Macro namespaces serve as containers for static macro methods and fields. Users can access the members of namespaces when writing macro expressions, for example *{% Math.Pi %}* or *{% Math.Log(x) %}*. Namespaces also appear in the macro autocomplete help. The system uses several default namespaces such as *Math*, *String* or *Util*, and you can create your own namespaces for custom macros.

To add a custom macro namespace:

1. Create a class inheriting from **MacroNamespace<Namespace type>**. In *web site* projects, you can either add the class as part of a custom assembly (recommended) or into the **App_Code** folder.
2. Register macro fields or methods into the namespace – add **Extension** attributes to the class, with the types of the appropriate container classes as parameters.

```
using CMS.Base;
using CMS.MacroEngine;

[Extension(typeof(CustomMacroFields))]
[Extension(typeof(CustomMacroMethods))]
public class CustomMacroNamespace : MacroNamespace<CustomMacroNamespace>
{
}
```

See Registering custom macro methods and Adding custom macro fields to learn about creating container classes for macro fields and methods.

## Registering macro namespaces

Once you have defined the macro namespace class, you need to register the namespace as a source into a macro resolver (typically the global resolver).

We recommend registering your macro namespaces at the beginning of the application's life cycle (during initialization). The following steps describe how to register a macro namespace into the global resolver:

1. 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.

2. Override the module's **OnInit** method.
3. Call the **SetNamedSourceData** method for the global resolver with the following parameters:
   - A string that sets the visible name of the namespace (used in macro syntax).
   - An instance of your macro namespace class.
   - (Optional) By default, the registered namespace appears in the high priority section of the autocomplete help and macro tree. To add namespaces with normal priority, add **false** as the third parameter.

```
using CMS;
using CMS.Base;
using CMS.DataEngine;
using CMS.MacroEngine;

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

public class CustomMacroModule : Module
{
        // Module class constructor, the system registers the module under the name
"CustomMacros"
        public CustomMacroModule()
                : base("CustomMacros")
        {
        }

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

                // Registers "CustomNamespace" into the macro engine
                MacroContext.GlobalResolver.SetNamedSourceData("CustomNamespace",
CustomMacroNamespace.Instance);
        }
}
```

The system registers your custom macro namespace when the application starts. Users can access the namespace's members when writing macro expressions.

### Registering namespaces as anonymous sources

By registering a macro namespace as an anonymous source, you can allow users to access the namespace's members directly without writing the namespace as a prefix. For example, *{% Field %}* instead of *{% Namespace.Field %}*.

```
// Registers "CustomNamespace" as an anonymous macro source
MacroContext.GlobalResolver.AddAnonymousSourceData(CustomMacroNamespace.Instance);
```

You can register the same namespace as both a named and anonymous source. If you only register a namespace as an anonymous source, users cannot access the members using the prefix notation, and the namespace does not appear in the macro autocomplete help.

> **Note**: Data items registered through anonymous macro sources do NOT appear in the macro autocomplete help. As a result, the autocomplete help only displays namespace members when using the prefix notation, even when the namespace is registered as both a named and anonymous source.