



[Output caching](#) is a powerful tool for improving page performance. However, some pages have sections of content that need to be dynamic. Substitution macros allow you to add variables onto pages, which the system dynamically resolves even when loading the content from the output cache. By using substitutions, you can cache the output of pages, but still keep pieces of content that update according to the current time or other context information.

By default, the system does not provide any substitution macros. Developers need to define the substitutions according to the requirements of your website.

To insert output cache substitutions into page content, write expressions in format: **{~SubstitutionName~}**

 **Note:** Substitution macros work even on pages that do not use output caching.

 **Tip:** You do not need to create substitution macros for all types of dynamic content. By default, the output cache stores multiple versions of pages based on variables such as the user name, language and browser type. The system serves the appropriate cache version to visitors according to these variables.

See the [Caching multiple output versions of pages](#) section for more information.

## Defining substitution macros


The system resolves substitution macros by calling the methods registered as handlers for the **ResponseOutputFilter.OnResolveSubstitution** event. To define substitutions, implement a handler method that converts individual substitution expressions into the appropriate results.

You need to register the event handlers at the beginning of the application's life cycle – create a [custom module class](#) and override the module's **OnInit** method.

### Example

The following example demonstrates how to create a substitution macro that displays the current time:

1. Open your Kentico web project in Visual Studio (using the **WebSite.sln** or **WebApp.sln** file).
2. Create a [custom module class](#).
  - Either add the class into a custom project within the Kentico solution (recommended) or directly into the Kentico web project (into a custom folder under the **CMSApp** project for *web application* installations, into the **App\_Code** folder for *web site* installations).

 For basic execution of initialization code, you only need to register a "code-only" module through the API. You do NOT need to create a new module within the **Modules** application in the Kentico administration interface.

3. Override the module's **OnInit** method and assign a handler method to the **ResponseOutputFilter.OnResolveSubstitution** event.

```
using System;

using CMS;
using CMS.DataEngine;
using CMS.OutputFilter;

// Registers the custom module into the system
[assembly: RegisterModule(typeof(CustomCacheSubstitutionModule))]

public class CustomCacheSubstitutionModule : Module
{
    // Module class constructor, the system registers the module under the
```



```

name "CustomCacheSubstitutions"
    public CustomCacheSubstitutionModule()
        : base("CustomCacheSubstitutions")
    {
    }

    // Contains initialization code that is executed when the application
starts
    protected override void OnInit()
    {
        base.OnInit();

        // Assigns a handler method to the event that the system triggers
when resolving substitution macros
        ResponseOutputFilter.OnResolveSubstitution +=
ResolveCustomOutputSubstitutions;
    }

    /// <summary>
    /// Resolves output substitutions.
    /// </summary>
    /// <param name="sender">Sender object</param>
    /// <param name="e">Event argument object representing the substitution
expression that is being resolved</param>
    private void ResolveCustomOutputSubstitutions(object sender,
SubstitutionEventArgs e)
    {
        // Checks that the substitution expression is not resolved yet
        if (!e.Match)
        {
            // Branches according to the expression text of the
substitution macro
            // You can define any number of substitution macros by
adding further cases
            switch (e.Expression.ToLower())
            {
                // Handles the {~TimeStamp~} substitution macro
                case "timestamp":

                    // Flags the substitution expression as
resolved
                    e.Match = true;

                    // Returns the current time as the result
of the substitution
                    e.Result = DateTime.Now.ToString();
                    break;
            }
        }
    }
}

```

#### 4. Save the class file.

The system now recognizes the `{~TimeStamp~}` expression, and resolves it into the current time. You can place the expression directly into the text content of pages, or anywhere within the output code.



### Output substitutions vs. Macro expressions

The functionality of output substitutions is similar to Kentico [macro expressions](#). The main difference is that the system resolves substitutions even when loading pages from the [output cache](#).

For example, the `{% DateTime.Now %}` macro also displays the current time. If you place both the macro and the sample `{~TimeStamp~}` substitution onto a page that uses output caching, you get the following results:

- **Macro** – displays the time when the page was loaded for the first time and saved into the output cache.
- **Substitution** – updates the time whenever the visitor refreshes the page, even if the system loads the content from the output cache.