

Kentico allows you to modify the behavior of file system providers. You can use different file system providers to access different parts of the application's file repository, and you can change the default locations of different types of files.

See the following use cases:

- [Using different file system providers for specific parts of the application's file repository](#)
- [Changing content file paths to a different disk drive](#)
- [Changing content file paths to a shared storage](#)

## Using different file system providers for specific parts of the application's file repository

The CMS.IO API allows you to access different sections of the application's file repository using different [file system providers](#). For example, you can store media files in the Azure blob storage, while all other files stay in the default Windows file system.

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 in 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 perform the following:
  - Create a new instance of the *StorageProvider* class.
  - Map a directory to the provider.

```
using CMS;
using CMS.Base;
using CMS.DataEngine;
using CMS.IO;


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

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

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

        // Creates a new StorageProvider instance (Azure storage provider in this
        example)
        var mediaProvider = StorageProvider.CreateAzureStorageProvider();

        // Maps a directory to the provider
        StorageHelper.MapStoragePath("~/MySite/Media/", mediaProvider);
    }
}
```

 To create *StorageProvider* instances for the default file storage providers, call the following static methods of the *StorageProvider* class:

- **CreateFilesystemStorageProvider** – returns instance of the default provider for the Windows file system. Has an optional *bool* parameter that specifies whether the storage is shared.
- **CreateAzureStorageProvider** – returns an instance of the default provider for the Microsoft Azure Blob storage.
- **CreateAmazonStorageProvider** – returns an instance of the default provider for the Amazon S3 storage service.

4. Save the file. If your project is installed in the web application format, rebuild the solution.

The application now uses the given file system provider to access the specified project folder.

## Changing content file paths to a different disk drive

CMS.IO allows you to change default file paths. For example, when using the Windows file system, you can store media files on a different disk drive.

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 in the **CMSApp** project for *web application* installations, into the **App\_Code** folder for *web site* installations).

**i** 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 perform the following:

- Create a new instance of the **StorageProvider** class.
- Specify the target directory via the **CustomRootPath** property of the provider. This sets the **root** of the target path – the provider creates the relative path of the mapped folders within the given root directory.
- Map a directory to the provider.

```
using CMS;
using CMS.Base;
using CMS.DataEngine;
using CMS.IO;

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

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

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

        // Creates a new StorageProvider instance for the Windows file
        system
        var mediaProvider = StorageProvider.
        CreateFileSystemStorageProvider();

        // Specifies the target root directory. The provider creates the
        relative path of the mapped folders within the given directory.
        mediaProvider.CustomRootPath = @"D:\CustomMediaRoot";

        // Maps a directory to the provider
        StorageHelper.MapStoragePath("~/MySite/Media/", mediaProvider);
    }
}
```

**i** To create a *StorageProvider* instance for the Windows file system, call the **CreateFileSystemStorageProvider** static method of the *StorageProvider* class.

4. Save the file. If your project is installed in the web application format, rebuild the solution.

This sample code ensures that every time the application requests or creates a file in the **~/MySite/Media** directory, the location specified in the **CustomRootPath** property is used instead. In this example, the target folder is **D:\CustomMediaRoot\MySite\Media**.

## Changing content file paths to a shared storage

The system allows you to change default file paths to a shared storage directory on a specific server. This can be particularly useful if you are running a [web farm](#). This way, you can for example map media files to a folder shared among all web farm servers, and thus eliminate the need to synchronize these files among the servers.

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 in 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 perform the following:
  - Create a new instance of the **StorageProvider** class (the parameter of the *CreateFileSystemStorageProvider* method must be *true*).
  - Specify the target directory via the **CustomRootPath** property of the provider. This sets the **root** of the target path – the provider creates the relative path of the mapped folders within the given root directory.
  - Map a directory from your site to the provider.



```
using CMS;
using CMS.Base;
using CMS.DataEngine;
using CMS.IO;

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

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

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

        // Creates a new StorageProvider instance for the Windows file
        system
        var sharedMediaProvider = StorageProvider.
        CreateFileSystemStorageProvider(isSharedStorage : true);

        // Specifies the target root directory. The provider creates the
        relative path of the mapped folders within the given directory.
        sharedMediaProvider.CustomRootPath = @"\\Server\CustomMediaRoot";

        // Maps a directory from your website to the provider
        StorageHelper.MapStoragePath("~/MySite/Media/",
        sharedMediaProvider);
    }
}
```



To create a *StorageProvider* instance for the Windows file system, call the **CreateFileSystemStorageProvider** static method of the *StorageProvider* class.

The value of the method's *bool* parameter specifies whether the storage is shared among web farm servers.

4. Save the file. If your project is installed in the web application format, rebuild the solution.

Now every time the application requests or creates a file in the **~/MySite/Media** directory, the location specified in the **CustomRootPath** property is used instead. In this example, the target folder is **\\Server\CustomMediaRoot\MySite\Media**.

[Web farm servers](#) will not attempt to synchronize media files and will instead use the files in the shared storage.