ASPX page. This is case sensitive.

Registering the ASPX page as a page template

Now that we have created a new ASPX page, we need to register it in Kentico CMS as a page template, so that it can be used by content editors.

8. Sign in to **Site Manager** and go to **Development -> Page templates**. Select the **Corporate Site/Examples** folder and click  **New template**. Enter the following values:

- **Template display name:** Two column template
- **Template code name:** TwoColumnTemplate

Click **OK**. The template will be created and its **General** tab will be displayed. Since the template was defined in Visual Studio as a standard ASPX page, the **Template type** must be set to **ASPX** (this is the case by default). Now enter the following value into the **File name** field:

`~/CMSTemplates/CorporateSite/TwoColumnTemplate.aspx`

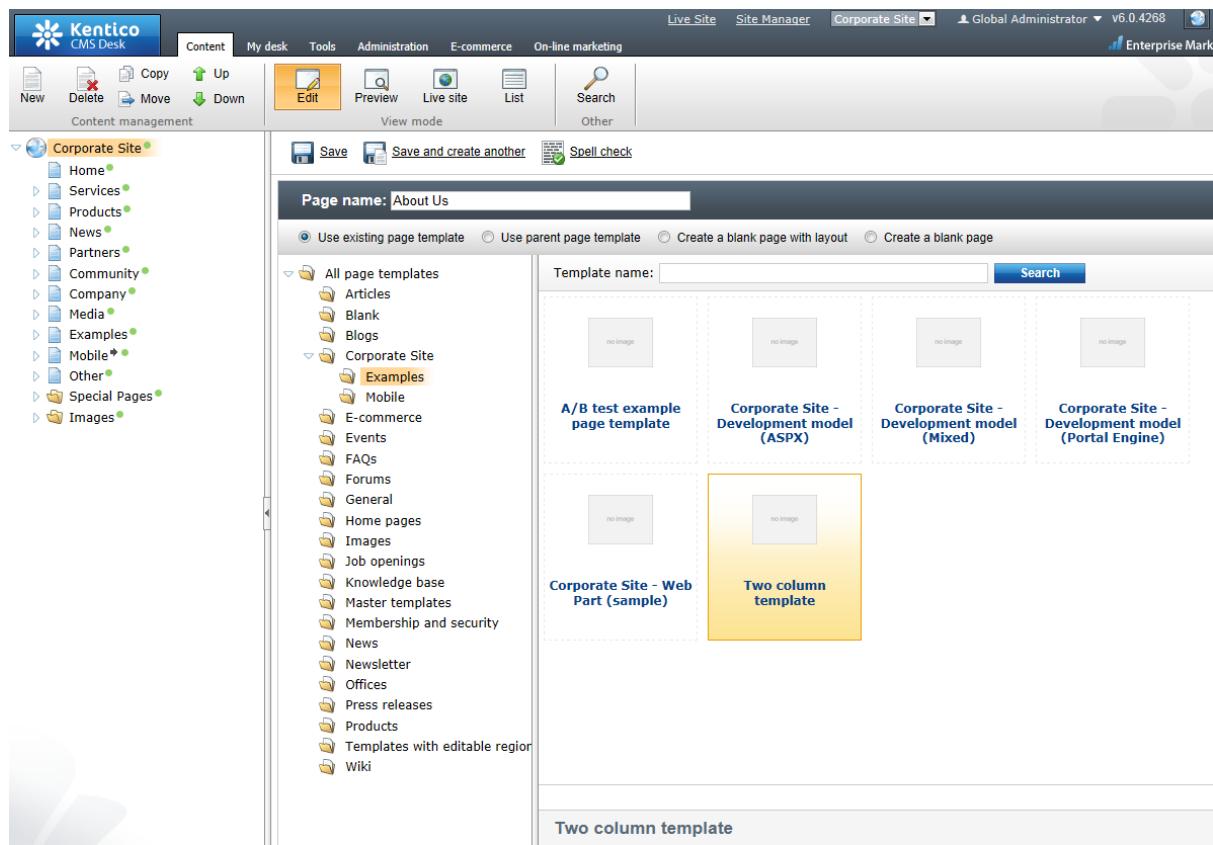
It is the virtual path of our ASPX page. Alternatively, the **Select** button can be used to manually select the file. Click  **Save**.

Save the changes.

9. Now switch to the **Sites** tab and click the **Add sites** button. Choose your site to assign the page template to it.

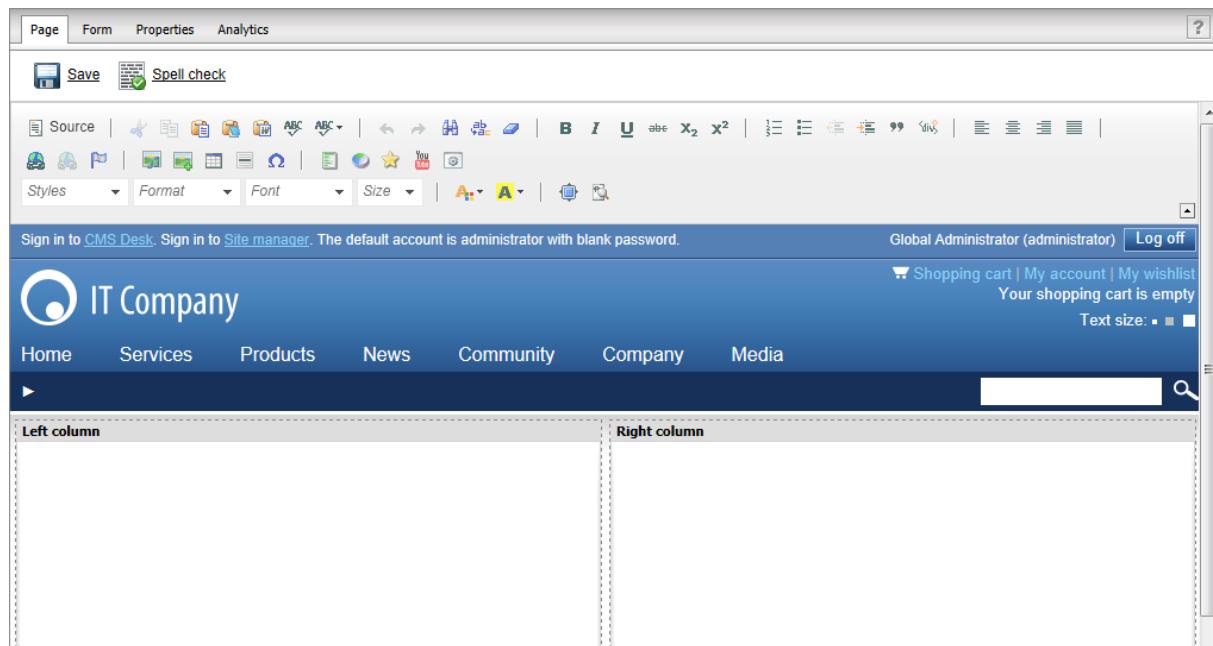
Creating an About Us page based on the new page template

10. Go to Kentico **CMS Desk -> Content**. Select **Corporate Site** (the root of the content tree) and click **New** in the main menu of the **Content** section. Choose to create a new **Page (menu item)**. Enter *About Us* into the page name field and choose to create a page using the **Corporate Site/Examples/Two column template** page template.



Click **Save** to create the new page.

11. Open the **Page** tab and you will see a page with editable regions like this:



Congratulations, you have just created your first page based on an ASPX page template. Now you can

enter some text and click  **Save** to store the content.



Please note

If you want to move the About Us page to another position in the menu, you can either use the **Up** and **Down** arrows on the main toolbar to change the order of the menu items or drag-and-drop the page to the desired location in the content tree.

5.3 Using master pages

Kentico CMS allows you to use standard ASP.NET master pages together with ASPX page templates. This is a very powerful concept that allows you to share the same site header and footer with a logo, main menu, search box, etc. over all pages without having to create these sections on each page template again and again.

Master pages are defined in files with the **.master** extension. You can assign a single master page to each ASPX page. The master page must always contain the **ContentPlaceHolder** control as shown here:

```
<asp:ContentPlaceHolder ID="plcMain" runat="server"></asp:ContentPlaceHolder>
```

The **ContentPlaceHolder** control specifies where the content of page templates that use this master page should be loaded. So the master page typically contains the main logo and navigation elements and the content is displayed by ASPX pages loaded into the master page.

The following code sample defines a very simple master page:

Please note: If you installed the Kentico CMS project as a web application, you need to rename the *CodeFile* attribute on the first line to *Codebehind* for the code example to be functional. Also, the attribute's value must be set to the name of the master page's code behind file and the *Inherits* attribute must be set according to the location and name of the master page.

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="Custom.master.cs"
Inherits="CMSTemplates_CorporateSite_Custom" %>

<%=DocType%>

<html xmlns="http://www.w3.org/1999/xhtml" <%=XmlNamespace%>>
<head id="Head1" runat="server">
    <title id="Title1" runat="server">My site</title>
    <asp:literal runat="server" id="l1lTags" enableviewstate="false" />
</head>
<body class="<%=BodyClass%>" <%=BodyParameters%>>

    <form id="form1" runat="server">
```

```
<cms:CMSPortalManager ID="CMSPortalManager1" runat="server"
EnableViewState="false" />
<ajaxToolkit:ToolkitScriptManager ID="manScript" runat="server"
EnableViewState="false"
ScriptMode="Release" />
<cms:CMSMenu ID="cmsmenul" runat="server" Cursor="Pointer"
HighlightAllItemsInPath="true"
Layout="Horizontal"
Padding="0"
Spacing="1" />
<asp:ContentPlaceHolder ID="plcMain" runat="server">
</asp:ContentPlaceHolder>

</form>
</body>

</html>
```

The **CMSPortalManager** control ensures the loading and saving of content between the database and editable regions. It also provides the management necessary for portal engine zones if any are defined on child ASPX pages.

The **CMSMenu** control generates a drop-down menu used for navigation. The **ContentPlaceHolder** control defines the area where the content of sub-pages should be loaded.

If you are planning to use AJAX components on the pages of your site, you need to add the **ToolkitScriptManager** control in addition to the CMSPortalManager control.

```
<ajaxToolkit:ToolkitScriptManager ID="manScript" runat="server" EnableViewState
=false" />
```

The code behind file also needs to be modified. Open it and add the following reference:

[C#]

```
using CMS.UIControls;
```

[VB.NET]

```
Imports CMS.UIControls
```

The master page must be inherited from the **TemplateMasterPage** class, so the class definition must look like this (the name of the class may be different):

[C#]

```
public partial class CMSTemplates_CorporateSite_Custom : TemplateMasterPage
```

[VB.NET]

```
Partial Class CMSTemplates_CorporateSite_Custom  
    Inherits TemplateMasterPage
```

And you also need to add the following code to the master page code behind class:

[C#]

```
protected override void OnPreRender(EventArgs e)  
{  
    base.OnPreRender(e);  
  
    this.ltlTags.Text = this.HeaderTags;  
}
```

[VB.NET]

```
Protected Overloads Overrides Sub OnPreRender(ByVal e As EventArgs)  
    MyBase.OnPreRender(e)  
  
    Me.ltlTags.Text = Me.HeaderTags  
End Sub
```

It is recommended to store master pages in the **CMSTemplates** folder together with page templates, so that they are exported with your website.

5.4 Adding portal engine functionality to ASPX templates

When developing or maintaining a website using ASPX page templates, one of the drawbacks is that the code of a page must be modified manually every time you wish to change its design. It is possible to add flexibility to ASPX templates by defining areas that may be edited directly through the browser in the Kentico CMS administration interface. The techniques used to manage these areas are the same as those used to design portal engine page templates. To learn more about portal engine features, please read the version of this tutorial dedicated to the portal engine.

Example

In this example, you will learn how to create a simple ASPX page template with areas that can be designed via the portal engine.

1. Open the web project in Visual Studio again (using either the **WebProject.sln** file or **File -> Open -> Web site** in the menu), right-click the **CMSTemplates -> CorporateSite** folder in the Solution Explorer and select **Add new item**.

2. Choose to create a new **Web form** and call it *TwoZones.aspx*, check the **Select master page** box and click **Add**. In the master page selection dialog, choose the **Root.master** page from the **CMS Templates/CorporateSite** folder and click **OK**.

3. Open the **Source** view of the newly created ASPX page and place the following code inside the **<asp:Content>** control:

```
<cms:CMSPagePlaceholder ID="plcZones" runat="server">
    <LayoutTemplate>

        <table width="100%" cellspacing="0" cellpadding="0">
            <tr valign="top">
                <td width="50%">
                    <cms:CMSWebPartZone ID="zoneLeft" runat="server" />
                </td>
                <td width="50%">
                    <cms:CMSWebPartZone ID="zoneRight" runat="server" />
                </td>
            </tr>
        </table>

    </LayoutTemplate>
</cms:CMSPagePlaceholder>
```

The **CMSPagePlaceholder** control creates an area on the page that will behave in a way similar to a portal engine page template.

The **<LayoutTemplate>** element is used to define the graphical layout of the area. This example uses a basic two column table structure, but setting a CSS-based layout applied through HTML elements (e.g. **<div>**, ****, etc.) is also a valid option.

The table contains two **CMSWebPartZone** controls, which represent fully functional portal engine zones. These zones may be managed when editing a page based on the page template on the **Edit -> Design** tab of **CMS Desk**. When some content is defined for a zone, information about it is stored in the database along with the respective page template object, not in the actual code of the ASPX page. Communication with the database is ensured by the **CMSPortalManager** control, which is located on the **Root.master** page.

4. Now switch to the code behind file and add the following namespace:

[C#]

```
using CMS.UIControls;
```

5. Next, modify the class definition to inherit from the **TemplatePage** class as shown below:

[C#]

```
public partial class CMSTemplates_CorporateSite_TwoZones : TemplatePage
```

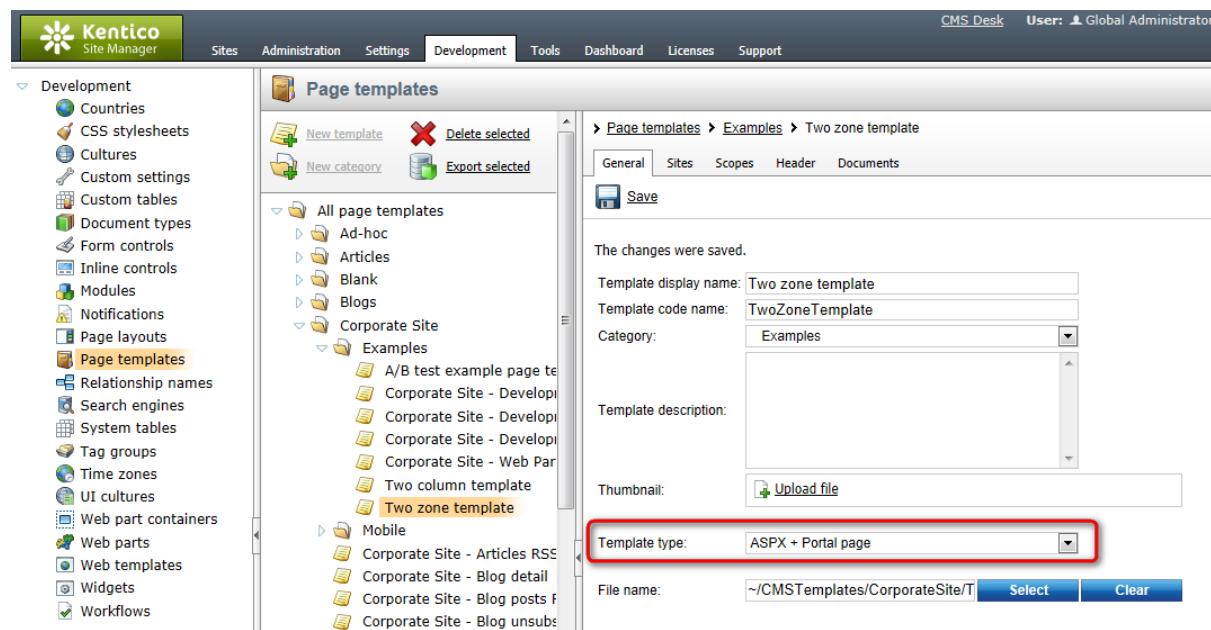
Save the changes made to both the ASPX page and its code behind file. Now the page can be used as a page template in Kentico CMS.

6. Sign in to **Site Manager**, go to **Development > Page templates** and register the created page as a template. To do this, select the **Corporate Site -> Examples** folder and click **New template**. Enter the following values:

- **Template display name:** Two zone template
- **Template code name:** TwoZoneTemplate

Click **OK**.

7. Next, select the *ASPX + Portal page* option for the **Template type** property. This is necessary in order for the **Design** tab of **CMS Desk** to be available when editing pages using the template. Enter the following path into the **File name** field: `~/CMSTemplates/CorporateSite/TwoZones.aspx`



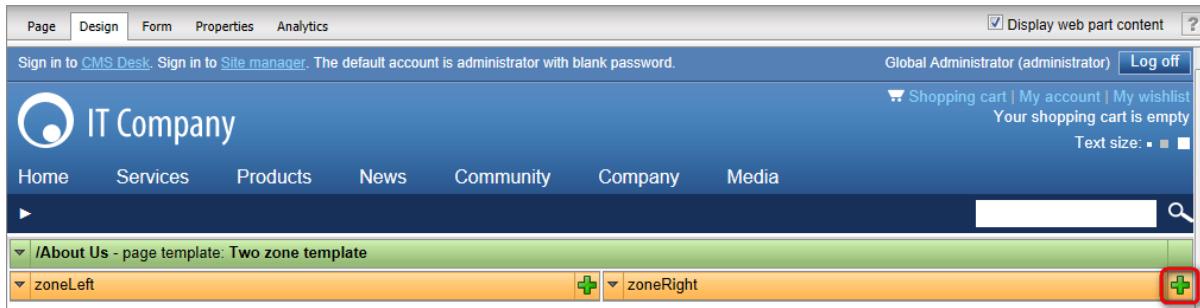
Save the changes.

8. Switch to the **Sites** tab and use the **Add sites** button to assign the page template to the site that you are using (Corporate Site).

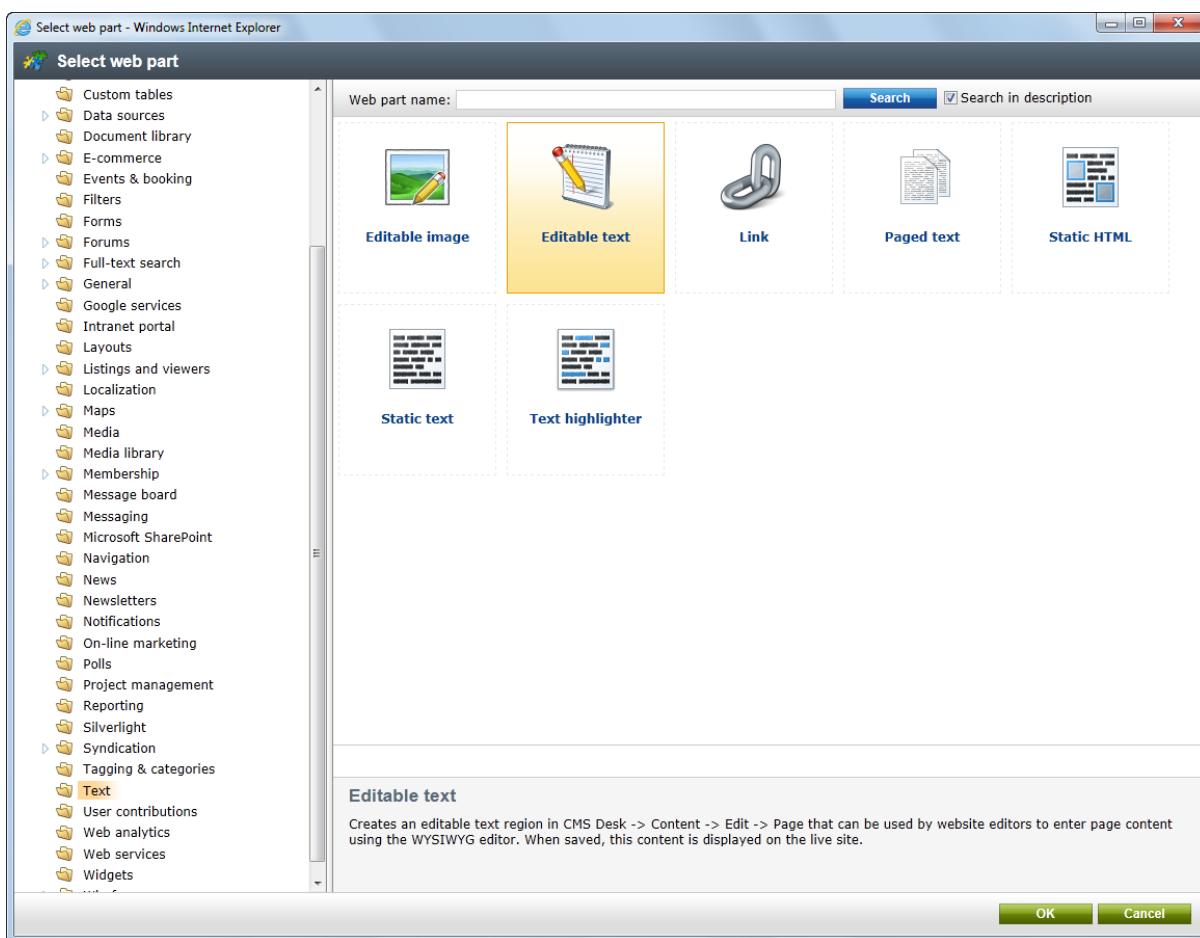
9. Now go to **CMS Desk -> Content**. We will modify the **About Us** page created in the previous example to use our new template. Select the page and switch to its **Properties -> Template** tab. Click the **Select** button and choose the **Corporate Site/Examples/Two zone template** page template from the catalog. Click **Save** to confirm the change of page template.

10. Refresh your browser and switch to the **Design** tab, which is now available for the **About Us** page. You will see two empty zones on the page as defined in the ASPX code of the template. The content of standard zones are components called web parts. Add a web part to the **zoneRight** zone by clicking

the **Add web part** (+) icon in the top right corner.

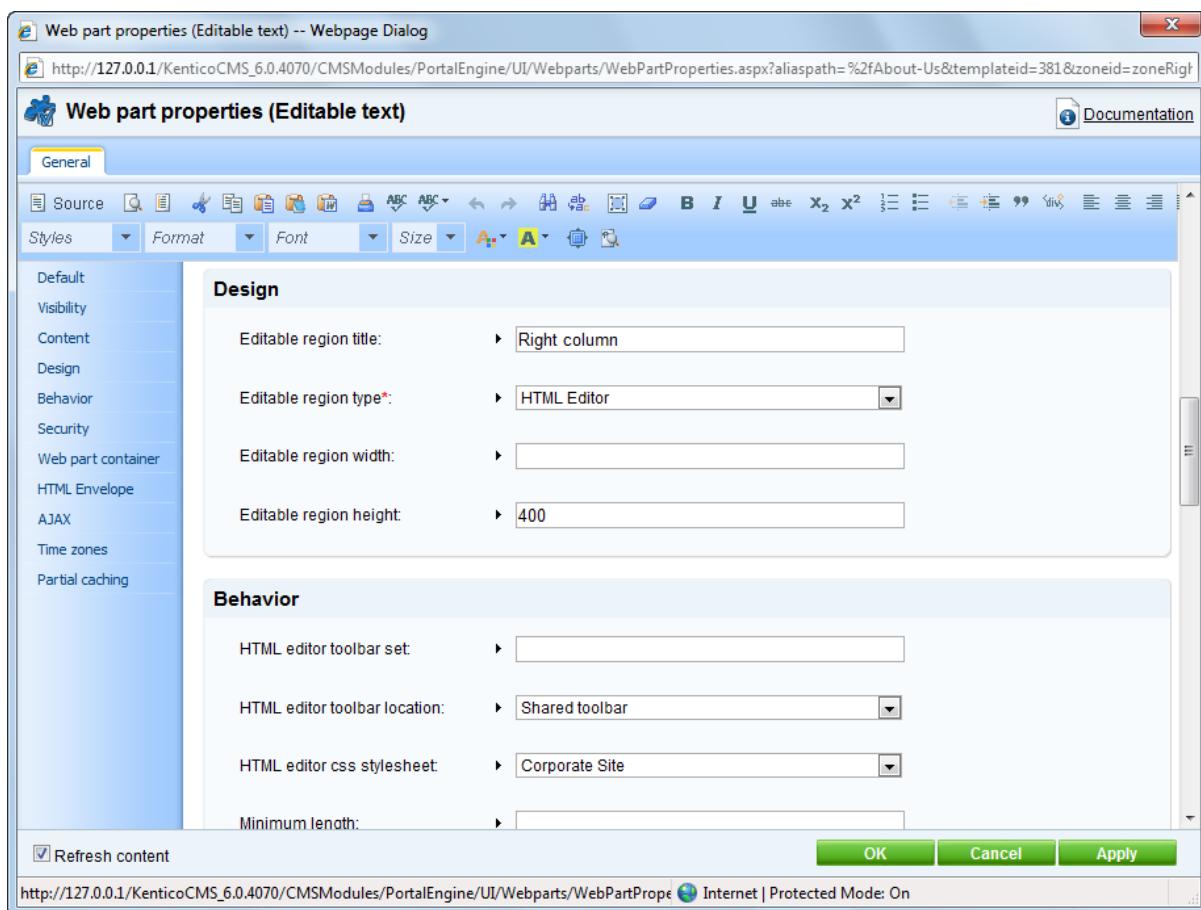


The **Select web part** dialog will be opened. Choose the **Text -> Editable text** web part and click **OK**.



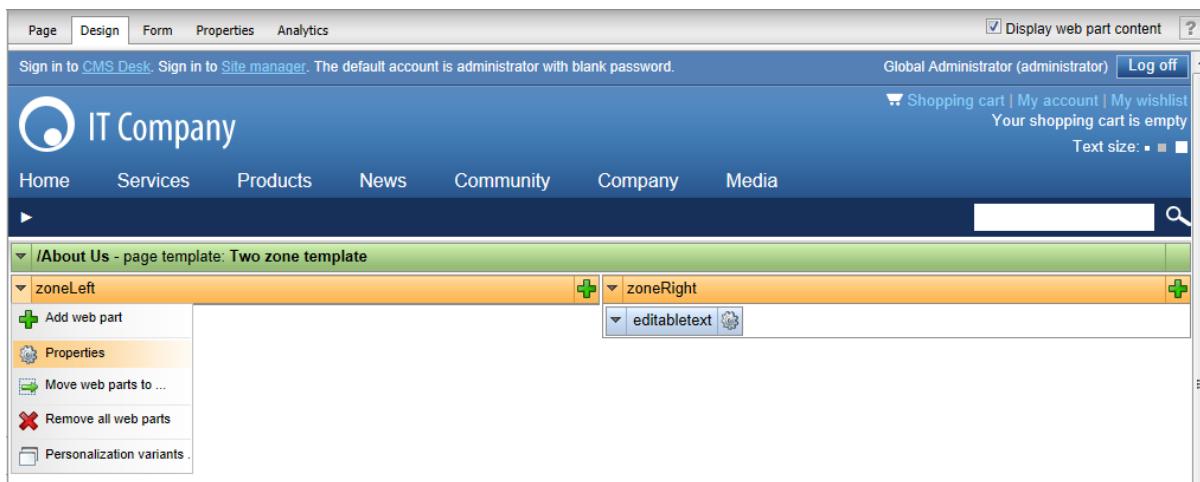
The **Web part properties** dialog will be opened and the web part can be configured. Enter the following values:

- **Design -> Editable region title:** Right column
- **Design -> Editable region height:** 400

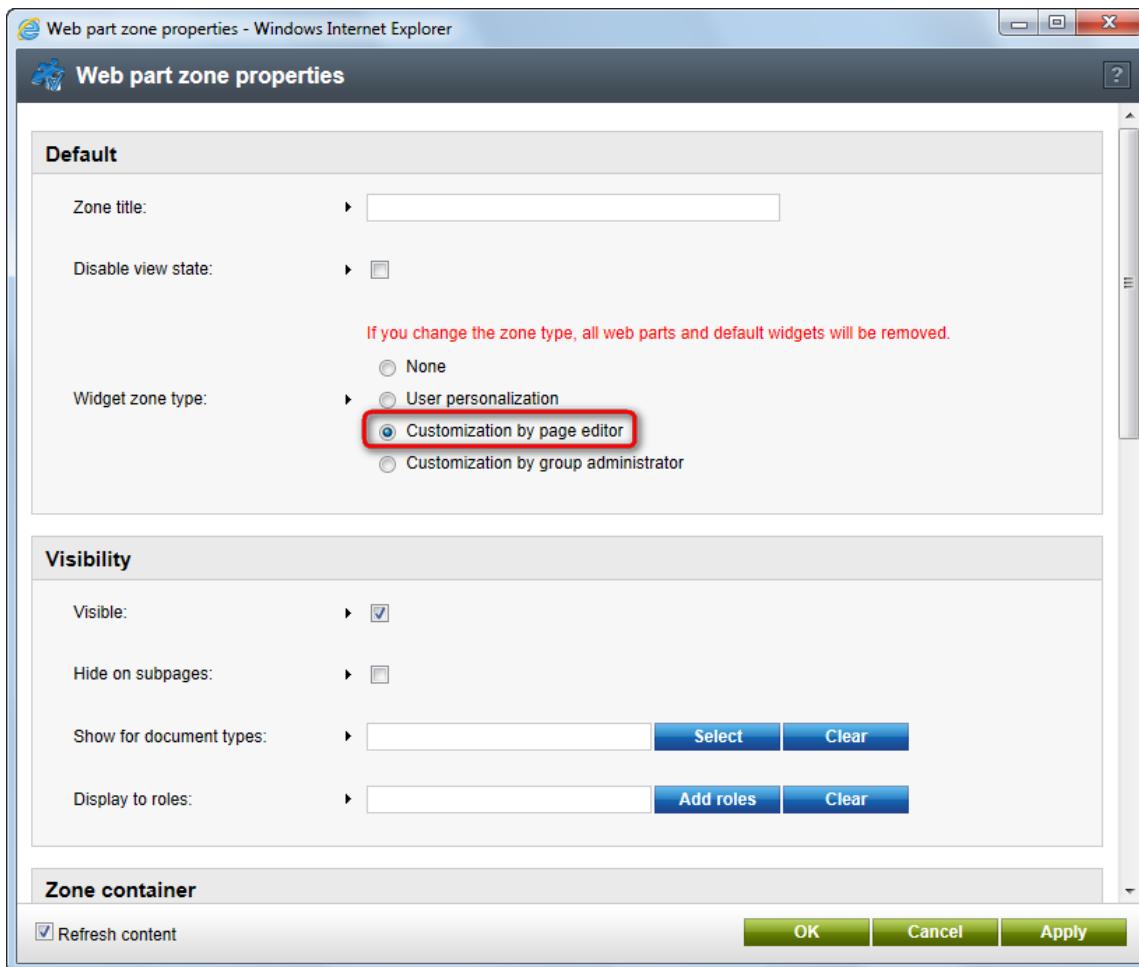


Click **OK** to save the web part and add it to the zone. This web part provides a text area on the page that can be edited on the **Page** tab of **CMS Desk**, just like the editable regions in the previous example. As you can see, the design of the page can be built using a simple browser-based interface. A web part zone may contain multiple web parts.

11. Zones may also be configured to use various types of widgets, which are objects similar to web parts, but they allow page customization by different kinds of website users, not just the administrators or designers. Expand the menu (▼) of the **zoneLeft** zone and select **Properties** to configure the zone.



Change the **Widget zone type** property from *None* to *Customization by page editor*.



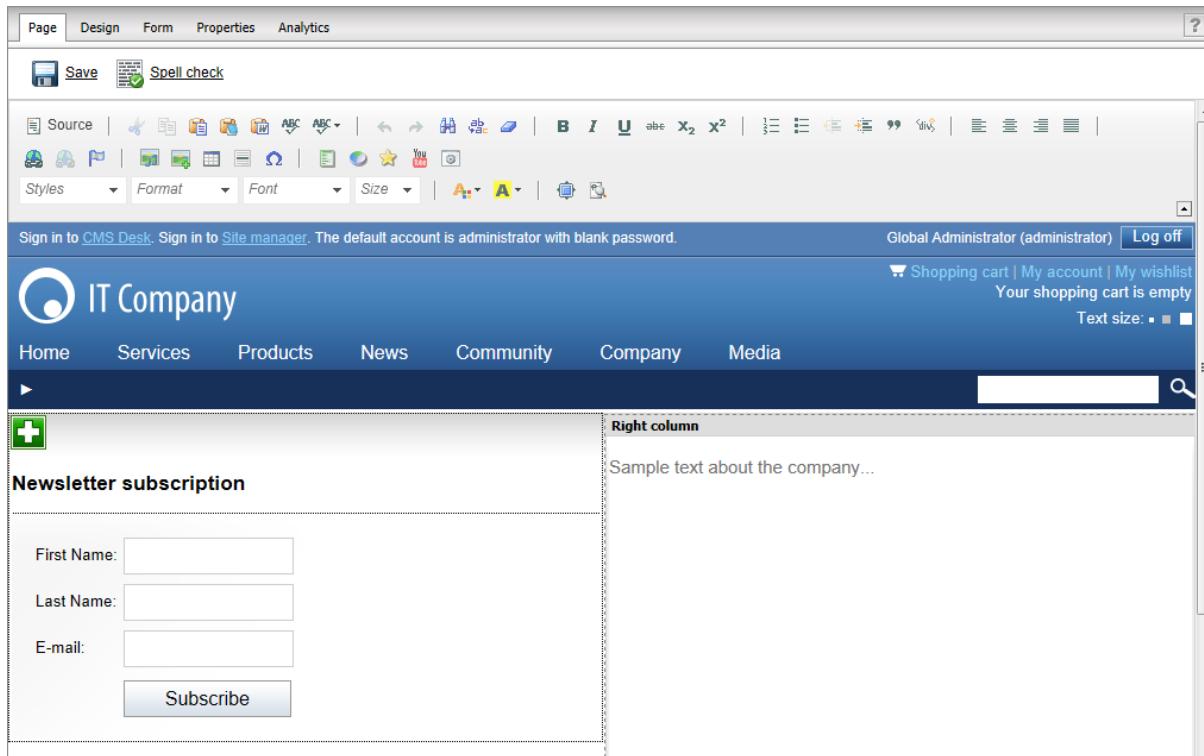
Click **OK**. This zone will now serve as a widget zone for page editors.

12. Switch to the **Page** tab, enter some content into the editable text region displayed by the web part on the right and click **Save**. You will also be able to manage the editor widget zone on the left. Click

the **Add widget** (+) button in the top left corner of the widget zone to place a widget onto the page. Select the **Newspapers > Newsletter subscription** widget from the catalog and set the following values for its properties:

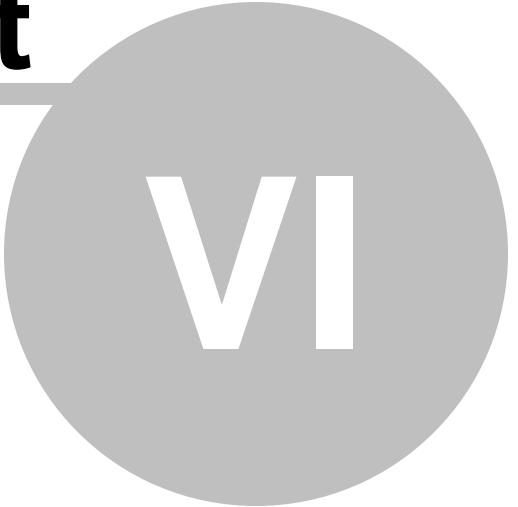
- **Newsletter name:** Corporate Newsletter
- **Allow user subscribers:** disabled (unchecked)
- **Widget container:** Corporate site - Light gradient box
- **Widget container title:** Newsletter subscription

Click **OK** to add the widget to the page. It provides a form which users may use to subscribe to the site's newsletter.



As shown in this example, it is possible to use web parts or widgets to build the design of pages based on ASPX page templates. This approach combines the standard architecture and development process of ASPX templates with the flexibility and user-friendliness of the portal engine.

Part



VI

Managing styles and design

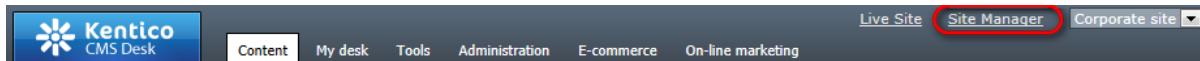
6 Managing styles and design

6.1 CSS styles

The design of the website relies on standard CSS styles. Each website has its global CSS stylesheet that can be chosen in **Site Manager -> Sites -> ... edit site ... -> General**. Here you can also choose a different stylesheet for the site's WYSIWYG editors.

In addition, each page can override the global website stylesheet. You can assign a stylesheet to a specific page by editing its corresponding document in **CMS Desk -> Content -> Properties -> General** and using the **CSS stylesheet** selector.

The actual content of the CSS stylesheets can be managed in Site Manager. When you are in CMS Desk, you can easily switch to Site Manager by clicking the **Site Manager** link in the header:



Then click **Development** in the **Site Manager** main menu and select **CSS stylesheets** from the left menu:

A screenshot of the Kentico Site Manager interface. The top navigation bar includes the Kentico logo, Site Manager, and links for Sites, Administration, Settings, Development (which is selected and highlighted with a grey background), Tools, Dashboard, Licenses, Support, and Buy. The left sidebar under the Development menu is expanded, showing options like Countries, CSS stylesheets (which is highlighted with an orange background), Cultures, Custom settings, Custom tables, Document types, Form controls, and Inline controls. The main content area is titled "CSS stylesheets" and contains a "New CSS stylesheet" button. Below that is a table with three rows, each representing a CSS style sheet:

Actions	Display name	Code name
	Corporate Site	CorporateSite
	Corporate Site - Mobile device	CorporateSiteMobile
	Corporate site printer styles	CorporateSitePrinter

Edit () the **Corporate Site** stylesheet:

Check out the stylesheet to the c:\inetpub\wwwroot\KenticoCMS4248.10360\CMSCSS\Stylesheets\CorporateSite.css file to edit it externally.

Stylesheet display name: Corporate Site
Stylesheet code name: CorporateSite

```

/*#Global*/
body, form
{
    background-color: #ffffff;
    font-size: small;
    font-family: Arial;
    margin: 0px;
    padding: 0px;
}

h1
{
    font-size: 28px;
    margin-top: 0px;
    padding-top: 35px;
}

h2
{
    font-size: 140%;
    margin-top: 0px;
    padding-top: 35px;
}

```

Stylesheet text:

BizForms
Blogs
Boxes
Content box
Light gradient box
List box head container
Buttons
Captcha controls
Careers
Community section
Wiki
Company
Content zone
2 columns
3 columns
Home page
Context menu
Document library
General styles
Library
Override DesignMode styles
Popup control
Uploaders

Please note: you may need to refresh your browser in order to see the latest version of the style.



Browser and language-specific styles

Pages automatically have CSS classes assigned to their `<body>` element according to the characteristics of their language (its text direction and specific culture) and depending on the browser in which they are currently viewed. For example:

```
<body class="LTR IE IE9 ENUS">
```

As you can see, four types of classes are added:

- **Text direction** - the `LTR` class is assigned for left-to-right languages and `RTL` for right-to-left.
- **Browser type** - this class is added according to the browser in which the page is currently opened. The following classes are used:

Browser	Class name(s)
Internet Explorer	IE
Firefox	Gecko
Google Chrome	Chrome, Safari
Safari	Safari
Opera	Opera

- **Browser version** - the class name is the same as for the browser type, but with the number of the browser's major version appended, e.g. *IE9*, *Gecko5* etc.
- **Culture** - the name of the class is added based on the culture code of the page's content (without the hyphen), for example *ENUS* for pages using the *en-US* culture.

This feature allows you to style page elements differently according to the browsing environment of the current visitor. You can define styles for any combination of the classes mentioned above.

For example, you can add the following into a website's stylesheet:

[CSS]

```
.IE8 .MyFont
{
    font-size: 20px;
}

.Opera .MyFont
{
    font-size: 18px;
}
```

Now elements styled using the *MyFont* class will have a different font size when viewed in the Internet Explorer 8 or Opera (all versions) browsers.

6.2 App themes

In some cases, you may leverage the built-in support for [ASP.NET themes](#). You can use them to set styles for controls that do not have their own CSS class name, such as Datagrid, Calendar or web parts with complex dialogs (logon form, registration form, ...).

Themes must be defined in a folder located under the **App_Themes** directory. The name of this folder needs to be the same as the code name of the site's CSS stylesheet. So if you use the *Green* stylesheet on your site, your themes must be stored in the *App_Themes\Green* sub-folder.

Skins for your controls must be added to the **Default.skin** file under this folder. Here's an example of a skin for the **CMSCalendar** control / **Calendar** web part:

```
<cms:CMSCalendar Runat="server">
    <NextPrevStyle ForeColor="Red"></NextPrevStyle>
    <WeekendDayStyle BackColor="#E0E0E0"></WeekendDayStyle>
</cms:CMSCalendar>
```

The code above defines the appearance of the calendar control. You can see this control on the Events page of the sample Corporate Site.



Where should I store website design files?

It's recommended to store all images, flash movies and other resources that are part of the website design template in the **App_Themes/<stylesheet code name>** folder. This ensures that the files are exported together with the stylesheet if you deploy it to some other server.

6.3 Menu design

Now you will learn how to change the design of the main menu. The main menu used on the sample Corporate Site is displayed using the **Drop-down menu** web part which is based on the **CMSMenu** server control.

The menu design depends primarily on the applied CSS styles. Here's an example of the CSS styles for the drop-down menu:

```
.zoneMenu .CMSListMenuUL
{
    list-style: none;
    margin: 0px;
    padding: 0px;
    position: relative;
}

.zoneMenu .CMSListMenuUL li
{
    float: left;
    padding: 0px 22px 0px 0px;
}

.zoneMenu .CMSListMenuUL li a
{
    color: #fff;
    text-decoration: none;
    display: block;
    height: 23px;
    font-size: 16px;
    line-height: 23px;
    padding: 0px 8px;
    border: 1px solid transparent;
    font-family: Arial;
}

.zoneMenu .CMSListMenuUL .CMSListMenuHighlightedLIfirst a,
.zoneMenu .CMSListMenuUL .CMSListMenuLIfirst a
{
    padding-left: 0px;
}

...
```

As you can see, these are standard CSS styles. You can modify the styles in the global CSS stylesheet of the given site.

The default menu looks like this:



Now we will change the background color of selected menu items to orange. Go to **Site Manager -> Development -> CSS stylesheets** and edit (edit icon) the **Corporate Site** stylesheet. Select **Top menu styles -> Horizontal** in the right navigation panel. Change the highlighted line:

```
.zoneMenu .CMSListMenuHighlightedLI a,  
.zoneMenu .CMSListMenuHighlightedLIfirst a  
{  
    color: #8cd2f8 !important;  
    background-color: orange;  
    text-decoration: none;  
}
```

Click **OK** to save the changes. When go to the live site now, you will see a menu like this:



Defining the style of a single menu item

Every document may have its own style that is used when the document is displayed in the menu. We will try to modify the style of the **Home** menu item. Go to **CMS Desk -> Content** and select **Home** from the content tree. Click **Properties -> Menu**. Here you can define:

- **Menu caption** - the name of the document when it's displayed in a menu.
- **Show in navigation** - indicates if the document should be displayed by navigation controls.
- **Show in sitemap** - indicates if the document should be displayed in the sitemap.
- **Menu actions** - sets special behaviour for when the menu item is clicked by a user. The link can be disabled, a JavaScript command can be executed, or a specified URL can be opened.
- **Menu design** - sets styles applied to the document's menu item (standard, mouse-overed and highlighted).

Enter the following value into the **Menu item style** value (in the Menu item design section): *background-color: red;* and click **Save**. Click **Live site**. Click **Services**. The **Home** menu item is now displayed in red:



Part



VII

Creating a new site using ASPX templates

7 Creating a new site using ASPX templates

7.1 Overview

This tutorial will guide you through the creation of a simple website using ASPX page templates developed in Visual Studio. You will learn how to define site structure, design, how to create your own pages and page templates.

During this tutorial, we will use a static website template that is similar to what a developer gets from a graphic designer. It looks like this:

The screenshot shows a static website template for "Our Business, Inc.". The header is orange with the company name. Below it is a blue navigation bar with the text "Home Page 1". The main content area features a large image of a woman with glasses and a white shirt, sitting at a desk with a laptop. To her right is a text box with the heading "Welcome to Our Business, Inc. Web Site" and a brief description. Below that is another text box with the heading "How to convert a web site template into a master page" and a list of steps. To the right of the main content is a sidebar with the heading "Contact us" and the company's address and phone/fax numbers. At the bottom of the page is a footer note: "This is a sample web site for Kentico CMS".

You can find the static page template in the **C:\Program Files\Kentico CMS\<version>\CodeSamples\SampleWebTemplate** folder. The template consists of the `home.htm` file, a `styles` folder and an `app_themes` folder with images.

7.2 Creating a new web site using the wizard

The following topics assume that you have previously installed a sample Corporate Site. We will leave the existing website and add a new website that will run on `http://127.0.0.1`.

Multiple sites and Visual Studio's built-in web server

If you are using the **built-in web server in Visual Studio instead of IIS**, you need to **stop the CorporateSite** site in the Site Manager -> Sites dialog first and then you can

continue. Since the built-in web server doesn't support any other domain than localhost, you will use the **localhost** domain again.

Sign in as Administrator to **Site Manager -> Sites** and click **New site wizard**.

The screenshot shows the Kentico Site Manager interface. At the top, there's a navigation bar with links for CMS Desk, User (Global Administrator), v6.0.4281, Sign Out, and a help icon. Below the navigation is a toolbar with icons for Sites, Administration, Settings, Development, Tools, Dashboard, Licenses, and Support. The main area is titled 'Sites' and contains a sub-menu with 'New site wizard', 'Import site or objects', and 'Export objects'. Below this is a table listing sites. The first row, 'Corporate site', has columns for Actions (with edit and delete icons), Site name (Corporate site), Status (Running), Main domain (localhost), Documents (524), and Default culture (English - United States). A dropdown menu for 'Items per page' is set to 25. The entire interface has a dark blue header and a light gray body.

In the first step, choose to **Use website template**. Click **Next**.

The screenshot shows the 'Step 1' wizard screen. On the left, it says 'Step 1'. To its right, under 'Choose default website', there's a note: 'If you choose to create a blank site, the wizard will guide you through the process of creating a new site. If you choose to use a template, you will be able to choose one of the predefined website templates.' Below this note are two radio buttons: 'Create a new site using a wizard' (unchecked) and 'Use website template' (checked). At the bottom right of the screen is a green 'Next >' button.

In the second step, choose the **Blank site ASPX** website template. Click **Next**.

The screenshot shows the 'Step 2' screen for choosing a website template. At the top, there's a header with the title 'Choose website template' and a sub-instruction: 'Choose the predefined website template that will be used for your new website. The website template may contain site structure, design, basic content, new document types and other settings.' Below this, there are three template options:

- Intranet Portal**: A screenshot of the portal interface is shown, followed by a description: 'This is a ready-to-use website template for a sample intranet portal site. The site showcases many features useful for a company intranet, such as sub-sections dedicated to departments or workgroups and document or project management. You may use it as a base for your own company intranet by replacing the sample data with your own. It uses the portal engine and may be used by developers who are new to Kentico CMS.'
- Blank Site**: A screenshot of the blank site interface is shown, followed by a description: 'This template is intended for developers who want to create a new web site from scratch. It uses the portal engine.'
- ASPX** **Blank Site ASPX**: A screenshot of the ASPX interface is shown, followed by a description: 'This template is intended for developers who want to create a new web site from scratch. It uses the ASPX page templates that require a higher level of .NET knowledge than the portal engine.'

At the bottom right, there are navigation buttons: '< Previous' and 'Next >'.

In the third step, enter the following details:

- **Site display name:** My website
- **Site code name:** mysite
- **Domain:** 127.0.0.1 (if you are using Visual Studio's built-in web server, set the **Domain** value to *localhost*)

Click **Next**.

Step 3

Enter new site settings

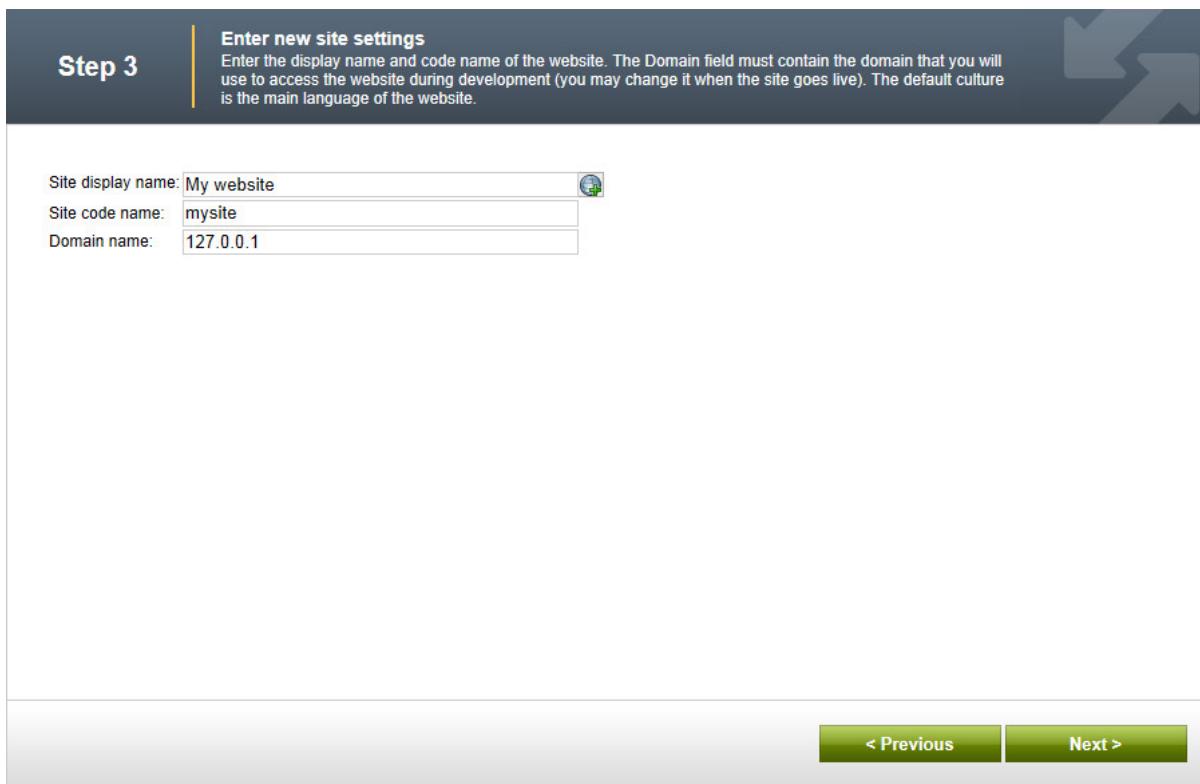
Enter the display name and code name of the website. The Domain field must contain the domain that you will use to access the website during development (you may change it when the site goes live). The default culture is the main language of the website.

Site display name: My website 

Site code name: mysite

Domain name: 127.0.0.1

< Previous | Next >



In the fourth step, you are asked to select objects which should be imported to your new site. Do not change anything and click **Next**.

Step 4

Objects selection

Please select objects which should be imported.

Import objects

Please note: The import process may overwrite your existing objects. The existing objects are marked with * and will be overwritten if checked.

Please select the object type from the tree if you wish to change the default selection. Click **Next** to start the import of selected objects.

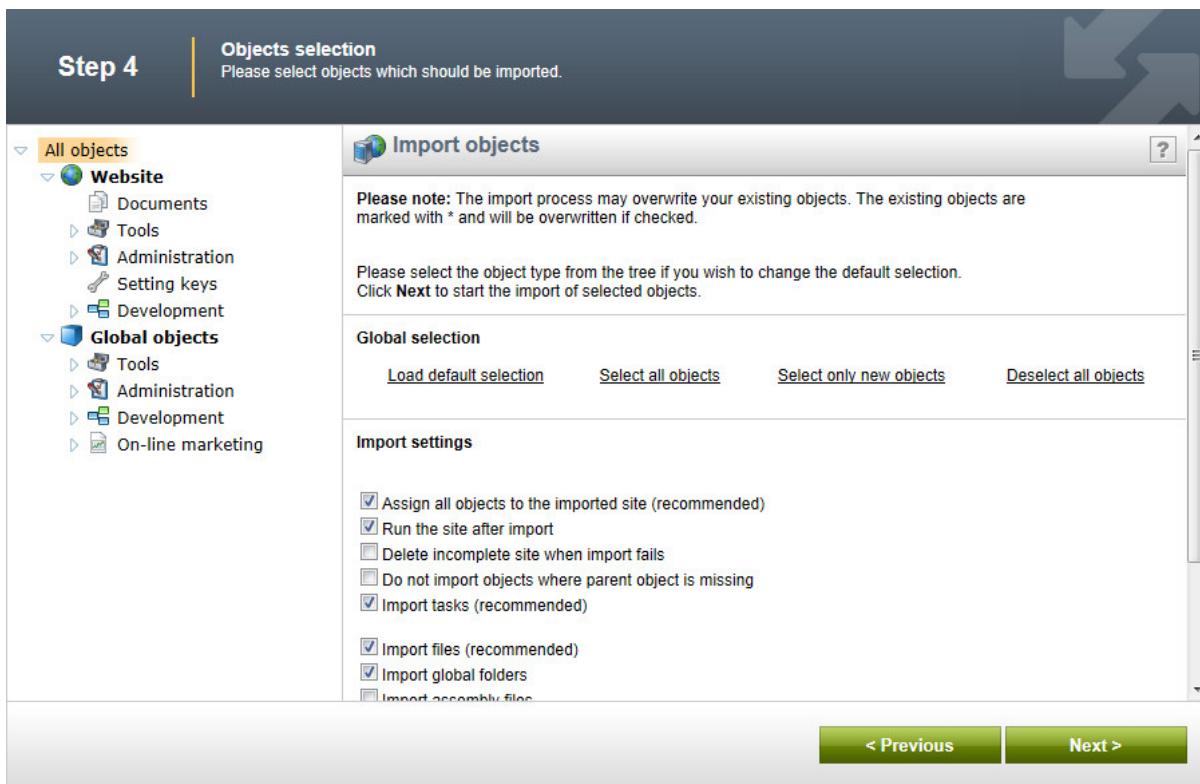
Global selection

[Load default selection](#) [Select all objects](#) [Select only new objects](#) [Deselect all objects](#)

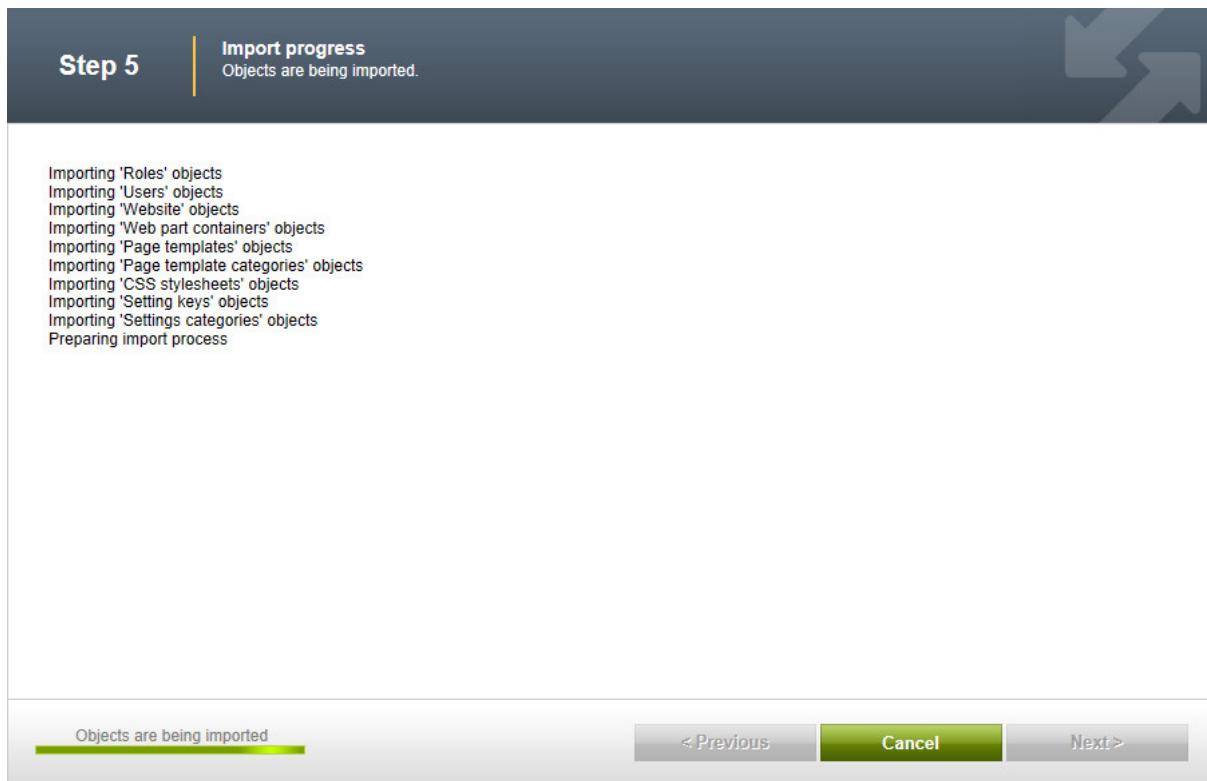
Import settings

Assign all objects to the imported site (recommended)
 Run the site after import
 Delete incomplete site when import fails
 Do not import objects where parent object is missing
 Import tasks (recommended)
 Import files (recommended)
 Import global folders
 Import assembly files

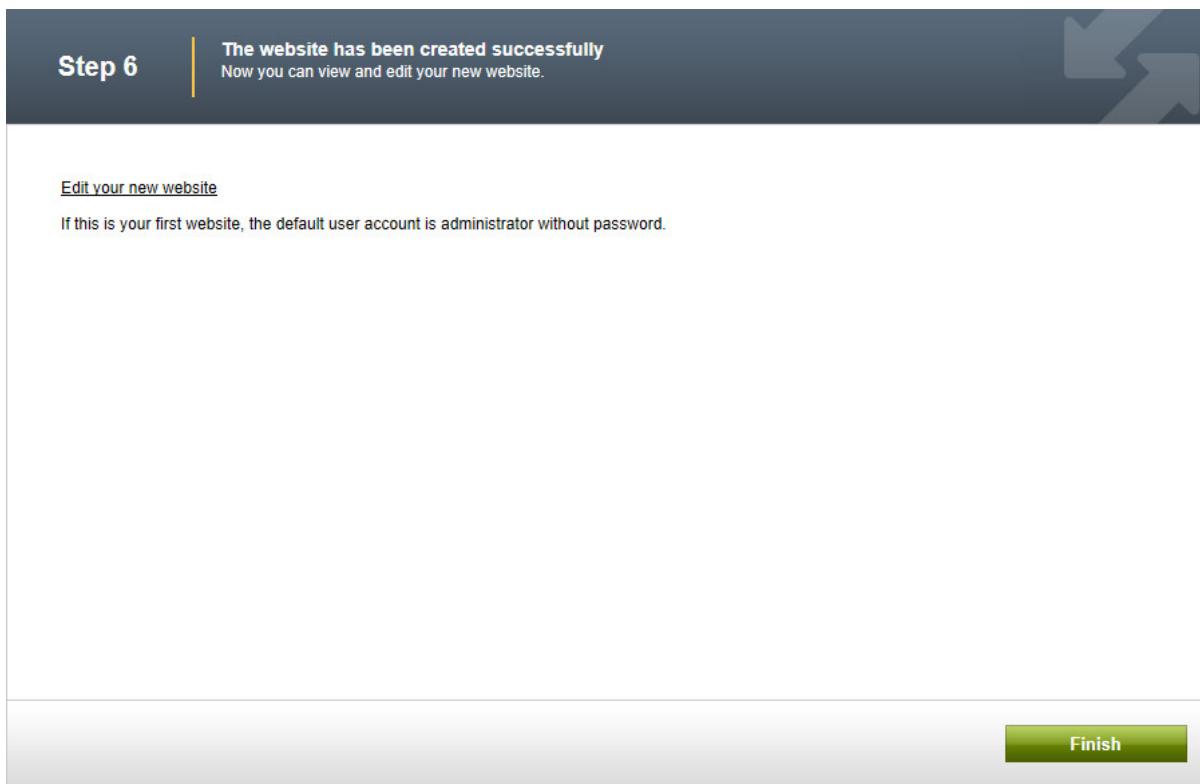
< Previous | Next >



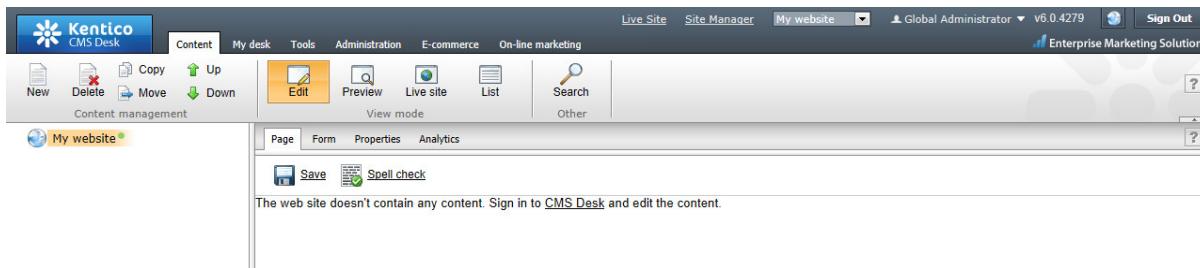
In the fifth step, the progress of the object import is displayed. Click **Next** after **Import has successfully finished** appears.



You will see the confirmation message.



Click the **Edit your new website** link. A new window with Kentico CMS Desk opens at domain 127.0.0.1. You need to sign in again since authentication is not shared over different domains by default. After you sign in, you will see your new, empty website:



You have created the base for your new website. In the next topics, you will learn how to implement the required design.

7.3 Creating the CSS stylesheet

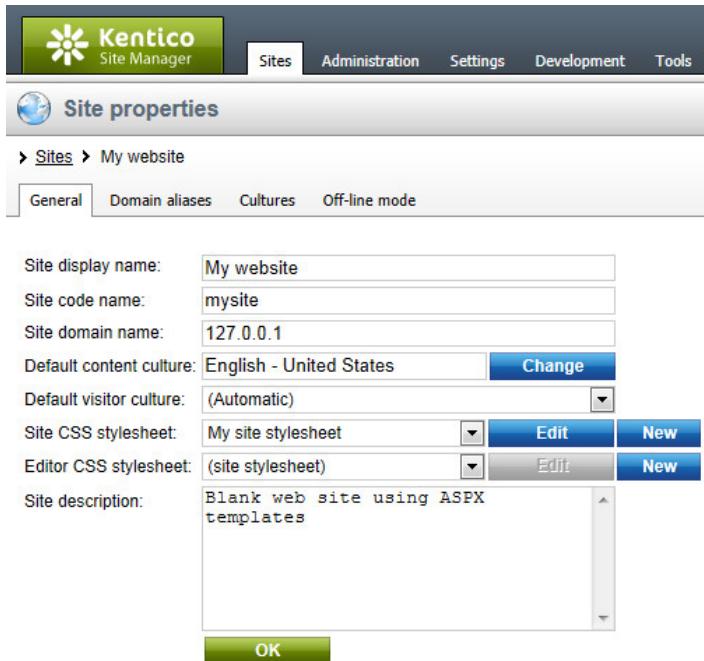
Before we start editing our new website, we will prepare the CSS styles and images based on our website template. Go to **Site Manager -> Development -> CSS Stylesheets** and click **New CSS stylesheet**. Enter the following values:

- **Stylesheet display name:** My site stylesheet
- **Stylesheet code name:** MySite
- **Stylesheet text:** copy and paste all text from the **SampleWebTemplate\Styles\main.css** file (you can find it in the **C:\Program Files\Kentico CMS<version>\CodeSamples** folder)

- assign to website **My website**: enabled

Click **OK**.

Go to **Site Manager -> Sites** and **Edit** (edit icon) the properties of **My website**. Select **My site stylesheet** in the **Site CSS stylesheet** drop-down list and click **OK**. This will ensure that the stylesheet is used on all pages of your new website.



Now copy the **SampleWebTemplate\app_themes\MySite** folder to **<web project>\app_themes**. The folder contains graphics for this website template. It will ensure that the images are exported as a part of the website if you decide to move the website in the future. Please note that the folder under **app_themes** must have the same name as the code name of the CSS stylesheet: **MySite**.



CSS stylesheet URL and relative paths

We have adjusted the image paths in the sample CSS stylesheet so that they match the target folders in your new website. In real-world scenarios, you will need to adjust the paths manually. **The URLs of images in the CSS stylesheets are always relative to the location of the web project.**

The URL of the CSS stylesheet is:

<web project>/CMSPages/GetResource.ashx?stylesheetname=MySite

which means, you need to link to files in the app_themes folder like in the example below:

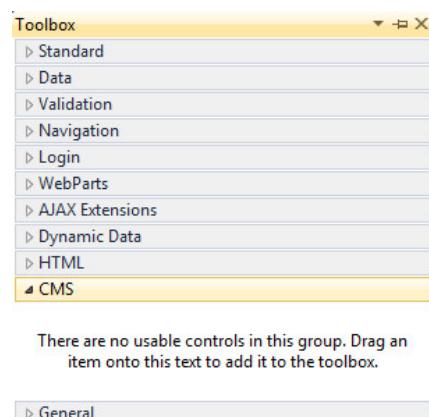
../app_themes/mysite/images/imagename.gif.

7.4 Opening and configuring the web project

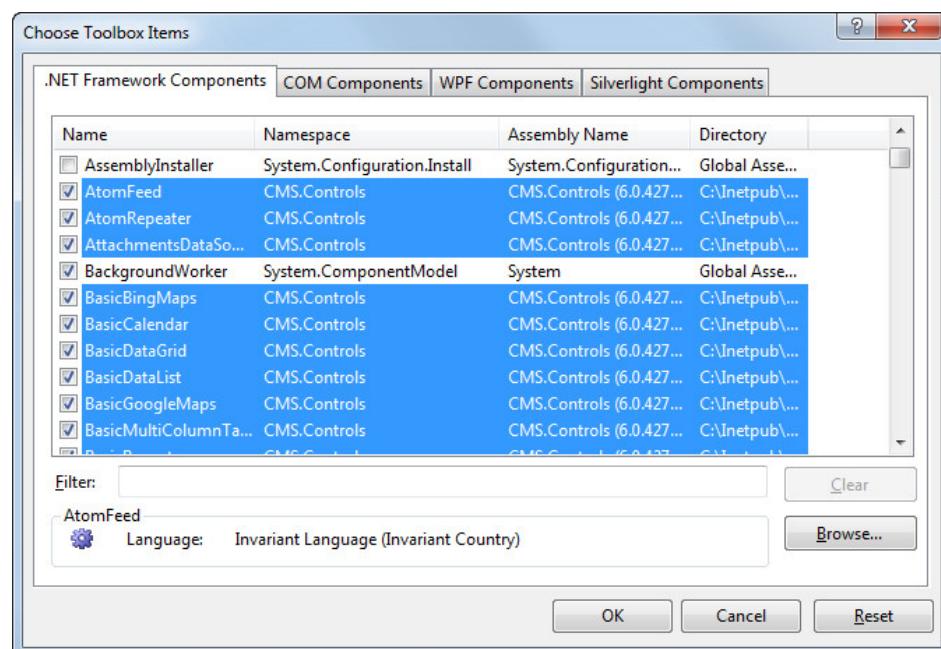
Open the web project in Visual Studio. You can open it either using the **WebProject.sln** file or using the **File -> Open -> website** menu.

Now we need to add Kentico CMS Controls to the Visual Studio Toolbox.

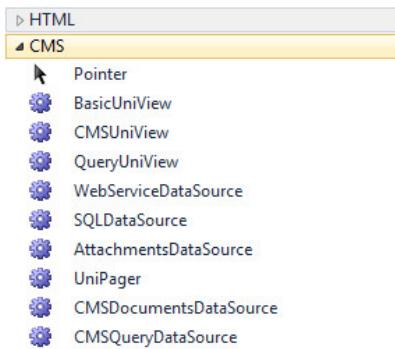
1. Open the website project in Visual Studio and open some ASPX page.
2. Right-click the **Toolbox** and choose **Add tab** from the context menu.
3. Type the name of the new tab (e.g. CMS) and press Enter:



4. Right-click the new tab and choose **Choose items...** from the context menu.
5. In the **Choose Toolbox Items** dialog, click **Browse** and locate the **CMS.Controls.dll** library in the **bin** folder under your website. Click **Open** and then click **OK**.



6. The controls are now added to the Toolbox:

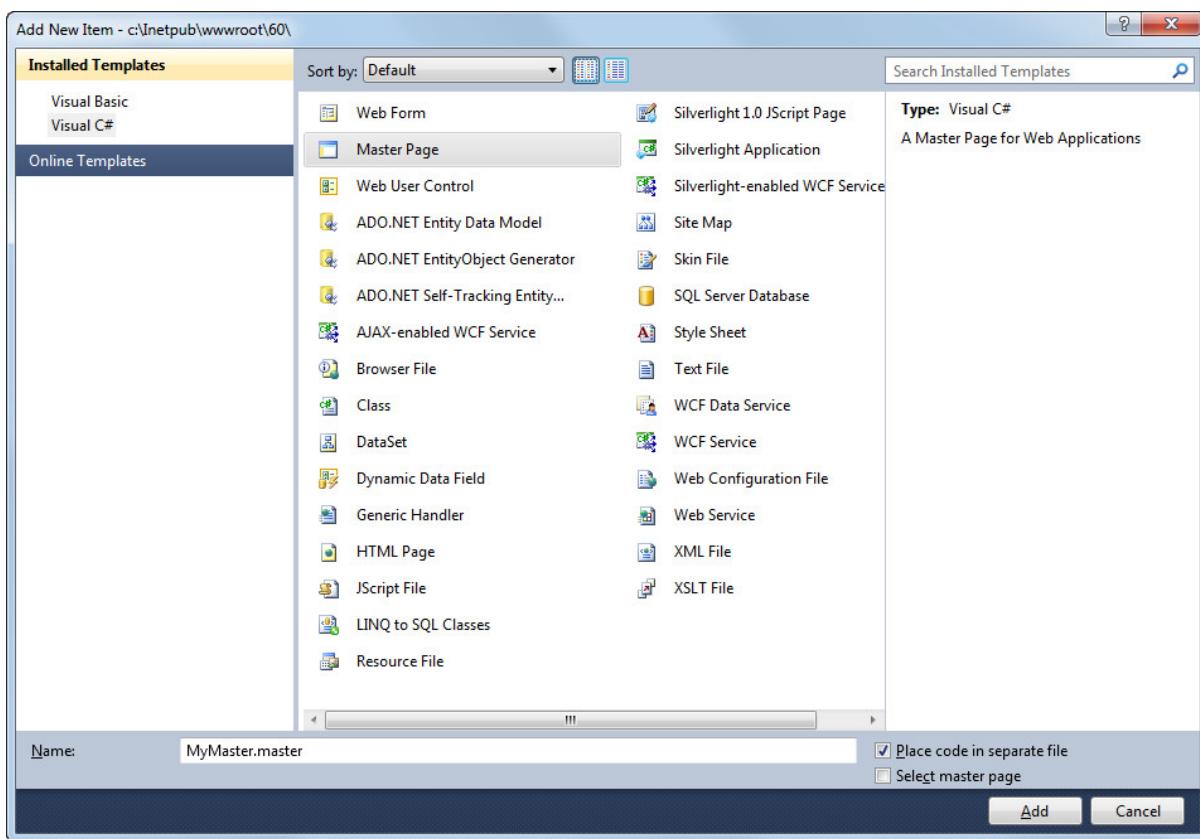


7. Now you can easily drag and drop the controls on your Web forms.

7.5 Master page

Open the web project in Visual Studio, right-click the **CMS**Templates folder in the Solution Explorer window and create a new sub-folder named *MySite*. Please note that the folder name must be the same as the code name of your site to ensure that the contents are exported/imported along with the website if it is deployed to another location.

Right-click the **MySite** folder, select **Add new item...** and create a new **Master page** called *MyMaster.master*. If you are a VB developer, you may want to choose Visual Basic in the **Language** drop-down list.



Replace all default ASPX code from the master page (in the Source view) except for the first line with the **<%@ Master %>** directive with the following code:

```
<%=DocType%>
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title id="Title1" runat="server">My website</title>
    <asp:literal runat="server" id="ltlTags" enableviewstate="false" />
</head>
<body class="<%$BodyClass%>" <%$BodyParameters%>>
    <form id="form1" runat="server">
        <cms:CMSPortalManager ID="CMSPortalManager1" runat="server" />
    </form>
</body>
</html>
```

The **CMSPortalManager** control ensures the loading and saving of content between the database and editable regions. It also provides the management necessary for web part or widget zones defined on child ASPX pages.

Add the **ToolkitScriptManager** control after the CMSPortalManager control to allow AJAX components to work on the pages of your site:

```
<ajaxToolkit:ToolkitScriptManager ID="manScript" runat="server" EnableViewState  
= "false" />
```

Switch to code behind and add a reference to the CMS.UIControls namespace:

[C#]

```
using CMS.UIControls;
```

[VB.NET]

```
Imports CMS.UIControls
```

Change the class definition so that the master page inherits from the **TemplateMasterPage** class:

[C#]

```
public partial class CMSTemplates_MySite_MyMaster : TemplateMasterPage
```

[VB.NET]

```
Partial Class CMSTemplates_MySite_MyMaster  
    Inherits TemplateMasterPage
```

Add the following code after the **Page_Load** method:

[C#]

```
protected override void OnPreRender(EventArgs e)  
{  
    base.OnPreRender(e);  
  
    this.ltlTags.Text = this.HeaderTags;  
}
```

[VB.NET]

```
Protected Overloads Overrides Sub OnPreRender(ByVal e As EventArgs)  
    MyBase.OnPreRender(e)  
  
    Me.ltlTags.Text = Me.HeaderTags  
End Sub
```

Now switch back to the **Source mode** (HTML mode) and copy and paste the HTML code from inside the **<body>...</body>** tags of the sample **home.htm** file onto the master page after the **<cms:PortalManager>** control.

However, we only need the logo, main menu and footer, so replace the entire code in the **<!-- main content --> ... <!-- /main content -->** section with the following control:

```
<asp:ContentPlaceHolder ID="plcMain" runat="server"></asp:ContentPlaceHolder>
```

This is a standard ASP.NET control that ensures the loading of pages into the master page.

So the final code of the **<body>** element of the master page will look like this:

```
<body class="<%$BodyClass%>" %$BodyParameters%>
    <form id="form1" runat="server">
        <cms:CMSPortalManager ID="CMSPortalManager1" runat="server" />

        <div class="MainDiv">
            <!-- logo -->
            <br />
            <div class="Logo">
                &nbsp;
            </div>

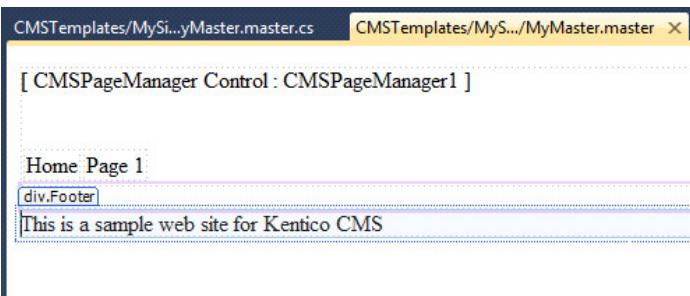
            <!-- main menu -->
            <div class="MainMenu">
                <table cellspacing="2" cellpadding="2" border="0">
                    <tr>
                        <td class="MainCMSMenuHighlightedMenuItem">Home</td>
                        <td class="MainCMSMenuItem">Page 1</td>
                    </tr>
                </table>
            </div>

            <!-- main content -->
            <asp:ContentPlaceHolder ID="plcMain" runat="server"></asp:ContentPlaceHolder>
            <!-- /main content -->

            <!-- footer -->
            <div class="Footer">
                This is a sample web site for Kentico CMS
            </div>
        </div>

    </form>
</body>
```

When you switch to the **Design** tab, you should see a page preview like this:



Save the changes.



Using CSS-based layout instead of tables

If you prefer using a CSS-based layout, you can easily change the HTML code here and replace the tables with other elements (`<div>`, ``, etc.). We use a table-based layout by default since it's easier to understand, although we are aware of the advantages of a CSS-based layout.

7.6 Main menu

Now we will add a dynamic **drop-down menu** to our master page. The drop-down menu can be implemented using either the **CMSMenu** or **CMSListMenu** control. The first option is easier to use if you are not familiar with complex CSS styles, so we will use it now.

Please note: If you prefer a drop-down menu based on CSS styles and UL/LI elements, you can try using the **CMSListMenu** later (you can find more details and examples in the Kentico CMS Controls Reference).

Switch to the **Source** mode of the **MyMaster.master** page and drag and drop the **CMSMenu** control inside the `<div class="MainMenu">` element. Remove the original `<table>` element used for the static menu. The main menu section will look like this:

```
<!-- main menu -->
<div class="MainMenu">
    <cms:CMSMenu ID="CMSMenu1" runat="server" />
</div>
```

Now switch back to the **Design** tab and set the following properties of the **CMSMenu** control:

- **Path:** /%
- **Layout:** Horizontal
- **CSSPrefix:** ;sub
- **Cursor:** Pointer

The **Path** property specifies that the menu should start displaying pages from the root of the site structure. The **Layout** property allows you to choose between a vertical and horizontal menu. The **CSSPrefix** property specifies the names of CSS classes applied to the main menu (standard style

class names) and for sub-menus (all style class names will have the **sub** prefix). The **Cursor** property specifies the type of cursor displayed when a user hovers over the menu.

Save the changes.

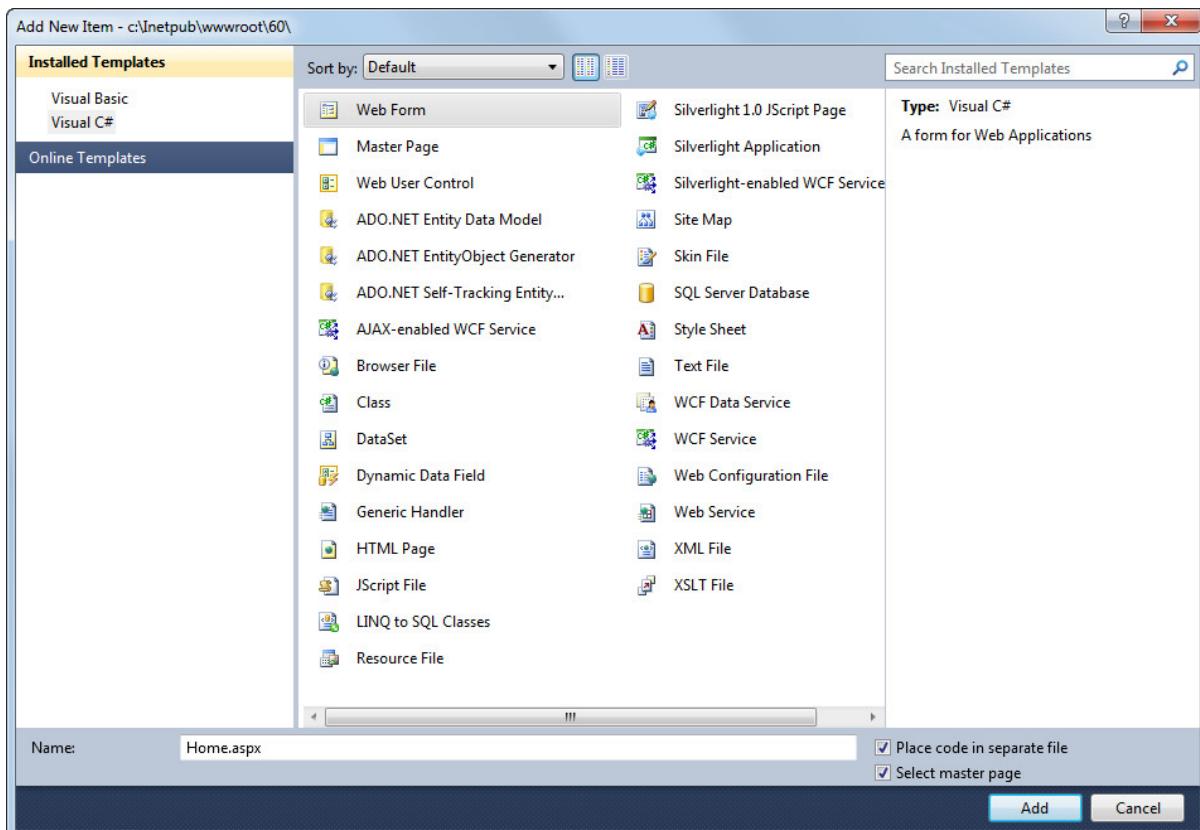


Kentico CMS Controls and Web Parts

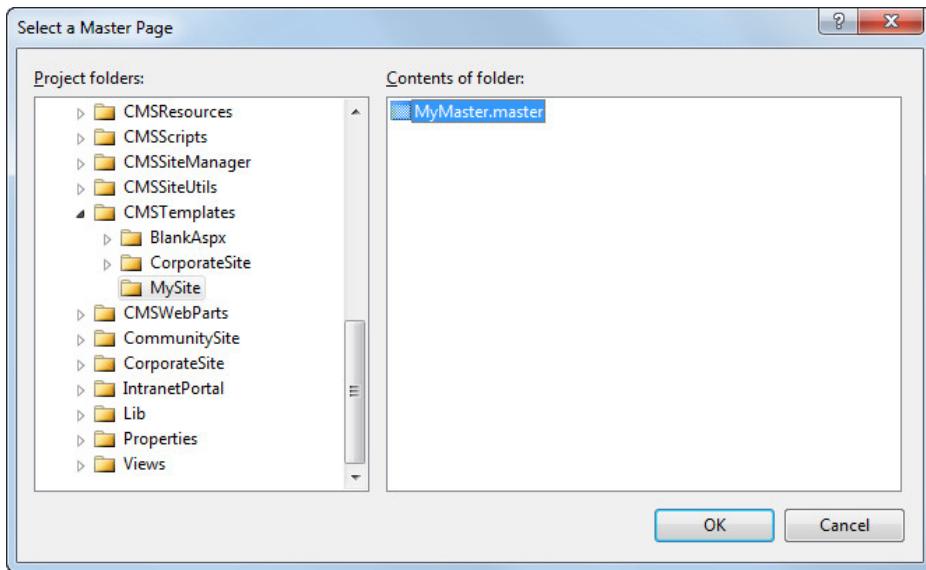
While Kentico CMS is delivered with a set of flexible server controls in the **CMSControls.dll** library, large amounts of functionality are only available through web parts that are stored in the **CMSWebParts** folders. These web parts are standard ASCX user controls and they can be used on both portal engine templates and on ASPX pages. You only need to drag and drop the web parts onto your ASPX page and set their properties in the Properties window of Visual Studio. All CMS controls have a corresponding web part as well.

7.7 Home page

Now we will create the home page template for our website. Right-click the **CMSTemplates/MySite** folder in the **Solution Explorer** and click **Add new item...**. Choose to create a new **Web Form** called *Home.aspx* and check the **Select master page** box:



Click **Add** and choose the **MyMaster.master** page from the **CMSTemplates/MySite** folder in the next dialog:



Copy the whole `<!-- main content -->` section from the `home.htm` file inside the `<asp:Content>` element. Now we will replace the static content with editable regions so that the page can be managed by content editors:

- Remove the whole "Welcome to Our Business, Inc. website..." text section.
- Remove the whole "Our Business, Inc. ..." content of the right box.

The complete code will look like the following:

Please note: If you installed the Kentico CMS project as a web application, you need to rename the **CodeFile** attribute in the **Page** directive on the first line to **Codebehind** for the code example to be functional.

```
<%@ Page Title="" Language="C#" MasterPageFile="~/CMSTemplates/MySite/MyMaster.master" AutoEventWireup="true" CodeFile="Home.aspx.cs" Inherits="CMSTemplates_MySite_Home" %>

<asp:Content ID="Content1" ContentPlaceHolderID="plcMain" Runat="Server">
<!-- main content -->
    <table style="width:100%;height:500px;border: 0px">
        <tr valign="top">
            <!-- left column -->
            <td style="width:280px" class="HomePageLeftColumn">

            </td>
            <!-- center column -->
            <td style="padding: 3px 5px 0px 5px; width:450px;">
                <!-- center box -->
                <table cellspacing="0" cellpadding="0" border="0" class="ContainerWithCorners" width="100%">
                    <tr class="ContainerWithCornersRow">
                        <td class="ContainerWithCornersTopLeft">&nbsp;</td>
                        <td class="ContainerWithCornersTop">&nbsp;</td>
                        <td class="ContainerWithCornersTopRight">&nbsp;</td>
```

```
</tr>
<tr>
    <td class="ContainerWithCornersLeft">&nbsp;</td>
    <td class="ContainerWithCornersContent" valign="top">
        </td>
    <td class="ContainerWithCornersRight">&nbsp;</td>
</tr>
<tr class="ContainerWithCornersRow">
    <td class="ContainerWithCornersBottomLeft">&nbsp;</td>
    <td class="ContainerWithCornersBottom"></td>
    <td class="ContainerWithCornersBottomRight">&nbsp;</td>
</tr>
</table>
</td>
<!-- right column -->
<td style="padding: 3px 0px 0px 5px; width:270px">
    <!-- text box -->
    <table cellpadding="0" cellspacing="0" style="width: 100%; margin-bottom: 10px;" class="Blue">
        <tr>
            <td class="BoxTitle">Contact us</td>
        </tr>
        <tr>
            <td class="BoxArea">
                </td>
            </tr>
        </table>
    </td>
</tr>
</table>
<!-- /main content -->
</asp:Content>
```

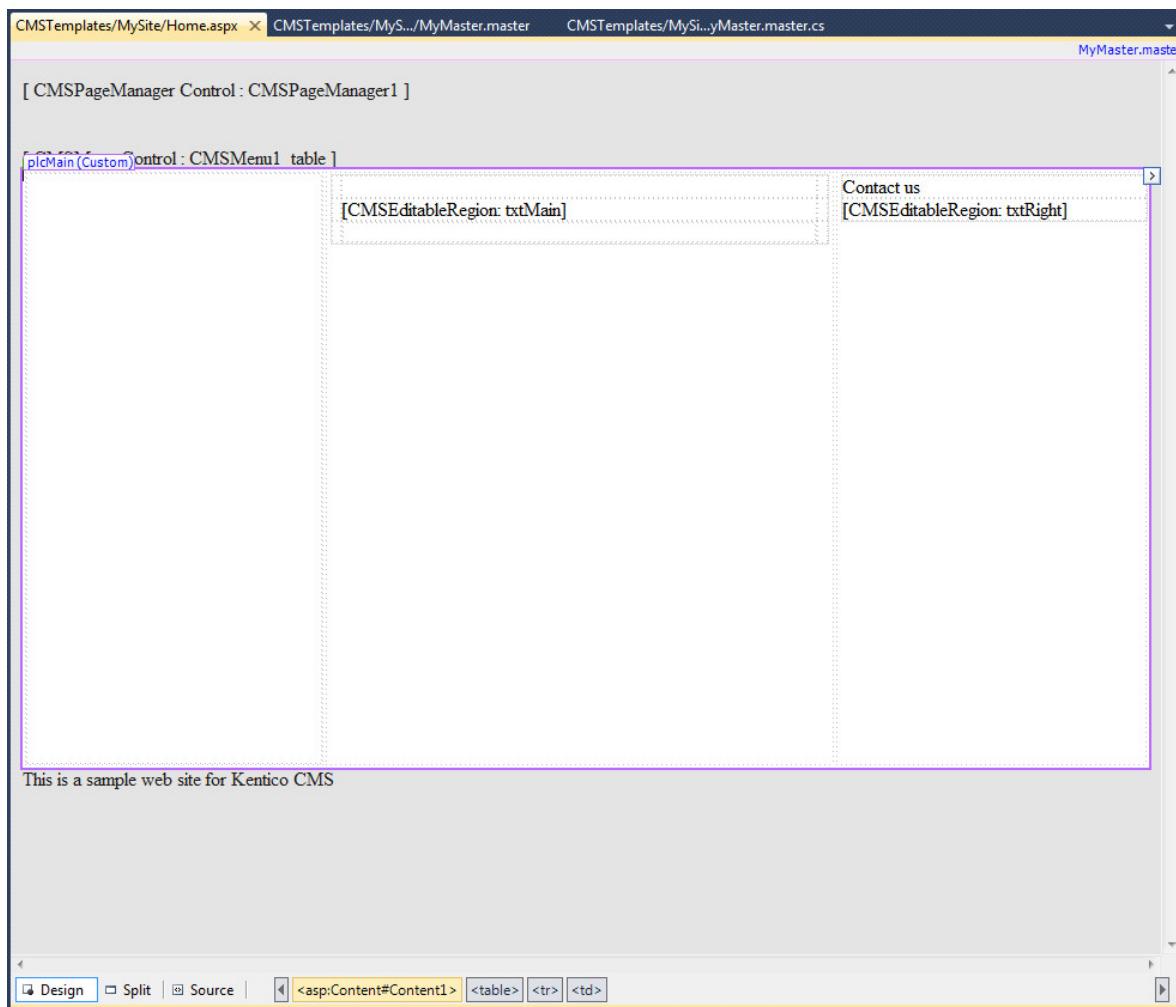
Modify the code of the table in the **<!-- center box -->** section according to the following:

```
<!-- center box -->
<table cellspacing="0" cellpadding="0" border="0" class="ContainerWithCorners" width="100%">
    <tr class="ContainerWithCornersRow">
        <td class="ContainerWithCornersTopLeft">&nbsp;</td>
        <td class="ContainerWithCornersTop">&nbsp;</td>
        <td class="ContainerWithCornersTopRight">&nbsp;</td>
    </tr>
    <tr>
        <td class="ContainerWithCornersLeft">&nbsp;</td>
        <td class="ContainerWithCornersContent" valign="top">
            <cms:CMSPagePlaceholder ID="plcZone" runat="server">
                <LayoutTemplate>
                    <cms:CMSWebPartZone ID="zoneMain" runat="server" />
                </LayoutTemplate>
            </cms:CMSPagePlaceholder>
        </td>
        <td class="ContainerWithCornersRight">&nbsp;</td>
    </tr>
    <tr class="ContainerWithCornersRow">
```

```
<td class="ContainerWithCornersBottomLeft">&nbsp;</td>
<td class="ContainerWithCornersBottom"></td>
<td class="ContainerWithCornersBottomRight">&nbsp;</td>
</tr>
</table>
```

As you can see, a **CMSPagePlaceholder** control was added to the center cell of the middle row, which defines an area of the page that will be editable through the browser. This area will later be configured to allow content editors to easily customize the design of the Home page.

Switch to the **Design** tab and drag and drop a **CMSEditableRegion** control into the bottom cell of the table on the right of the page:



Set the following properties of the **CMSEditableRegion** control:

- **ID:** txtRight
- **DialogHeight:** 280
- **RegionType:** HtmlEditor
- **RegionTitle:** Right content

Switch to the **code behind** and add a reference to the **CMS.UIControls** namespace:

[C#]

```
using CMS.UIControls;
```

[VB.NET]

```
Imports CMS.UIControls
```

You also need to change the class definition so that it inherits from the **TemplatePage** class:

[C#]

```
public partial class CMSTemplates_MySite_Home : TemplatePage
```

[VB.NET]

```
Partial Class CMSTemplates_MySite_Home  
    Inherits TemplatePage
```

Save the changes.

Our master page and page template for the home page are ready. Now we need to register the home page template in Kentico CMS. Open Kentico CMS in a web browser and go to **Site Manager -> Development -> Page templates**.

Click the  **New category** button and set the following properties:

- **Category display name:** My website
- **Category name:** mysite

Click  **New template** and enter the following values:

- **Template display name:** Home page
- **Template code name:** HomePage

Click **OK** and set the following values on the **General** tab:

- **Template type:** ASPX + Portal page
- **File name:** ~/CMSTemplates/MySite/Home.aspx

The screenshot shows the Kentico CMS 6.0 Page templates interface. On the left, there is a tree view of 'All page templates' under 'My website'. The 'Home page' template is selected. The main panel shows the 'General' tab of the 'Home page' template configuration. The 'Template display name' is set to 'Home page', 'Template code name' is 'HomePage', and 'Category' is 'My website'. There is a 'Template description' field, a 'Thumbnail' upload field, and a 'Template type' dropdown set to 'ASPX page'. A file path '~/CMSTemplates/MySite/home.aspx' is shown in the 'File name' field with 'Select' and 'Clear' buttons.

Click **Save** and switch to the **Sites** tab. Assign the new page template to your website.

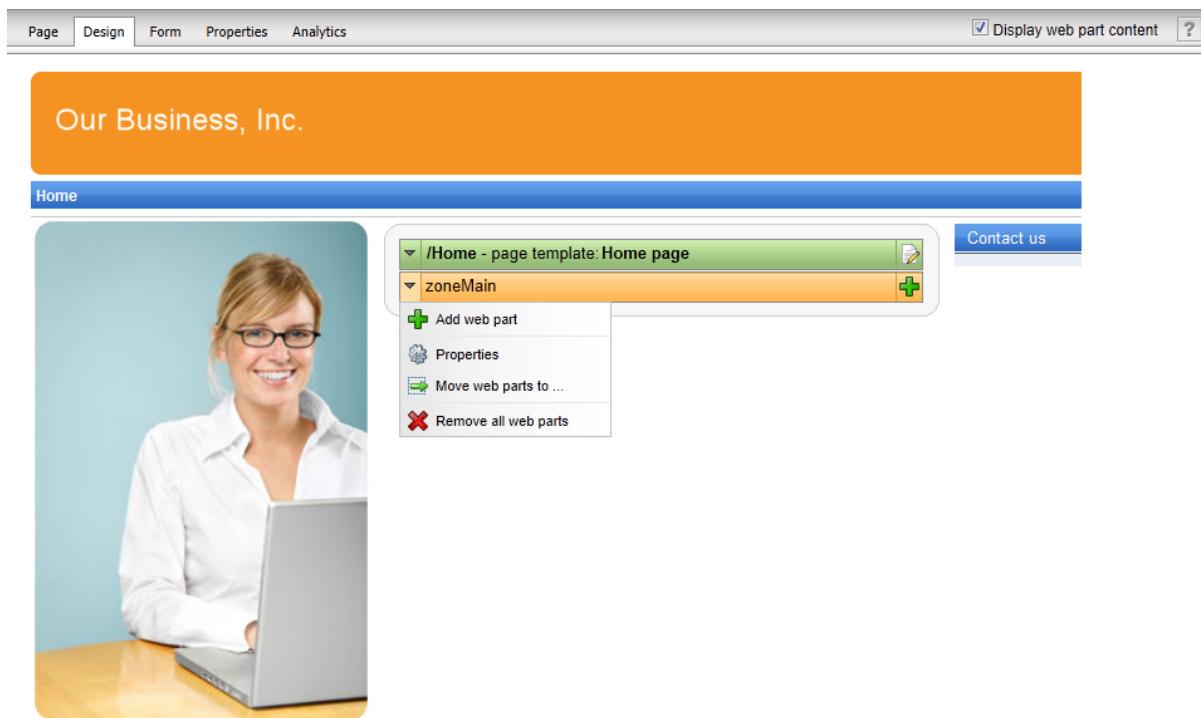
The screenshot shows the Kentico CMS 6.0 Page templates interface with the 'Sites' tab selected. It displays a message 'The changes were saved.' and a list of websites where the page template is available: 'My website'. Below this is a table with columns 'Site name' and 'My website'. At the bottom are 'Remove selected' and 'Add sites' buttons.

Go to **CMS Desk -> Content**. Select the root of the content tree and click **New**. Choose to create a **Page (menu item)** and use the following values:

- **Page name:** Home
- **Use existing page template:** My website/Home page

Click **Save** to create the page. Switch to the page's **Design** tab. You will see the editable area that was defined in the code of the page template. Expand the menu (▼) of the **zoneMain** zone and select

Properties to open its configuration dialog.

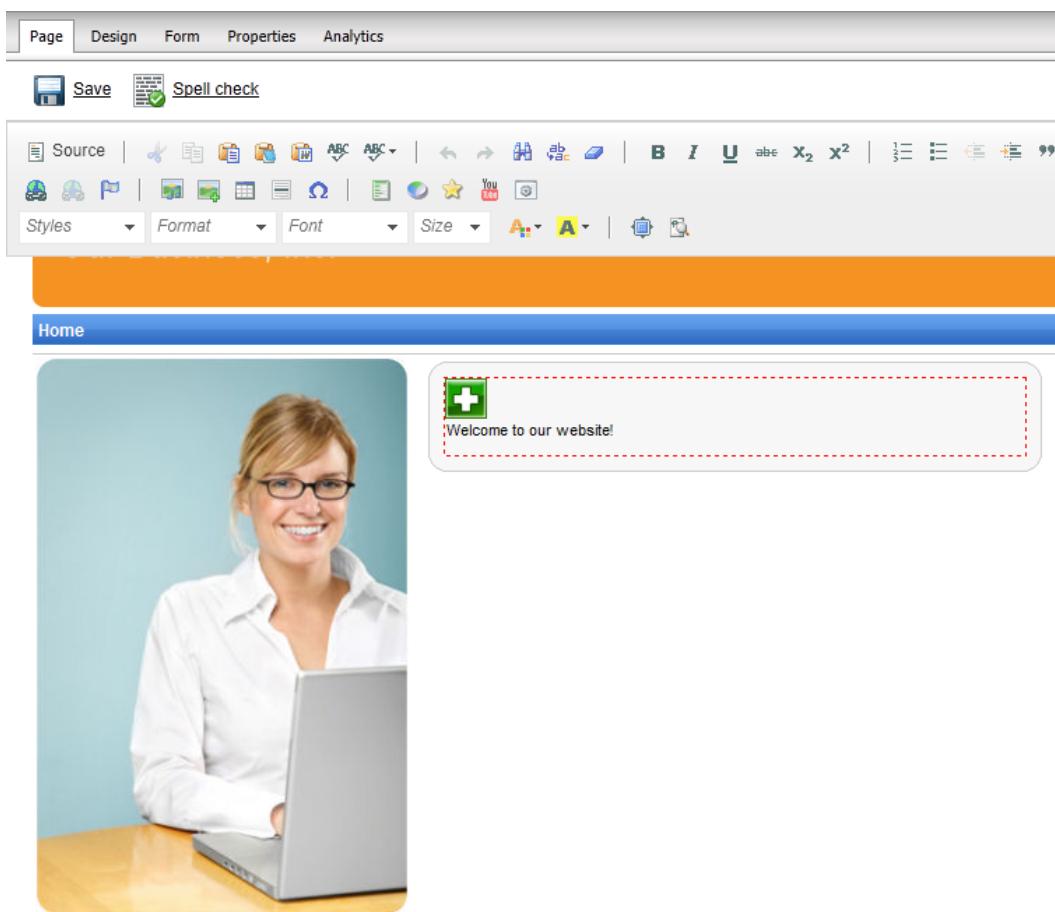


Change the **Widget zone type** property from *None* to *Customization by page editor* and click **OK**.

Open the **Page** tab and use the **Add widget** (+) button to place the **General -> Text** widget onto the page. Set its **Text** property to *Welcome to our website!* and click **OK**. As you can see, the design of the page can quickly be modified directly through the browser using widgets. This approach can be useful once the website has some more content or features to be displayed.

Next, enter the following text into the **Right content** editable region: *Call 800 111 2222*

Click **Save**.



You can view the page in **Live site** mode to see how the home page of your website will appear to visitors.

Configuring the website home page

When a site visitor comes to the root of your website (e.g. to *http://www.example.com*), the system needs to know which page should be displayed by default as the home page. Go to **Site Manager -> Settings**, select **My website** in the **Site** drop-down menu, click the **Content** setting category and make sure the value of **Default alias path** is set to */Home*, which is the alias path of our new home page. If not, please uncheck the **Inherit from global settings** box, enter the value and click **Save**.

The screenshot shows the Kentico Site Manager interface. The left sidebar has a tree view under 'Settings' with 'Content' selected. The main content area is titled 'Content' and contains sections for 'Page not found', 'Multilingual', and 'Web site content'. In the 'Web site content' section, the 'Default alias path' field is set to '/Home' and is highlighted with a red border. There are also 'Select' and 'Inherit from global settings' buttons.

7.8 News page

Now we will create the News section of our website. Go to Visual Studio and create a new web form in the **CMSTemplates\MySite** folder, call it *NewsPage.aspx*, check the **Select master page** box and click **Add**. Choose the **MyMaster.master** master page and click **OK**.

Switch to **Design** mode, drag and drop and configure the following controls:

CMSBreadCrumbs (no properties to be set)

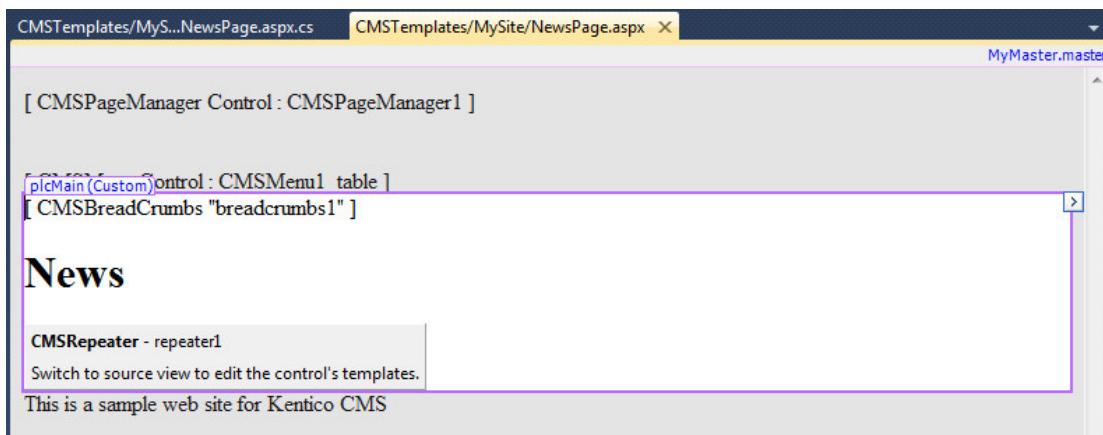
CMSRepeater

- **ClassNames:** cms.news
- **TransformationName:** cms.news.preview
- **SelectedItemTransformationName:** cms.news.default
- **ItemSeparator:** <hr />

Switch to **Source** mode and add the following HTML code between the two controls:

```
<h1>News</h1>
```

When you switch back to the **Design** tab, you should see a page like this:



Switch to the **code behind** and add a reference to the **CMS.UIControls** namespace:

[C#]

```
using CMS.UIControls;
```

[VB.NET]

```
Imports CMS.UIControls
```

You also need to change the class definition so that it inherits from the **TemplatePage** class:

[C#]

```
public partial class CMSTemplates_MySite_NewsPage : TemplatePage
```

[VB.NET]

```
Partial Class CMSTemplates_MySite_NewsPage  
    Inherits TemplatePage
```

Save the changes.

Page Template Registration

Go to **Site Manager -> Development -> Page templates**, select the **My website** category and click **New template**. Create a new page template with the following values:

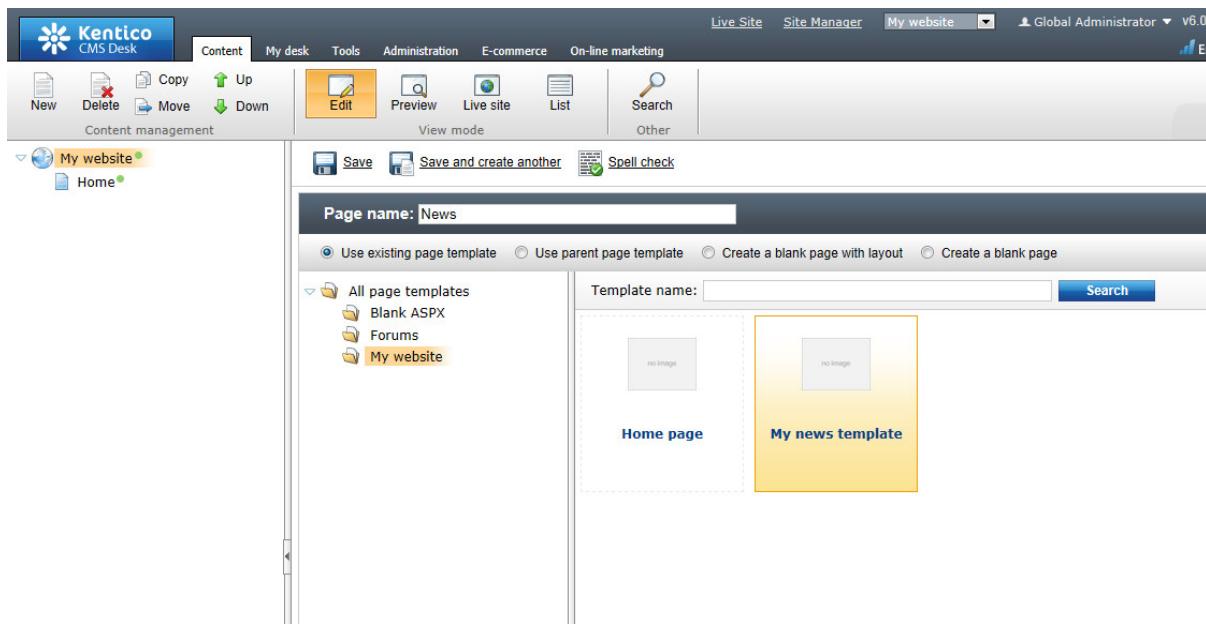
- **Template display name:** My news template
- **Template code name:** mynewstemplate

On the **General** tab of the page template, please set the following:

- **Template type:** ASPX page
- **File name:** ~/CMSTemplates/MySite/NewsPage.aspx

Click **Save** and switch to the **Sites** tab. Assign the new page template to **My website**.

Go to **CMS Desk -> Content**, select the root and click **New**. Choose to create a new **Page (menu item)** using the **My website/My news template** page template and name the page **News**.



Click **Save**. Select the **News** page, click **New** and create a **News** document with the following values:

- **News title:** News 1
- **Release date:** click the date-time picker, click **Now** and then **OK**.
- **News summary:** News 1 summary.
- **News text:** News 1 text.
- **Publish from/to:** leave the fields blank.

Click **Save and create another** and enter the following values:

- **News title:** News 2
- **Release date:** click the date-time picker, click **Now** and then **OK**.
- **News summary:** News 2 summary.
- **News text:** News 2 text.
- **Publish from/to:** leave the fields blank.

Click **Save**.

When you select **/News** and **Live site** now, you will see the list of news under the **News** section:

Our Business, Inc.

Home News

News

News

[News 1](#)

News 1 summary.

[News 2](#)

News 2 summary.

This is a sample web site for Kentico CMS

As you can see, the main **/News** page displays the list of the news items that are placed under it. This is an example of how the content is logically structured in Kentico CMS. When you click **/News/News 1** now, you will see the detail view:

Our Business, Inc.

Home News

[News](#) > News 1

News

News 1

9/21/2011

News 1 summary.

News 1 text.

This is a sample web site for Kentico CMS

The breadcrumbs now show the current path on the website: **News > News 1**. The position is also reflected in the URLs:

- The URL of the News page is **/news.aspx**
- The URL of the News 1 page is **/news/news-1.aspx**

This makes the website more accessible to both people and search engines, such as Google.

How it works

1. You go to the **/News page**.
2. The **NewsRepeater** web part checks if you have selected some particular news item (based on its Document types property value).
3. It finds out that you have selected a page document, so it looks for all underlying news documents and displays them as a list using the **cms.news.preview** transformation.
4. When you select some particular news item, such as **/News/News 1**, the **NewsRepeater** web part uses the **cms.news.detail** transformation instead and displays the details.



Path expressions

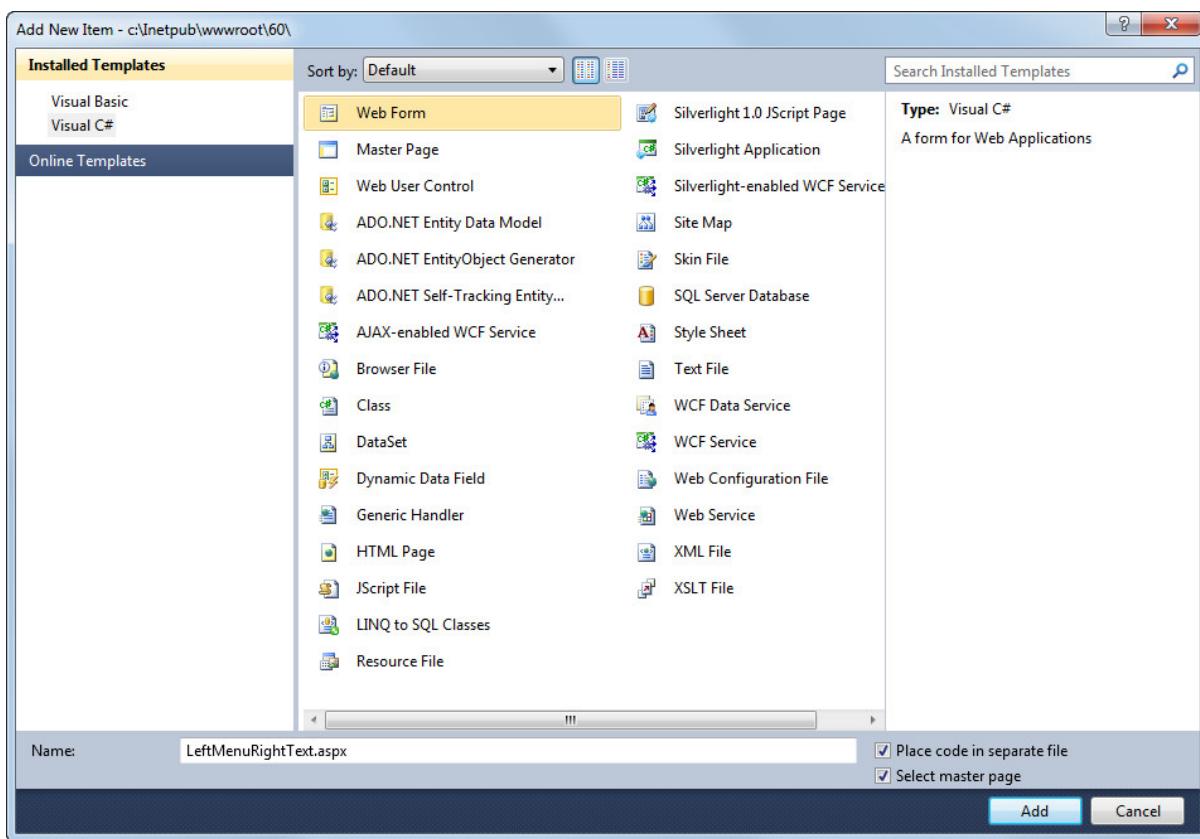
The **Path** property in web parts supports the following special expressions that allow you to select the content dynamically:

/%	All documents on the website.
/news/%	All documents under /News.
/news/news1	The News1 document.
./%	All items under the current document.
./logo	The Logo document under the current document.
./images/%	All images under the Images child document.
../contacts/%	All documents under the Contacts document on the same content level.
	All documents under the current first level document.
/{}0}/%	Example: if the currently selected document is <u>/news/news1</u> the expression is evaluated as <u>/news/%</u>

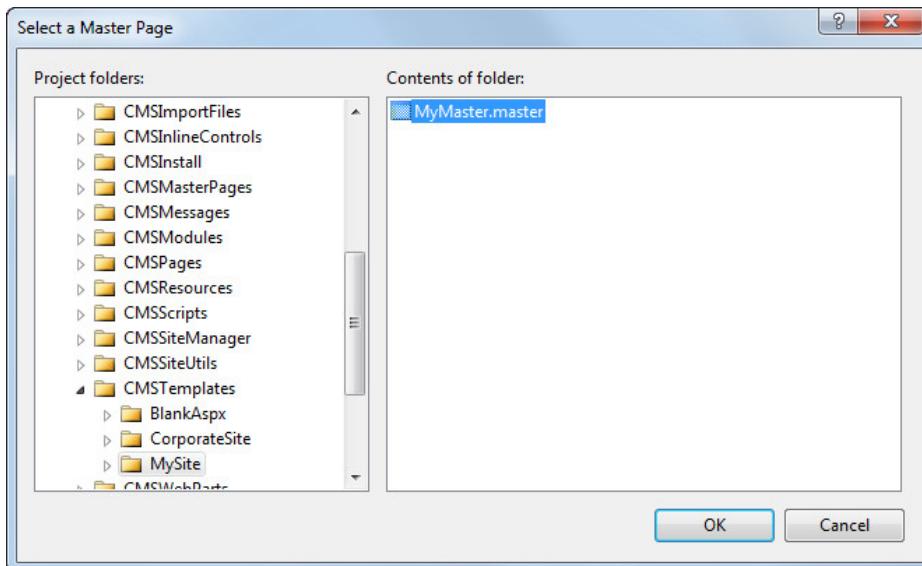
7.9 Services page

Now we will create a new site section for services. This section will contain a left tree menu and a single editable region.

Go to Visual Studio and choose to create a new web form in the **CMSTemplates\MySite** folder. Name it *LeftMenuRightText.aspx* and check the **Select master page** box:



Choose the **CMSTemplates\MySite\MyMaster.master** page in the next dialog:



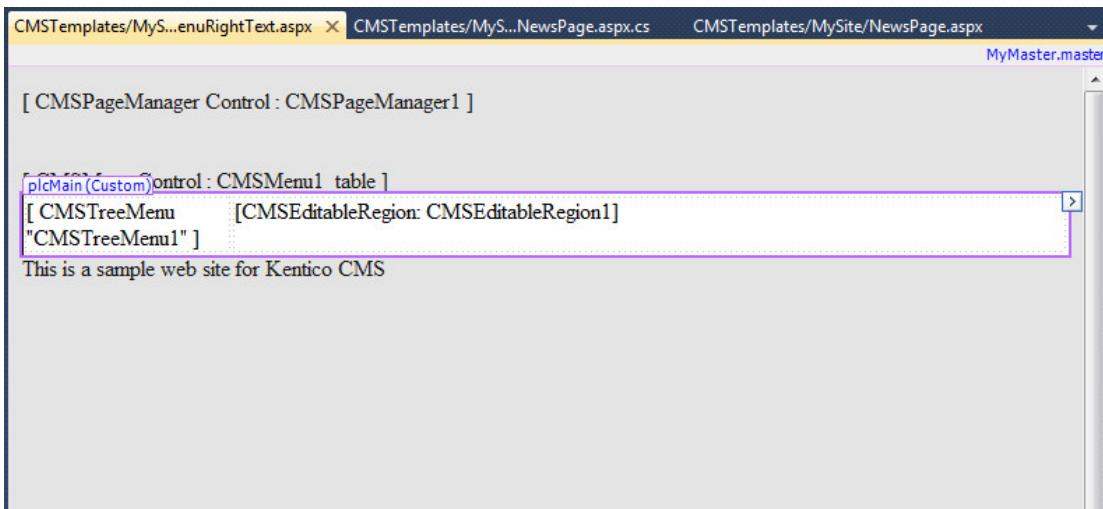
Now enter the following HTML layout code inside the **<asp:content>** element of the newly created page:

```
<table width="100%">
    <tr valign="top">
```

```
<td width="20%">  
  
</td>  
<td width="80%">  
  
</td>  
</tr>  
</table>
```

Switch to the **Design** tab and you will see a preview of the page, including the inherited master page.

Drag and drop the **CMSTreeMenu** control into the left column and the **CMSEditableRegion** control into the right column:



Set the following properties for the controls:

CMSTreeMenu1:

- **Path:** /{0}/% (this means that the menu starts from the second content tree level)
- **MenuItemImageUrl:** ~/app_themes/mysite/images/bullet.gif
- **MenuItemOpenImageUrl:** ~/app_themes/mysite/images/bullet.gif

The ~ character represents the root of the website and it ensures that the image will be displayed correctly whether you run the websites in the root or in a virtual directory.

CMSEditableRegion1:

- **RegionType:** HTMLEditor
- **DialogHeight:** 400
- **RegionTitle:** Main Text

Switch to the **code behind** and add a reference to the **CMS.UIControls** namespace:

[C#]

```
using CMS.UIControls;
```

[VB.NET]

```
Imports CMS.UIControls
```

You also need to change the class definition so that it inherits from the **TemplatePage** class:

[C#]

```
public partial class CMSTemplates_MySite_LeftMenuRightText : TemplatePage
```

[VB.NET]

```
Partial Class CMSTemplates_MySite_LeftMenuRightText  
    Inherits TemplatePage
```

Save the changes.

Now we need to register the new page template. Go to **Site Manager -> Development -> Page templates**, select the **My website** category and click **New template**. Create a new page template with the following values:

- **Template display name:** Left menu with right text
- **Template code name:** LeftMenuWithRightText

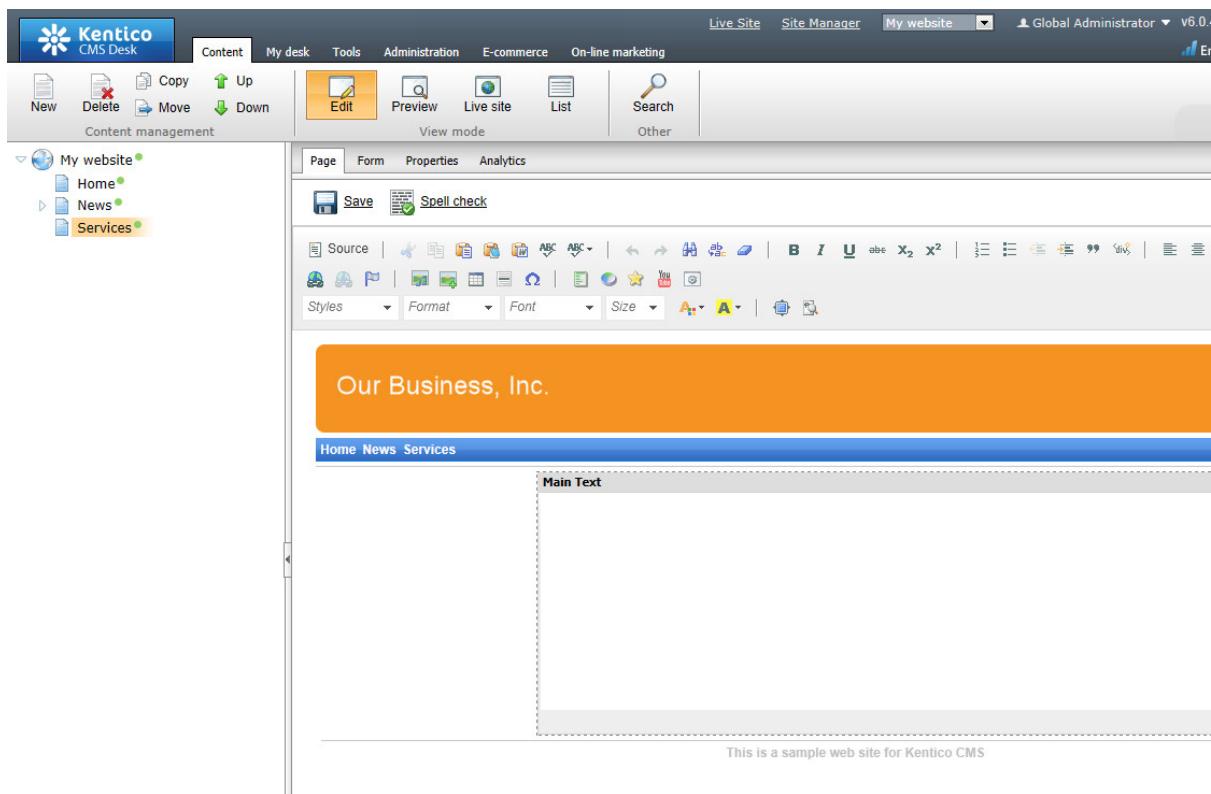
On the **General** tab of the page template, please set the following:

- **Template type:** ASPX page
- **File name:** ~/CMSTemplates/MySite/LeftMenuRightText.aspx

Click **Save** and switch to the **Sites** tab. Assign the new page template to **My website**.

Now that the page template is created, you can start adding pages based on it. Go to **CMS Desk -> Content**, select the root and click **New**. Choose to create a new **Page (menu item)** using the **My website/Left menu with right text** template and name the page **Services**. Click **Save**.

You will be redirected to the **Page** tab and here you can enter some text content onto the Services page:



Adding sub-pages

Click **New** in the main toolbar to create a new page under the **/Services** page. Call the page **Service 1** and choose to use the **My website/Left menu with right text page** template . Click **Save**. Enter some text into the editable region and save again.

7.10 Products page

7.10.1 Overview

Now we will add a new Products section displaying a list of computers and their technical specification. You will learn how to create a new *Computer* document type and how to display a list of computers on the site. You will also learn how to write transformations.

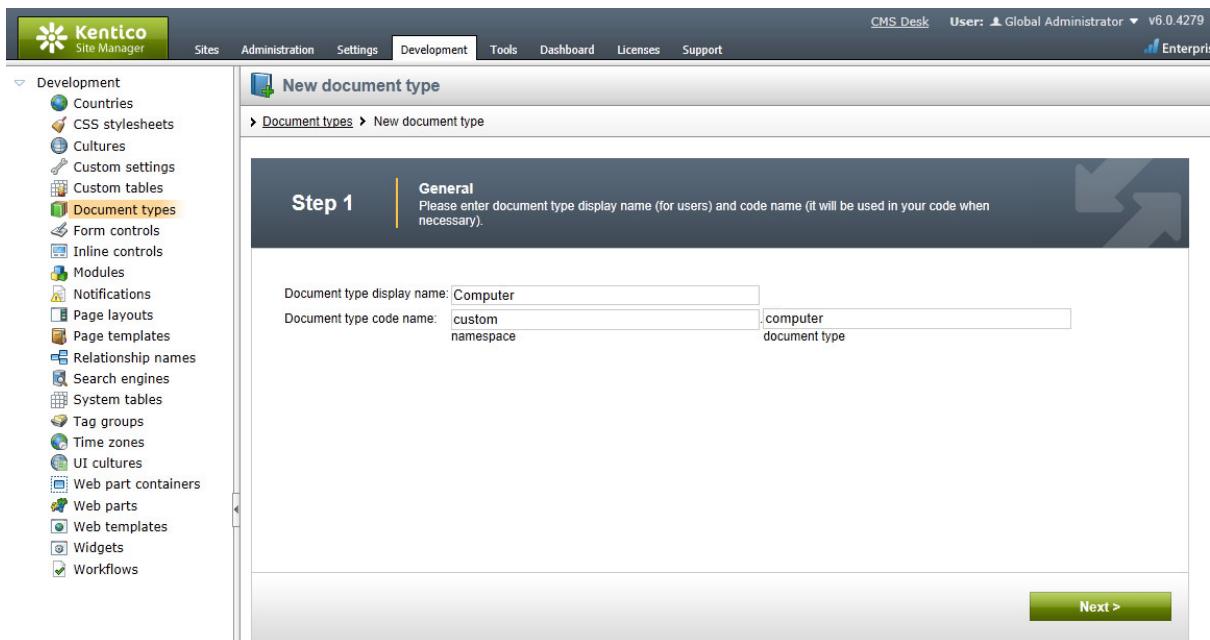
7.10.2 New document type

Each document in the Kentico CMS repository is of some type, such as news, product, article, etc. Each document type has its own fields. Our document type will describe a computer, so it will have a computer name, processor type, RAM size, disk size and a product image field.

Go to **Site Manager -> Development -> Document types** and click **New document type**. You will be redirected to the New document type wizard. In the first step, enter the following values:

- **Document type display name:** Computer (*this name will be displayed to the users in the administration interface*)
- **Document type code name:** custom.computer (*custom is your namespace to distinguish your*

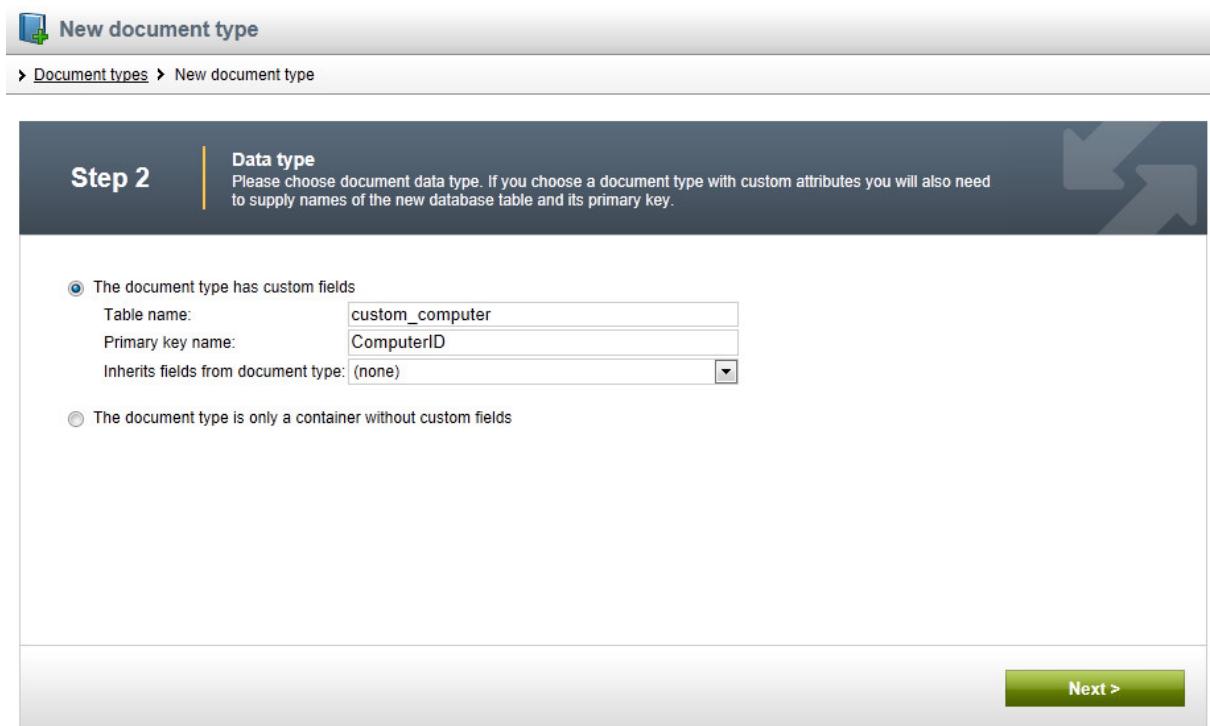
(document types from system types that use the cms namespace, computer is the document type). You will use this value in the properties of controls later.



The screenshot shows the Kentico CMS 6.0 Site Manager interface. On the left, there's a navigation sidebar with various options like 'Development', 'Custom tables', 'Document types', and 'Workflows'. The 'Document types' option is highlighted. The main content area has a title 'New document type' and a breadcrumb 'Document types > New document type'. It's step 1 of the process, titled 'General'. It asks for a 'Document type display name' (set to 'Computer') and a 'Document type code name' (set to 'custom namespace'). A note says: 'Please enter document type display name (for users) and code name (it will be used in your code when necessary.)'. At the bottom right is a green 'Next >' button.

Click **Next**.

In step 2, you need to choose the name of the database table that will be used for storing computer details. You also need to enter the name of the primary key in this table. Leave the default values:



The screenshot shows the 'Data type' step of the wizard. It has a note: 'Please choose document data type. If you choose a document type with custom attributes you will also need to supply names of the new database table and its primary key.' There are two radio button options: one for 'The document type has custom fields' (which is selected) and one for 'The document type is only a container without custom fields'. Under the first option, there are fields for 'Table name' (set to 'custom_computer'), 'Primary key name' (set to 'ComputerID'), and 'Inherits fields from document type' (set to '(none)'). At the bottom right is a green 'Next >' button.

Click **Next**.

The wizard has created a new database table for computers. Now you need to define the fields of the document type (columns of the table). Use the **New attribute** (+) button to create the following fields. For each field, enter the values, click  **Save field** and repeat the procedure until you have all the listed fields defined.

- **Column name:** ComputerName
- **Attribute type:** Text
- **Attribute size:** 200
- **Field caption:** Computer name
- **Form control type:** Input
- **Form control:** Text box

- **Column name:** ComputerProcessorType
- **Attribute type:** Text
- **Attribute size:** 200
- **Field caption:** Processor type
- **Form control type:** Input
- **Form control:** Drop-down list
- **Editing control settings -> Data source:** select *Options* and enter the following items into the text area, one per line:
Athlon;Athlon
Pentium XEON;Pentium XEON
Pentium Core 2 Duo;Pentium Core 2 Duo

- **Column name:** ComputerRamSize
- **Attribute type:** Integer number
- **Field caption:** RAM (MB)
- **Form control type:** Input
- **Form control:** Text box

- **Column name:** ComputerHddSize
- **Attribute type:** Integer number
- **Field caption:** HDD (GB)
- **Form control type:** Input
- **Form control:** Text box

- **Column name:** ComputerImage
- **Attribute type:** File
- **Allow empty value:** check the box
- **Field caption:** Image
- **Form control type:** Uploader
- **Form control:** Upload file

Step 3

Fields
Please define custom attributes of the document type and their appearance in the editing form. You can define attributes, such as product number, product weight, press release text, etc.

ComputerID*
ComputerName*
ComputerProcessorType*
ComputerRamSize*
ComputerHddSize*
ComputerImage

Save field
The changes were saved.

Database

Column name:	ComputerImage
Attribute type:	File
Attribute size:	
Allow empty value:	<input checked="" type="checkbox"/>

Display attribute in the editing form

Field appearance

Field caption:	Image
Form control type:	Uploader
Form control:	Upload file
Field description:	

Quick links:
[Database](#)
[Field appearance](#)
[Editing control settings](#)
[CSS styles](#)

Next >

Click **Next**.

Now you need to choose the field that will be used as the document name. Choose the **ComputerName** field. It means that when you create a new computer document, its name will be automatically taken from the ComputerName value and this value will appear in site navigation and in the CMS Desk content tree.

Step 4

Additional settings
Please choose the source field that will be used as a document name. You can choose either one of the custom fields or you can choose to use document name as a separate field.

Document name source: ComputerName

Next >

Click **Next**. In step 5, you need to select the document types under which the computers can be added

in the content tree. Click the **Add document types** button and add the **Page (menu item)** document type, which means the editors will be able to create computer documents only under pages, not under article or news documents in the content tree.

Step 5

Parent types
Please select document types under which this document template can be placed.

<input type="checkbox"/>	Document type name
<input checked="" type="checkbox"/>	Page (menu item) (CMS.MenuItem)

Remove selected **Add document types** ▾

Next >

Click **Next**. In step 6, you need to choose which websites will use this document type. Click the **Add sites** button and choose **My website** in the following dialog.

Step 6

Sites
Please select sites where this document type can be used:

<input type="checkbox"/>	Site name
<input checked="" type="checkbox"/>	My website

Remove selected **Add sites** ▾

Next >

Click **Next**. In Step 7, you are asked to specify how documents of this type should be indexed for searching and displayed in search results. Select the following values in the drop-downs:

- **Title field:** ComputerName
- **Content field:** DocumentContent

- **Image field:** ComputerImage
- **Date field:** DocumentCreatedWhen

Leave the default values for the rest of the options and click **Next**.

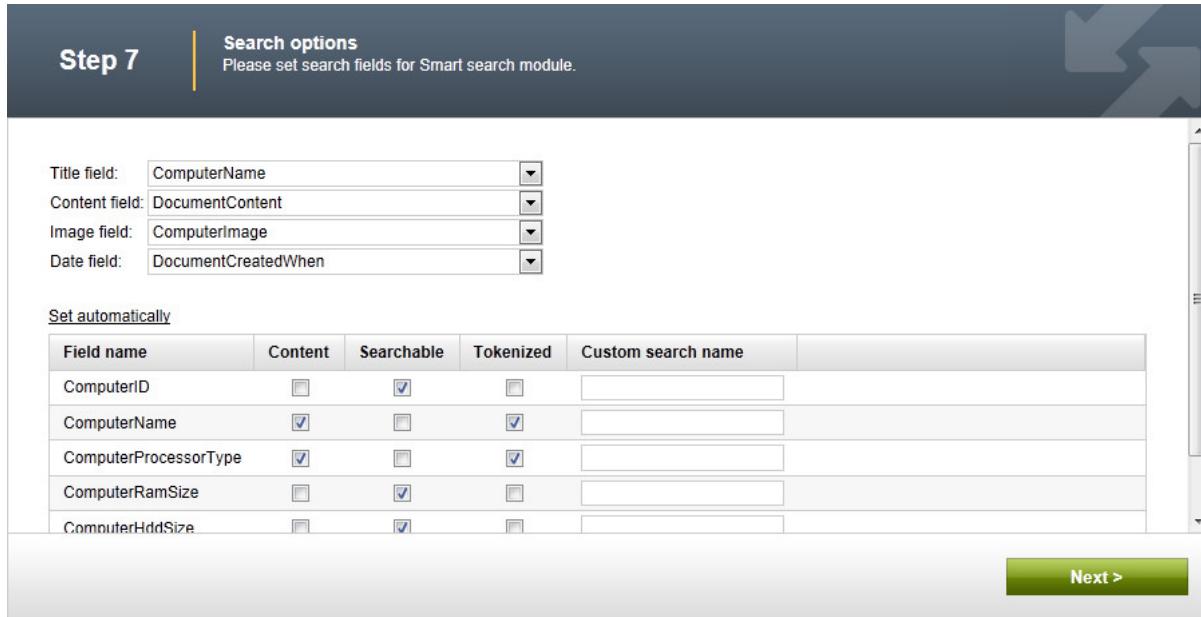
Step 7 | **Search options**
Please set search fields for Smart search module.

Title field: ComputerName
Content field: DocumentContent
Image field: ComputerImage
Date field: DocumentCreatedWhen

Set automatically

Field name	Content	Searchable	Tokenized	Custom search name
ComputerID	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
ComputerName	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
ComputerProcessorType	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
ComputerRamSize	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
ComputerHddSize	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Next >



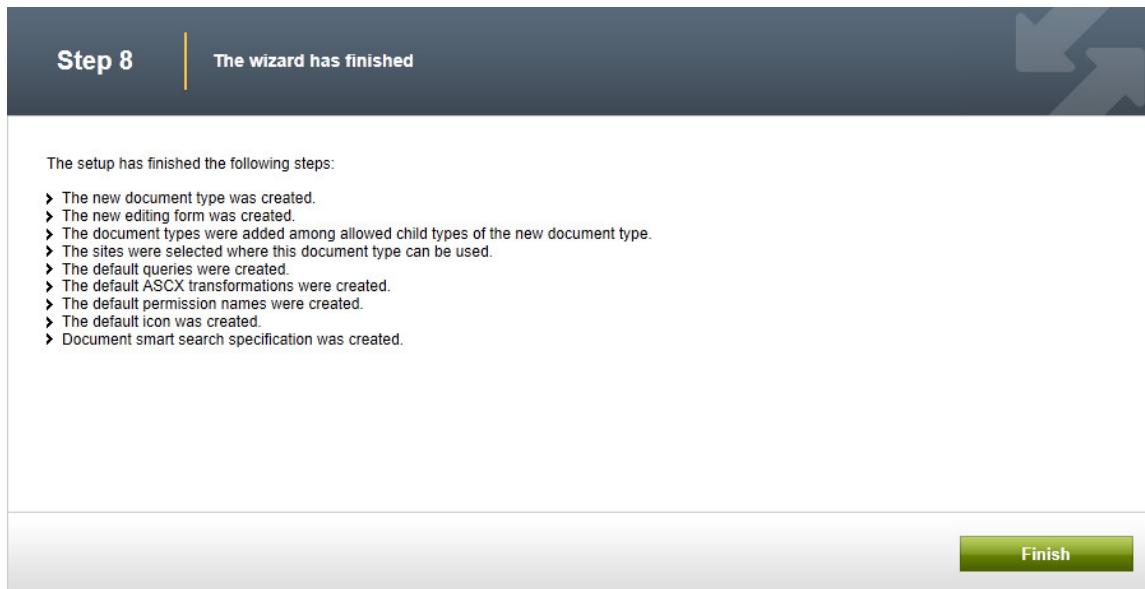
The wizard has finished the configuration of the new document type. It has automatically created not only the database table, but also the SQL queries for SELECT, INSERT, UPDATE, DELETE operations and a default transformation.

Step 8 | **The wizard has finished**

The setup has finished the following steps:

- The new document type was created.
- The new editing form was created.
- The document types were added among allowed child types of the new document type.
- The sites were selected where this document type can be used.
- The default queries were created.
- The default ASCX transformations were created.
- The default permission names were created.
- The default icon was created.
- Document smart search specification was created.

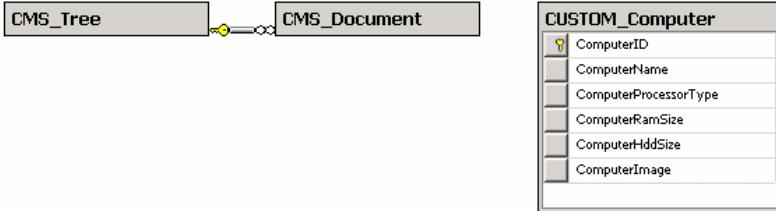
Finish



You have learned how to define a new document type.

 **How the content is stored**

As you already know, the new *Computer* document type has its own database table. Each document is stored in three tables: **CMS_TREE** (tree structure), **CMS_Document** (document properties and metadata) and the specific table of the given document type, in this case **CUSTOM_Computer**:

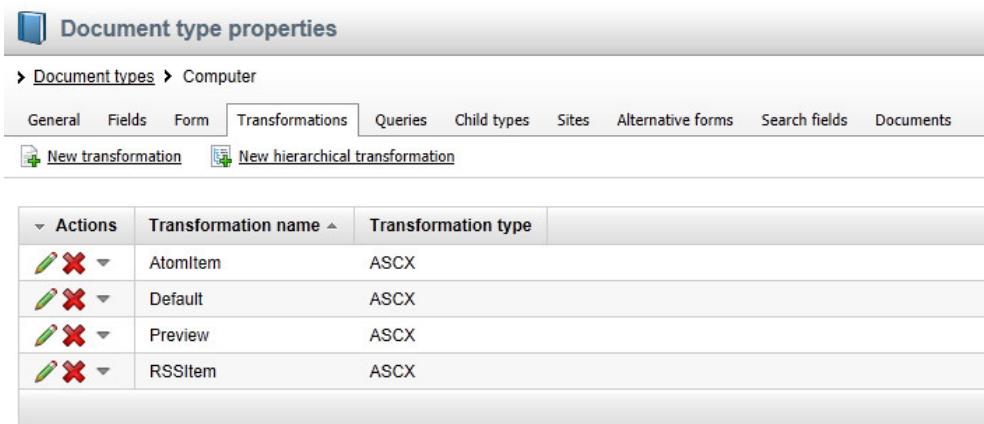


The system automatically ensures all operations on these tables. The advantage of this storage is that it's very fast and **you can easily write standard SQL SELECT queries to retrieve data** from the repository (i.e. from the Microsoft SQL Server database).

7.10.3 Transformations

Now that we have created a new document type, we need to prepare the transformations that will be used for displaying product details in list and in detail view mode.

In the **Computer** document type properties dialog, switch to the **Transformations** tab:



Actions	Transformation name	Transformation type
	AtomItem	ASCX
	Default	ASCX
	Preview	ASCX
	RSSItem	ASCX

As you can see, the wizard has created some default transformations. We will use them for our detail view. **Edit** () the **Default** transformation, clear the default code and enter the following code:

```
<h1>
<%# Eval("ComputerName") %>
</h1>
<table>
```

```

<tr>
  <td>
    Processor:
  </td>
  <td>
    <%# Eval("ComputerProcessorType") %>
  </td>
</tr>
<tr>
  <td>
    RAM ( MB ) :
  </td>
  <td>
    <%# Eval("ComputerRamSize") %>
  </td>
</tr>
<tr>
  <td>
    HDD ( GB ) :
  </td>
  <td>
    <%# Eval("ComputerHddSize") %>
  </td>
</tr>
<tr>
  <td>
    Image:
  </td>
  <td>
    <%# GetImage("ComputerImage") %>
  </td>
</tr>
</table>

```

Click  **Save**. As you can see the transformation code is similar to standard ItemTemplate code that you may already know from ASP.NET Repeater and DataList controls. It combines HTML code with ASP.NET commands and data binding expressions. You can also use built-in functions, such as **GetImage()** that simplify some tasks. You can find the list of the most important functions directly under the transformation code.

Now we will create a transformation for viewing computers in list mode. Go back to the transformation list and edit the **Preview** transformation. Clear the default code and enter the following code:

```

<div style="text-align:center; border: 1px solid #cccccc">
  <h2>
    <a href="<%# GetDocumentUrl() %>"><%# Eval("ComputerName") %></a>
  <h2>
    ?maxsidesize=120" />
  </div>

```

Click  **Save**.

Please note how the link to the document is created:

```
<a href="<%# GetDocumentUrl() %>"><%# Eval("ComputerName") %></a>
```

It consists of standard HTML tags for links and it inserts the URL and link text dynamically.

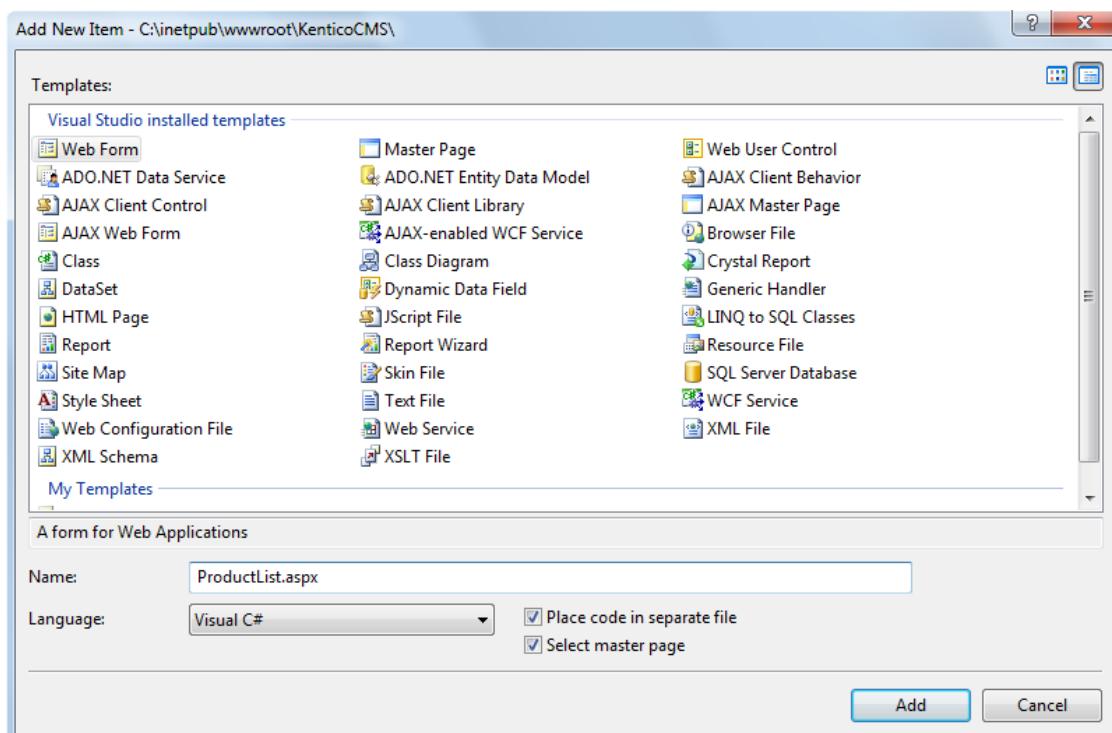
Similarly, you can create an image tag with a parameter that ensures automatic resizing of the longest side to 120 pixels on the server side:

```
?maxsidesize=120" />
```

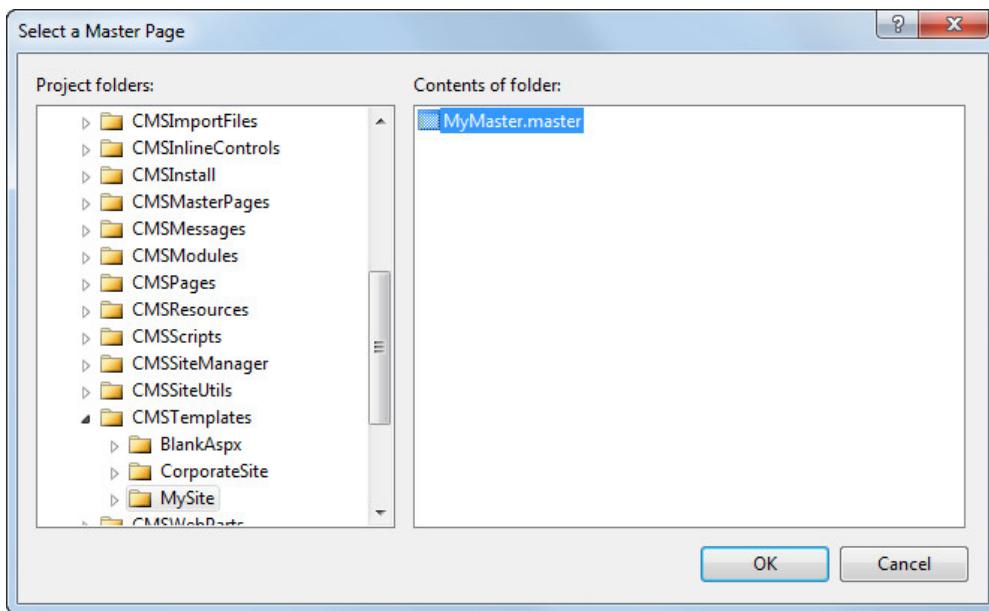
You have learned how to write transformations for displaying the content of structured documents.

7.10.4 Page template

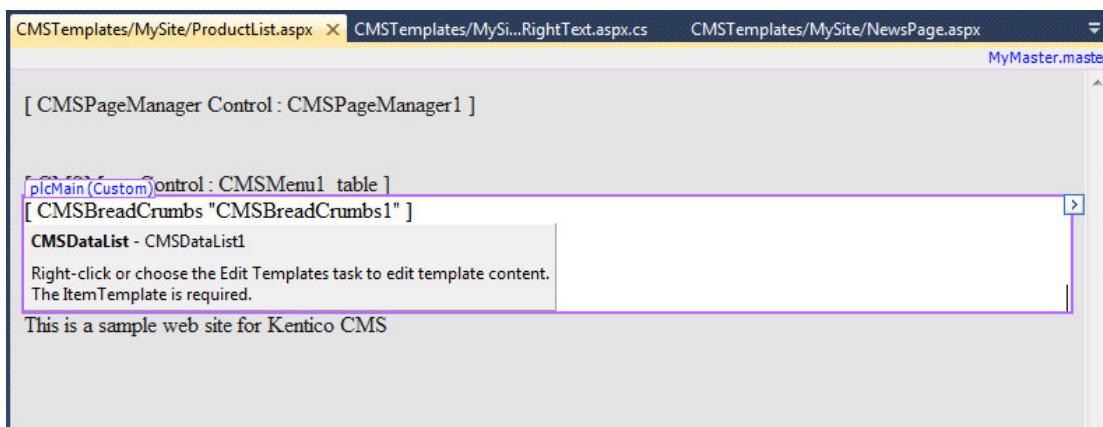
Now we get to the final step of this chapter: publishing computer specifications on your website. Go to Visual Studio and choose to create a web form in the **CMS\Templates\MySite** folder. Name the page *ProductList.aspx* and check the **Select master page** box:



Choose the **CMS\Templates\MySite\MyMaster.master** page in the next dialog:



Switch to the **Design** tab and you will see a preview of the page, including the inherited master page. Drag and drop the **CMSBreadCrumbs** and **CMSDataList** controls onto the page:



Set the following properties of the **CMSDataList** control:

- **ClassNames:** custom.computer (the document types to be displayed)
- **OrderBy:** ComputerName ASC
- **TransformationName:** custom.computer.preview
- **SelectedItemTransformationName:** custom.computer.default
- **RepeatColumns:** 2

Switch to the **code behind** and add a reference to the **CMS.UIControls** namespace:

[C#]

```
using CMS.UIControls;
```

[VB.NET]

```
Imports CMS.UIControls
```

You also need to change the class definition so that it inherits from the **TemplatePage** class:

[C#]

```
public partial class CMSTemplates_MySite_ProductList : TemplatePage
```

[VB.NET]

```
Partial Class CMSTemplates_MySite_ProductList  
    Inherits TemplatePage
```

Save the changes.

Now we need to register the new page template. Go to **Site Manager -> Development -> Page templates**, select the **My website** category and click **New template**. Create a new page template with the following values:

- **Template display name:** Product list
- **Template code name:** ProductList

On the **General** tab of the page template, please set the following:

- **Template type:** ASPX page
- **File name:** ~/CMSTemplates/MySite/ProductList.aspx

Click **Save** and switch to the **Sites** tab. Assign the new page template to **My website**.

Now that the page template is created, you can start adding pages based on it. Go to **CMS Desk -> Content**, select the root and click **New**. Choose to create a new **Page (menu item)** using the **My website/Product list** template and name the page *Products*. Click **Save**.

Now we need to enter some computer details. Select **/Products** in the content tree and click **New**. Choose to create a new **Computer**. Enter the following values:

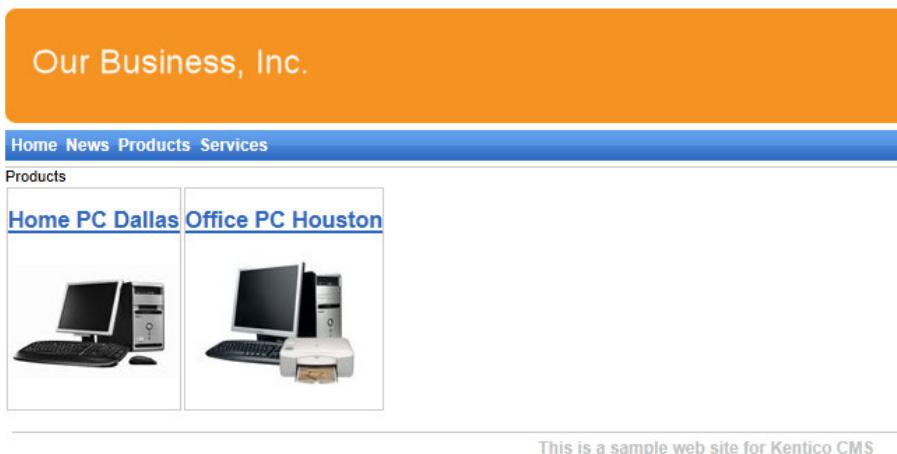
- **Computer name:** Home PC Dallas
- **Processor type:** Athlon
- **RAM (MB):** 512
- **HDD (GB):** 80
- **Image:** upload some image (you can find sample images in the <Kentico CMS installation>\CodeSamples\SampleWebTemplate\Computer_Images folder)
- **Publish from/to** - leave the values blank

Click **Save and create another** with the following values:

- **Computer name:** Office PC Houston
- **Processor type:** Pentium Core 2 Duo
- **RAM (MB):** 1024
- **HDD (GB):** 120
- **Image:** upload some image (you can find sample images in the <Kentico CMS installation>\CodeSamples\SampleWebTemplate\Computer_Images folder)
- **Publish from/to** - leave the values blank

Click  **Save**.

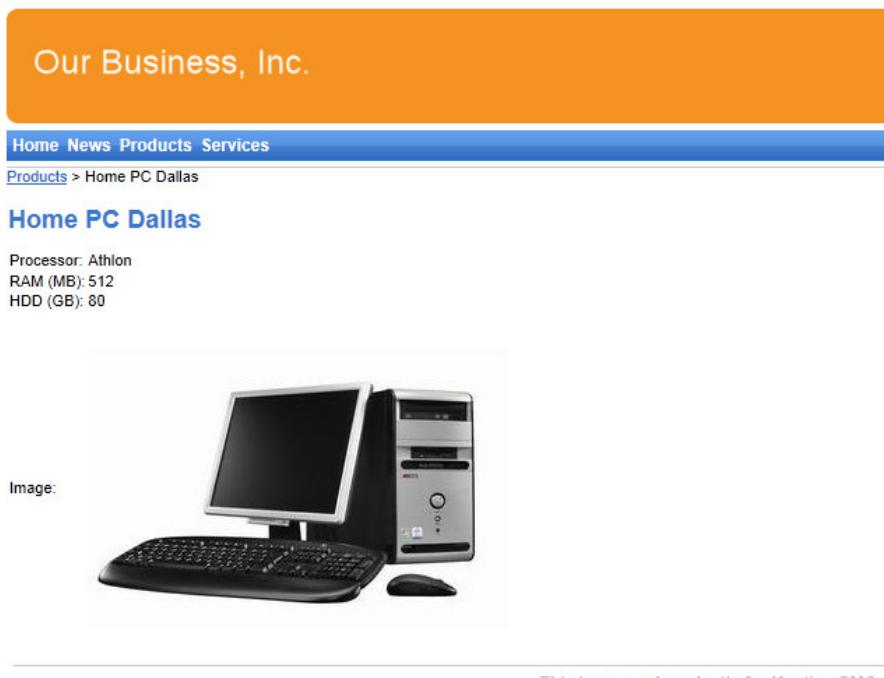
Now, when you click **/Products** you will see a page like this:



The screenshot shows a web page with a blue header bar containing the navigation menu: Home, News, Products, Services. Below the header, there is a section titled "Products" with two items listed: "Home PC Dallas" and "Office PC Houston". Each item has a thumbnail image of a computer setup (monitor, keyboard, mouse) and a small description below it.

This is a sample web site for Kentico CMS

When you click on some link, you will see computer details:



The screenshot shows a detailed view of the "Home PC Dallas" product. The top part of the page is identical to the previous screenshot, with the "Products" menu selected. Below the menu, the product title "Home PC Dallas" is displayed in bold. Underneath the title, the processor details are listed: Processor: Athlon, RAM (MB): 512, HDD (GB): 80. At the bottom of the page, there is a large image of the "Home PC Dallas" computer setup, which includes a monitor, a keyboard, a mouse, and a desktop tower.

This is a sample web site for Kentico CMS

You have learned how to define new document type and how to publish its documents on the website.

7.11 Search page

Kentico CMS allows you to perform index-based searching through all documents in the Kentico CMS repository. For this to work, you need to have a search index created. Go to **Site Manager -> Administration -> Smart search** and choose to create a **New index**. In the following dialog, enter the following details:

- **Display name:** My website
- **Code name:** MyWebSite
- **Index type:** Documents
- **Analyzer type:** Standard
- **Stop words:** Default
- **Assign index to website MyWebSite:** enabled

Click **OK**. You will be redirected to the index's editing interface. Switch to the **Index** tab and click **Add allowed content** (). In the following dialog, enter `/%` into the **Path** field and click **OK**. Switch to the **Sites** tab and make sure that the index is assigned to **My website**. Switch to the **Cultures** tab and choose the default culture of your site (typically English - United States).

Finally, switch to the **General** tab and **Rebuild** () the index. Once the index gets rebuilt, the documents on the site are prepared to be searched.



Searching through uploaded text files

It is possible to configure the CMS to search the text inside uploaded files, such as PDF, DOC or XLS documents. The configuration is described in **Developer's Guide -> Installation and deployment -> Additional configuration tasks -> Configuration of full-text search in files**. It's not necessary to configure it at this moment since we will only use the document content search.

Now go to Visual Studio and choose to create a new web form in the **CMS\Templates\MySite** folder. Name the page **SearchPage.aspx** and check the **Select master page** box. Choose the **CMS\Templates\MySite\MyMaster.master** page in the next dialog.

Add the following directive to the beginning of the page code:

```
<%@ Register src="~/CMSWebParts/SmartSearch/SearchDialogWithResults.ascx" tagname="SearchDialogWithResults" tagprefix="cms" %>
```

Now enter the following code inside the **<asp:content>** element of the newly created page:

```
<h1>Search</h1>  
  
<cms:SearchDialogWithResults ID="SearchDialogWithResults1" runat="server">
```

```
TransformationName="cms.root.smartsearchresultswithimages" Indexes="MyWebSite" />
```

This adds a header and the appropriate user control that will provide search functionality and display results. Notice that the code name of the previously created search index is specified as the value of the **Indexes** property of the user control.

Switch to the **code behind** and add a reference to the **CMS.UIControls** namespace:

[C#]

```
using CMS.UIControls;
```

[VB.NET]

```
Imports CMS.UIControls
```

You also need to change the class definition so that it inherits from the **TemplatePage** class:

[C#]

```
public partial class CMSTemplates_MySite_SearchPage : TemplatePage
```

[VB.NET]

```
Partial Class CMSTemplates_MySite_SearchPage  
    Inherits TemplatePage
```

Save the changes.

Now we need to register the new page template. Go to **Site Manager -> Development -> Page templates**, select the **My website** category and click **New template**. Create a new page template with the following values:

- **Template display name:** Search page
- **Template code name:** searchpage

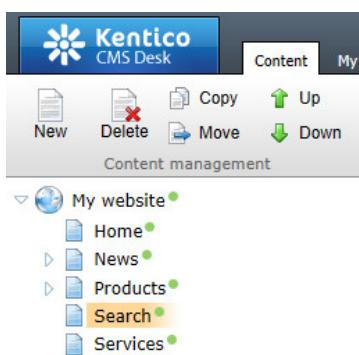
On the **General** tab of the page template, please set the following:

- **Template type:** ASPX page
- **File name:** ~/CMSTemplates/MySite/SearchPage.aspx

Click **Save** and switch to the **Sites** tab. Assign the new page template to **My website**.

Go to **CMS Desk -> Content**, select the root and click **New**. Choose to create a new **Page (menu)**

item) using the **My website/Search page** template and name the page **Search**. Click **Save**. Click the **Down** arrow in the main toolbar until you move the **Search** page to the end of the list.



Select the new **/Search** page and enter **PC** into the **Search for** box and click **Go**.

Our Business, Inc.

Home News Services Products Search

Search results

Search for: pc

Search mode: Any word

Search

 [Home PC Dallas](#)
http://127.0.0.1/KenticoCMS/Products/Home-PC-Dallas.aspx 12/16/2009 9:15:07 AM

 [Office PC Houston](#)
http://127.0.0.1/KenticoCMS/Products/Office-PC-Houston.aspx 12/16/2009 9:15:32 AM

This is a sample web site for Kentico CMS

When you click a search result, you will be redirected to the appropriate document.

Modifying the search results format

If you prefer a different design of the search results, you can modify the format by editing the **Site Manager > Development > Document types > Root > Transformations > SmartSearchResults** (or **SmartSearchResultsWithImages**) transformation.

7.12 Secured section for partners

Kentico CMS allows you to create secured site sections that can be accessed only by users who have a valid user name and password. We will create a simple page for partners that can be accessed only by registered users.

Go to **CMS Desk -> Content**, select the root and click **New**. Choose to create a new **Page (menu item)**. Enter **Partners** as the page name and use the **My website/Left menu with right text** page template. Click **Save**. Click the **Down** arrow in the main toolbar until you move the **Partners** page to the end of the list.

Click the **Page** tab and enter the following text: *This is a secured page for partners.*

Click **Save**.

Click **Properties -> Security**. In the **Access** section of the dialog, select **Yes** in the **Requires authentication** field and click **OK**. This will ensure that the page can be accessed only by authenticated users.

Now we need to create the logon page. Go to Visual Studio and choose to create a new web form in the **CMSTemplates\MySite** folder. Name the page *LogonPage.aspx* and check the **Select master page** box. Choose the **CMSTemplates\MySite\MyMaster.master** page in the next dialog.

Now enter the following HTML layout code inside the **<asp:content>** element of the new page:

```
<table border="0" width="100%">
  <tr valign="top">
    <td style="width:50%">
    </td>
    <td style="width:50%">
    </td>
  </tr>
</table>
```

Switch to the **Design** tab and drag and drop the **CMSWebParts/Membership/LogonForm.aspx** user control inside the left column and the **CMSWebParts/Membership/RegistrationForm.aspx** user control into the right column. Set their properties:

LogonForm1

- **AllowPasswordRetrieval:** true
- **SendEmailFrom:** <your e-mail address>

RegistrationForm1:

- **AssignRoles:** _notauthenticated_
- **EnableUserAfterRegistration:** true

Switch to the **code behind** and add a reference to the **CMS.UIControls** namespace:

[C#]

```
using CMS.UIControls;
```

[VB.NET]

```
Imports CMS.UIControls
```

You also need to change the class definition so that it inherits from the **TemplatePage** class:

[C#]

```
public partial class CMSTemplates_MySite_LogonPage : TemplatePage
```

[VB.NET]

```
Partial Class CMSTemplates_MySite_LogonPage  
    Inherits TemplatePage
```

Save the changes.

Now we need to register the new page template. Go to **Site Manager -> Development -> Page templates**, select the **My website** category and click **New template**. Create a new page template with the following values:

- **Template display name:** Logon page
- **Template code name:** LogonPage

On the **General** tab of the page template, please set the following:

- **Template type:** ASPX page
- **File name:** ~/CMSTemplates/MySite/LogonPage.aspx

Click **Save** and switch to the **Sites** tab. Assign the new page template to **My website**.

Now go to **CMS Desk -> Content**, select the root and click **New** and create a new **Folder**. Name the folder **Special pages**. Create a new page under this folder, name it **Logon** and use the **My website/ Logon page** template for it. Click **Save**.

Configuring the logon page

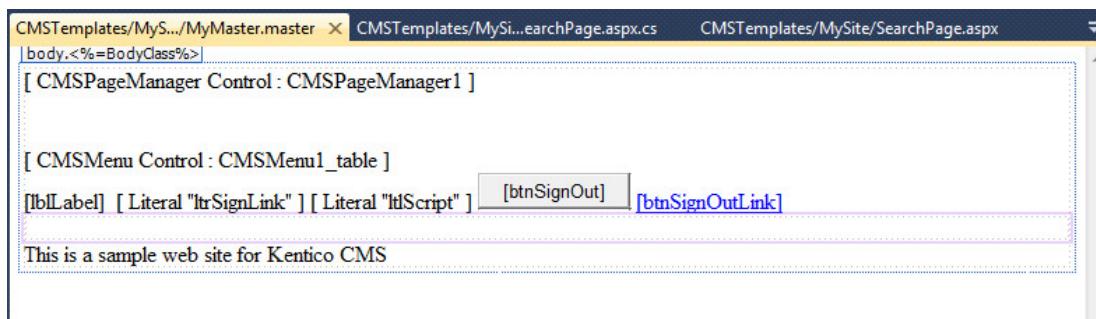
You will need to configure the system so that it uses the new logon page. Go to **Site Manager -> Settings**, choose **My website** in the drop-down list, click **Security & Membership**, clear the **Inherit from global settings** checkbox for the **Website logon page URL** field and set its value to **~/Special-pages/Logon.aspx**, which is the relative URL of the logon page (from the web application root). Click **Save**.

The screenshot shows the Kentico CMS 6.0 Site Manager interface. The top navigation bar includes links for CMS Desk, User (Global Administrator), v6.0.4281, Sign Out, and other options like Settings, Development, Tools, Dashboard, Licenses, and Support. On the left, a sidebar menu under the 'Site' dropdown for 'My website' lists various settings categories: Settings (Content, URLs and SEO, Security & Membership, System, On-line marketing, E-commerce, Community, Intranet & Collaboration, Versioning & Synchronization, Integration, Cloud services). The 'Security & Membership' option is selected and highlighted in orange. The main content area is titled 'Security & Membership' and contains several configuration sections:

- General**: Administrator's e-mail (admin@localhost.local) with 'Inherit from global settings' checked; Send membership reminder (days) set to 10 with 'Inherit from global settings' checked.
- Registrations**: Reserved user names (admin,root,administrator,sysadmin,sa) with 'Inherit from global settings' checked; Registration requires e-mail confirmation and Registration requires administrator's approval both have checkboxes checked; Delete non-activated user after (days) set to 5 with 'Inherit from global settings' checked; Require unique user e-mails checked with 'Inherit from global settings' checked.
- On-line users**: Update on-line users (minutes) set to 1 with 'Inherit from global settings' checked.
- Content**: Check page permissions (No page) with 'Inherit from global settings' unchecked; Website logon page URL (~~/Special-pages/Logon.aspx) with 'Inherit from global settings' unchecked; Access denied page URL (empty) with 'Inherit from global settings' checked.

Adding the Sign out button

Now we will add the "current user name" and "sign out" controls to our master page. Open the master page **MyMaster.master** in Visual Studio and view it on the **Design** tab. Drag and drop the **CurrentUser.ascx** and **SignOutButton.ascx** controls from the **CMSWebParts\Membership\Logon** folder in the Solution Explorer and place them under the **CMSMenu** control:



Set the following properties of these controls:

CurrentUser1:

- **ShowOnlyWhenAuthenticated**: true
- **CssClass**: CurrentUser

SignOutButton1:

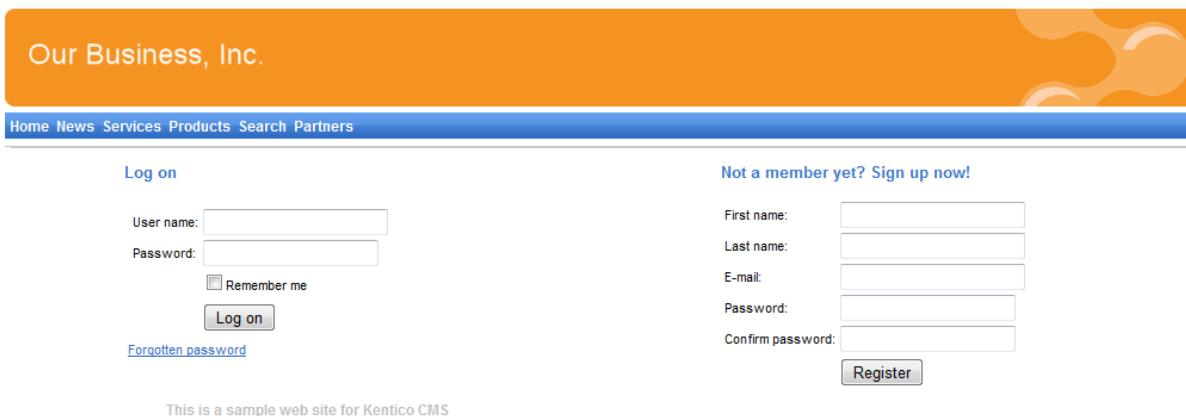
- **ShowOnlyWhenAuthenticated**: true

Save the changes.

Now go to **Site Manager -> Development -> CSS Stylesheets** and add the following code to the end of the stylesheet used by your site:

```
.CurrentUser  
{  
color: black;  
}
```

Sign out. Click **Partners** in the main menu. You will be redirected to the logon page:



Now you need either to sign in as the administrator or sign up and create a new account. After you sign in successfully, you will see the Partners page content together with the **Sign out** button and current user:



You have learned how to secure a part of the website so that it's only accessible by registered users.

Displaying personalized content based on a user's permissions

Kentico CMS also allows you to display personalized content based on the read permissions of users. You can e.g. grant the Read permission for the Gold partners section to gold partners and then only the gold partners will see the given menu item

and page content.

You can find more details on personalized content in **Developer's Guide -> Membership, permissions and security**.

You have just finished creating the sample website.

Part



VIII

Further steps

8 Further steps

8.1 Further steps

This is the end of the Kentico CMS Tutorial. If you need any further details, you will find them in [Kentico CMS Developer's Guide](#). It also covers other advanced topics, such as:

- Multi-lingual content
- Multi-site configuration
- Workflow and versioning
- Security administration
- Deployment to the live website
- Newsletters, Forms and other modules
- Kentico CMS API and extensibility
- and many other features.

If you cannot find the information that you require, please feel free to contact us at <http://www.kentico.com/Support.aspx>.

Index

- A -

- App themes 60
- ASPX page template
 - creating 42
 - editing in the portal engine 50
 - overview 40
 - registering the page as a template 45
 - web parts and widgets 50
 - writing the code 44

- C -

- Creating the CSS stylesheet 69
- CSS styles 58

- F -

- Further steps 114

- H -

- home page
 - editing content 21
 - template 77

- I -

- image
 - inserting 27
- installation
 - setup 8
 - web application 8

- K -

- Kentico CMS Overview 5

- L -

- link 28

- logon page
 - configuring 109

- M -

- Main menu 76
- master page
 - editing 72
 - using 48
- menu design
 - overview 61
- minimum requirements 7

- N -

- new page 23
- new site
 - using ASPX templates 64
 - wizard 64
- news item 31
- News page 85

- O -

- Overview 5

- P -

- prerequisites 7
- Products page
 - new document type 93
 - overview 93
 - page template 101
 - transformations 99

- S -

- Search page 105
- Secured section 108
- Services page 89
- Setup 12
- Sign out button
 - adding 110
- Site Development Overview 35
- Support 5, 114
- system requirements 7

- T -

Technical support 5, 114

- U -

User interface overview 19

- W -

web installer 8

web project

 configuring 71

widgets and web parts on ASPX templates 50