Web farm tasks are code executed by web farm servers. Create custom web farm tasks if you extended Kentico with functions that work directly with the application's memory or file storages. For example, if your custom function writes files to the file system and you want these files to be synchronized across all servers in your web farm, you need to write a custom web farm task that handles the synchronization.

Example

The following example shows how to create a custom synchronization task that logs information events into the event log of all servers in the web farm:

1. Open the 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. Add a method inside the module class, for example *RegisterCustomTask*.
4. Override the module's **OnInit** method and call the *RegisterCustomTask* method.

```
using System;

using CMS;
using CMS.Base;
using CMS.DataEngine;
using CMS.EventLog;
using CMS.Core;
using CMS.Helpers;

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

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

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

                RegisterCustomTask();
        }

        // Registers a custom web farm synchronization task in the system
        public void RegisterCustomTask()
        {
                // Creates a custom web farm task which logs information to the
event log
                WebFarmTask task = new WebFarmTask
                {
                        Type = "CustomTask",

                        // Logs a record into the system's event log, with
'Execute' as the event code
                        Execute = (target, textData, binaryData) =>
                        {
                                string message = String.Format("Slave {0} is
going to process task from master {1}", SystemContext.ServerName, textData);
                                EventLogProvider.LogInformation(target,
"Execute", message);

                                // TODO: Slave operation
                        }
                };

                // Registers the given web farm task
                WebFarmHelper.RegisterTask(task);
        }
}
```

5. Deploy these code changes to all of your web farm servers.

Each server in the web farm can now process the custom tasks.

To create the custom tasks, call the **WebFarmHelper.CreateTask** method anywhere in your custom code. For example:

1. Add a *CreateCustomTask* method into the module class that handles the task registration:

```
// Creates a custom web farm synchronization task
public void CreateCustomTask()
{
        // TODO: Master operation

        // Prepares the data parameter of the custom task
        string data = SystemContext.ServerName;

        // Creates the custom web farm synchronization task
        if (WebFarmHelper.CreateTask("CustomTask", "MyTask", data, null))
        {
                // Runs if the custom task was created successfully
                // Logs a record in the event log on the server that created the
web farm task
                string message = String.Format("Master {0} finished processing
operation and created task for slaves", SystemContext.ServerName);
                EventLogProvider.LogInformation("CustomTask", "Create", message);
        }
}
```

2. Call *CreateCustomTask()* within the module's **OnInit** method:

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

        RegisterCustomTask();

        CreateCustomTask();
}
```

The instance where you added the code now always creates the custom task during initialization (application start). If the task is successfully created, the system also logs a record into the event log, with *Create* as the event code. You can see the results by checking the event log in the **Event log** application.

You need to have web farms properly configured and working (have more than one server in the web farm) to try out this example.