You can modify the behavior of the Kentico application by registering a custom module and executing code during its initialization:

Tip: If you only wish to run custom initialization code without developing a complete module, you can register "code-only" modules. You do not need to create a matching module object within the **Modules** application in the Kentico administration interface.

1. Open your project in Visual Studio (using the **WebSite.sln** or **WebApp.sln** file).
2. Create a class in the module's code folder, for example *~/App_Code/CMSModules/CompanyOverview*.

   If your module uses a separate assembly, add the module class into the corresponding project.

   Keep in mind that the code of your custom assembly project must contain the **[assembly: AssemblyDiscoverable]** attribute (the recommended location is in the *Properties/AssemblyInfo.cs* file). To use the attribute, your assembly's project must contain a reference to the *CMS.Core.dll* library.

3. Make the module class inherit from **CMS.DataEngine.Module**.
4. Define the constructor of the module class:
   - Inherit from the base constructor
   - Enter the code name of the module as the base constructor parameter (for code-only modules, the value can be any string except for the names of existing modules)
   - If you are developing a module with installation package support, you need to set the optional second parameter *(isInstallable)* to ***true***
5. Register the module class using the **RegisterModule** assembly attribute.
6. Implement your custom functionality inside the module class.

You can achieve most customizations by running code during the initialization of the module – override the following methods:

- **OnInit (recommended)** - the system executes the code during the initialization (start) of the application. A typical example of *OnInit* code is assigning handler methods to system events.
- **OnPreInit** - the system executes the code before *OnInit.* Does *not* support any operations that require access to the database, such as working with the data of modules. For example, you can use *OnPreInit* to register custom implementations of interfaces.

Because you cannot manually set the initialization order of modules or define dependencies between modules, we do not recommend working with the data of other modules directly inside the *OnInit* method. The best approach is to assign handlers to system events, and perform the actual operations inside the handler methods. For general code that is not related to a specific system event, you can use the **ApplicationEvents.Initialized.Execute** event, which occurs after all modules in the system are initialized.

For example, the following code extends the sample *Company overview* module from the Creating custom modules page. The example uses event handling to log an entry in the system's Event log whenever a new office is created.

Example

```
using CMS;
using CMS.DataEngine;
using CMS.EventLog;

[assembly: RegisterModule(typeof(CompanyOverviewModule))]

public class CompanyOverviewModule : Module
{
        // Module class constructor, inherits from the base constructor with the code
name of the module as the parameter
        public CompanyOverviewModule()
                : base("CompanyOverview")
        {
        }

        // Initializes the module. Called when the application starts.
        protected override void OnInit()
        {
                base.OnInit();

                // Assigns a handler to the Insert.After event for OfficeInfo objects
                CompanyOverview.OfficeInfo.TYPEINFO.Events.Insert.After +=
Office_InsertAfter;
        }

        private void Office_InsertAfter(object sender, ObjectEventArgs e)
        {
                // Logs an information entry into the system's event log whenever a
new office is created
                string message = "New office '" + e.Object.GetStringValue
("OfficeDisplayName", "") + "' was created in the custom Company overview module.";
                EventLogProvider.LogInformation("Company overview module", "NEW
OFFICE", message);
        }
}
```