Kentico 2020 Beta Documentation

http://docs.kentico.com

Enabling alternative URLs for pages

Pages on Kentico MVC sites have their URLs determined in one of the following ways:

- Generated automatically by the system based on the page's position in the content tree (see Content tree-based routing)
- Manually by routes registered in the MVC application (together with matching URL patterns for page types in Kentico)

In addition to these main URLs, you can configure the system to support alternative URLs for pages. The feature allows individual pages to become available under any number of URLs. Depending on the site's configuration, the system either redirects to the page's main URL or displays the page content under the submitted alternative URL (URL rewrite).

Alternative URLs are managed directly in the administration interface, so they can be added or modified without changes in the application's routing code. This makes it possible for content editors and marketers to provide shorter and more relevant-looking links to your website's pages.

For example, in a scenario where a website displays an article page with the following URL:
<site domain>/Articles/Products/NewProducts

Marketers could add an alternative URL and make the article page available under a shorter URL like:
<site domain>/NewProducts

To set up alternative URL functionality for your website's pages, perform the following steps:

1. Enable the alternative URLs feature in your MVC application
2. Configure the alternative URLs editing interface and set guidelines for users
3. Set the alternative URLs mode (redirection or URL rewrite)

Enabling the alternative URLs feature

To allow alternative URLs for pages, you first need to enable the feature in the code of your MVC project:

1. Open your MVC project in Visual Studio.
2. Enable the alternative URLs feature at the start of your application's life cycle, for example in the **Application_Start** method of your project's **Global.asax** file.

   MVC projects created by the installer contain the *ApplicationConfig* class, whose *RegisterFeatures* method is called in the *Application_Start* method by default. You can use this class to encapsulate all of your *ApplicationBu ilder* code (enabling and configuring of Kentico MVC features).

   Note: The feature must be enabled **before** you register routes into the application's *RouteTable*. The *Kentico(). MapRoutes()* method adds routes that handle the alternative page URLs when the feature is enabled.

3. Prepare a **PageRoutingOptions** object with the **EnableAlternativeUrls** property set to *true*.
4. Call the **UsePageRouting** method of the *ApplicationBuilder* instance, with the *PageRoutingOptions* object as the parameter.

   Content tree-based routing

   Enabling alternative URLs also by default enables the Content tree-based routing functionality on the side of the MVC application. If you wish to enable only alternative URLs, set the **EnableRouting** property of the **PageRo utingOptions** object to *false*.

```
using Kentico.Content.Web.Mvc.Routing;
using Kentico.Web.Mvc;

...

protected void Application_Start()
{
    ...

    // Gets the ApplicationBuilder instance
    // Allows you to enable and configure selected Kentico MVC integration
features
    ApplicationBuilder builder = ApplicationBuilder.Current;

    // Enables the alternative URLs feature
    builder.UsePageRouting(new PageRoutingOptions
    {
        EnableAlternativeUrls = true
    });

    ...
}
```

The alternative URLs feature is now enabled. The *Kentico().MapRoutes()* method called in your MVC application's *RouteConfig* registers routes matching the alternative URLs that users set for pages in the Kentico administration interface.

To ensure that conflicts do not occur with your other routes, you need to configure restrictions and guidelines for alternative URLs (excluded URLs). The related settings for the alternative URLs editing interface are described in the following section.

Configuring the alternative URLs editing interface

After enabling alternative URLs in your MVC application, you need to set up the related editing interface for content editors and marketers in the Kentico administration interface.

Enabling the editing interface

1. Open the **Settings** application in the Kentico administration interface.
2. Navigate to the **URLs and SEO** category in the settings tree.
3. Under the **Alternative URLs (MVC)** section, select the **Enable editing interface** checkbox.
4. Click **Save**.

Users can now add and edit alternative URLs in the **Pages** application (after selecting a page in *Edit* mode, on the *Properties -> URLs* tab). See Managing page URLs for details.

The setting only controls the visibility of the editing interface. Any existing alternative URLs remain functional even if you disable the setting (as long as the alternative URLs feature is enabled in the related MVC application).

Additional requirements

- Alternative URLs are only available for pages whose page type has the **URL** feature enabled. Pages without a URL typically only store content and do not represent actual pages on the website.
- To work with alternative URLs, users need to have the **Manage alternative URLs (MVC)** permission for the **Cont ent** module.

Restricting alternative URLs

Giving unlimited control over URLs to end users is typically not desired (this could lead to broken URLs or conflicts). It is important to set guidelines that make it easy and safe for the website's users to add alternative URLs.

To configure alternative URL restrictions, use the following settings in the **URLs and SEO** category of the **Settings** application:

Setting	Description
Excluded alternative URLs	A list of text values specifying restricted alternative URL paths (without the prefix of the site's *Presentation URL*). If a user attempts to add an alternative URL matching one of the specified values, the system prevents the creation and displays the error message specified in the *Alternative URLs error message* setting. Every path value must be entered on a new line. The comparison is case-insensitive. If you need to exclude a large number of URL paths, you can add the asterisk ('*') character as a wildcard to the **start** or **end** of values. The wildcard represents any number of characters. Examples: • articles – the *articles* URL path • articles* – all URL paths that start with *articles* • articles/* – all URL paths whose first segment is *articles* • *articles – all URL paths that end with *articles* • *articles* – all URL paths that contain the word *articles* **Note**: • The system automatically excludes certain URLs paths, for example paths starting with *cms*, *getmedia* or *getfile*. Such URLs are reserved for page preview links, file requests, internal on-line marketing logging routes, etc. • We recommend excluding the URLs paths of any routes in your MVC application's routing table that are not used for the main URLs of content tree pages. Otherwise a conflicting alternative URL may override an important route on your website. • If you modify the setting, any existing alternative URLs that match the excluded values **remain in the system** (visible in the *Pages* application), but are **no longer handled** in the routing of the MVC application.
Alternative URLs constraint	A regular expression that restricts which alternative URLs are allowed. The expression is matched against the alternative URL path value (without the prefix of the site's *Presentation URL*). If a user attempts to add an alternative URL that does not match, the system prevents the creation and displays the error message specified in the *Alternative URLs error message* setting. Use the constraint to define which characters are allowed in alternative URLs or to enforce some type of structural pattern. By default, the setting contains a pattern that allows upper and lower case alphanumeric characters, underscores ('_'), hyphens ('-'), and forward slashes ('/'). The default constraint is represented by the following regular expression: ^[\w\-\/]+$ If you modify the setting, any existing alternative URLs that do not match the constraint **remain active** and **continue to be handled** in the routing of the MVC application. **Warning**: We do not recommend allowing characters that have specific meanings in URLs (such as '?' or '#'). This could cause users to submit values that result in invalid URLs or incorrect behavior.

Alternative URLs error message	The text displayed in the *Pages* application when a user attempts to add an alternative URL that either matches an excluded URL or does not fulfill the requirements of the URL constraint. We recommend writing a user-friendly message that describes how to add suitable alternative URLs according to the restrictions configured for your website.

Alternative URL validation and conflicts with the main URLs of pages

When users add alternative URLs in the *Pages* application, the system automatically prevents the creation of any URLs that would be in conflict with other alternative URLs or the main URLs of existing pages on the site.

If your system is configured to use custom routing based on the URL patterns of page types, URL conflicts can occur after creating a new page or editing an existing one, if the new main URL matches an existing alternative URL on the site. In these cases, the system does not stop the creation or update of the page. However, the *Pages* application displays a URL conflict warning when viewing such pages.

You can also keep track of URL conflicts by monitoring the event log. The system provides a site-specific scheduled task (*Validate alternative URLs*), which runs once per day by default and logs the following into the event log:

- Errors for any alternative URLs in conflict with the main URLs of pages on the site. For example, such conflicts can occur after updating a page type URL pattern or creating a new page. In this case, the **alternative URL overrides the main URL** in the routing of the MVC application.
- Errors for any alternative URLs that match the values of the *Excluded alternative URLs* setting. This can occur after changing the value of the setting. In this case, the alternative URL **remains in the system** (visible in the *Pages* application), but is **not handled in the routing** of the MVC application.
- Warnings for any alternative URL that do not match the *Alternative URLs constraint* setting. This can occur after changing the value of the setting. In this case, the alternative URL remains active and **continues to be handled in the routing** of the MVC application.

If your site's pages are under workflow, the main URLs of pages could be different between the published (live) and currently edited versions of pages. The *Validate alternative URLs* scheduled task evaluates the published versions of pages, while the validation rules and warnings in the *Pages* application check the latest edited versions.

Setting the alternative URLs mode

The system offers two ways of handling requests that target alternative page URLs:

- **Redirect** – the system performs a redirect (with the 301 HTTP status code) to the main URL of the given page.
- **Rewrite** – the alternative URL remains presented in the visitor's browser. The system rewrites the URL internally to the page's main URL, so that it can be served by the MVC application.

To configure the mode:

1. Open the **Settings** application in the Kentico administration interface.
2. Navigate to the **URLs and SEO** category in the settings tree.
3. Under the **Alternative URLs (MVC)** section, select the required option in the **Alternative URLs mode** setting.
4. Click **Save**.

The rewrite option is often preferred by content editors and marketers who want to present shorter or more relevant URLs to visitors. However, rewritten alternative URLs lead to duplicate content issues, which can have a negative impact on the site's search engine optimization (SEO). To prevent duplicate content issues with the rewrite option, we strongly recommend that you render pages with canonical link elements. See Providing canonical URLs for pages.

Providing canonical URLs for pages

If you use the *Rewrite* alternative URLs mode, we strongly recommend that you render pages with Canonical link elements. Canonical links specify the "main" or "preferred" URL for each page.

The Kentico API provides the **Kentico().PageMainUrl()** extension method, which can help you build the URL (*href* value) of canonical links.

- The method is available via the standard UrlHelper in Razor views.
- In cases where an alternative URL rewrite occurred, the method returns the **main URL** of the currently displayed page.
    - The main URL is either determined by content tree-based routing, or resolved for the specific page based on the page type's URL pattern.
    - The value is in the format of a virtual relative path (starting with ~/). You can use the UrlHelper.Content method to convert the virtual path to an application absolute path.
- If an alternative URL rewrite did not occur, the method returns an **empty string**.

We recommend adding a canonical link element to all pages, for example into your MVC website's main layout (and any other used layouts). This ensures that content editors can safely add alternative URLs for any page.

You can either add the *Url.Kentico().PageMainUrl()* method and other related code directly to your views, or prepare custom extension methods or HtmlHelpers that perform any logic required for your scenario.

The following basic example demonstrates how to add a canonical link element to a Razor view. The canonical URL is built in absolute form and works for all pages, based on either the displayed page's main URL (when handling requests targeting an alternative URL) or simply the current request's URI (for standard requests).

Example

```
@using System.Web.Mvc

@using Kentico.Content.Web.Mvc.Routing
@using Kentico.Web.Mvc

...

<head>
        ...
    @{
        string canonicalUrl;
        // Attempts to get the main URL of the displayed page
        string pageMainUrl = Url.Kentico().PageMainUrl();
        // Checks whether an alternative URL rewrite occurred for the displayed page
        if (!String.IsNullOrEmpty(pageMainUrl))
        {
            // Sets the canonical URL to the displayed page's main URL in absolute
form,
            // using the current request's scheme and authority
            string requestAuthority = Url.RequestContext.HttpContext.Request.Url.
GetLeftPart(UriPartial.Authority);
            canonicalUrl = requestAuthority + Url.Content(pageMainUrl);
        }
        // If an alternative URL rewrite did not occur, the page main URL is empty
        else
        {
            // Sets the canonical URL to the current request's URI (without the
fragment or query string parameters)
            canonicalUrl = Url.RequestContext.HttpContext.Request.Url.GetLeftPart
(UriPartial.Path);
        }
    }

    @* Renders a canonical link element *@
    <link rel="canonical" href="@canonicalUrl" />
</head>
```