

Transformations are a crucial part of the process used to display pages and other data in Kentico. A transformation is a code template that determines how listing web parts and controls render content. Data sources provide a collection of items containing raw data, and transformations convert the data into appropriately formatted page output code (HTML).

The functionality of transformations is very similar to that of item templates used by standard ASP.NET list controls, such as the Repeater. The main difference is that the system stores transformations as virtual objects in the database, and you can easily reuse transformations for different components.

Transformations are categorized under the objects whose data they display. You can manage transformations for:

- [Page types](#) (**Page types -> Edit a page type -> Transformations**)
 - [Container page types](#) do not represent actual objects on the website, but only serve as containers for transformations. Use container page types to store transformations that you want to share among multiple page types.
- [Custom tables](#) (**Custom tables -> Edit a table -> Transformations**)

To assign transformations to listing web parts or controls, use the available **Transformation** properties. Transformations are supported by all data listing web parts, as well as by [listing controls](#) that are designed to work with Kentico pages. The full name that identifies transformations is in format **<parent object code name>.<transformation name>**.





Note that transformation names can not contain the dot '.' character. Including dots in transformation names could cause unwanted system behavior.

The Kentico sample sites include many transformations for all built-in page types.

Transformation types

You can use several different approaches when writing the code of transformations. The following transformation types are available:

Transformation type	Description
ASCX	<p>The code of the transformation supports ASCX markup, i.e. the same syntax that you would use to edit standard web forms or user controls. This includes:</p> <ul style="list-style-type: none"> • Embedded controls (user controls or server controls, see the Transformation example) • Inline code (Kentico API, standard ASP.NET data binding expressions, special methods designed for use in transformations) <p>To access the fields of the transformed page or custom table, use data binding expressions in format: <code><%# Eval("ColumnName") %></code></p> <div style="background-color: #fff9c4; padding: 10px; margin-top: 10px;"> <p> Notes</p> <ul style="list-style-type: none"> • For security reasons, only users who have the Edit ASCX code permission for the Design module may edit the code of ASCX transformations. This permission can only be assigned by users with the Global administrator privilege level. • ASCX transformations need to be compiled, so you cannot create or edit them on precompiled applications. • ASCX transformations do NOT support macro expressions. • We do NOT recommend using web parts within the code of transformations. This could lead to poor performance and certain web parts may not work correctly within transformations due to their life cycle. </div>

Text/XML	<p>The system processes the code of <i>Text/XML</i> transformations as basic HTML. ASCX markup (controls or inline code) does not work. You can use Kentico macro expressions and methods to insert dynamic values into the content.</p> <p>To access the fields of the transformed page, custom table item or object, use expressions in format: <code>{% ColumnName %}</code></p> <div style="background-color: #e1f5fe; padding: 10px; border: 1px solid #bbdefb;"> <p> Tip: If a field of the transformed data object contains characters that conflict with the macro syntax (for example when using an XML data source), use the <i>DataItem</i> macro object and indexing to access the field value.</p> <p>For example: <code>{% DataItem["column-name"] %}</code></p> </div> <p>Text transformations have the following advantages:</p> <ul style="list-style-type: none"> • No compilation required. You can create and edit text transformations even in environments where compilation of virtual objects is not possible, for example on precompiled websites. • Text transformations do not allow users to execute inline code, so they cannot be used to compromise the security of the website.
HTML	<p>Work the same way as Text/XML transformations, but you edit the content through the WYSIWYG editor. The editor shows the rendered output of the transformation's HTML code.</p>
XSLT	<p>Use XSL elements to render the data. The code must be in valid XML format.</p> <p>When creating XSLT transformations for displaying pages, set the match attribute of the <code><xsl:template></code> element to Table.</p> <pre style="border: 1px dashed #ccc; padding: 10px;"> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"> <xsl:output method="xml" omit-xml-declaration="yes" /> <xsl:template match="Table"> ... </xsl:template> </xsl:stylesheet> </pre>
jQuery	<p>Provide a way to define the jQuery templates used by:</p> <ul style="list-style-type: none"> • The Chat application • Integrated Strands recommendations

Transformation example

The following is the code of the **CorporateSite.Transformations.ProductList** ASCX transformation, which displays lists of products on the sample Corporate Site:

**ASCX transformation code**

```

<%@ Register Src="~/CMSModules/Ecommerce/Controls/ProductOptions
/ShoppingCartItemSelector.ascx" TagName="CartItemSelector" TagPrefix="uc1" %>
<div class="ProductPreview">
    <div class="ProductBox">
        ##editbuttons##
        <div class="ProductImage">
            <a href="<## GetDocumentUrl() %>" style="display:block;">
                " />
            </a>
        </div>
        <a href="<## GetDocumentUrl() %>">
            <span class="ProductTitle textContent">
                <## EvalText("SKUName", true) %>
            </span>
        </a>
        <div class="ProductFooter">
            <div class="productPrice"><## GetSKUFormattedPrice() %></div>
            <uc1:CartItemSelector id="cartItemSelector" runat="server" SKUID='<##
EvalInteger("SKUID") %>' SKUEnabled='<## EvalBool("SKUEnabled") %>'
AddToCartImageButton="~/App_Themes/CorporateSite/Images/btn_addToShoppingCart.png" />
        </div>
    </div>
</div>

```

When assigned to a listing web part or control that has products (SKUs) in its data source, the output HTML code of individual products contains values returned by the methods and data binding expressions, like in the following example (part of the code is omitted to save space):

HTML output

```

<div class="ProductPreview">
    <div class="ProductBox">
        <div class="ProductImage">
            <a href="/8.0_4822.32562/Products/Smartphones/Apple-iPhone-4.aspx" style="
display:block;">
                
            </a>
        </div>
        <a href="/8.0_4822.32562/Products/Smartphones/Apple-iPhone-4.aspx">
            <span class="ProductTitle textContent">
                Apple iPhone 4 with inscription
            </span>
        </a>
        <div class="ProductFooter">
            <div class="productPrice">$759.99</div>
            ...
        </div>
    </div>
</div>

```

The final output of the product on the website then looks like this:



Adding CSS styles to transformations

You can define the CSS classes used in transformation code either in the [stylesheets of the website](#) or individual pages, or add them directly to the transformation object (we only recommend this approach for testing purposes).

To add styles to transformations:

1. Edit the transformation on the **General** tab.
2. Click **Add CSS styles** below the code editor. The **CSS styles** field appears, where you can add any required CSS classes.
3. Click **Save**.

If the styles require any additional files (such as images), you can add them on the **Theme** tab.

For more information about page component CSS styles, see [Adding CSS to page components](#).