This page describes how to configure external ASP.NET applications so that they support the Kentico API. Using Kentico functionality externally allows you to create scenarios such as:

- Separate applications integrated with your main Kentico website
- Custom websites or applications that use Kentico as a content repository

Start Visual Studio and open your external application's project. You need to perform the following steps:

1. Connect to the Kentico database
2. Integrate the Kentico API libraries
3. Initialize the Kentico application

License requirements

The Kentico API performs license checks even when running in an external application. For applications that run under a different domain than your main Kentico site (web applications, WCF services, etc.), you need to have a valid license for the given domain.

Add the license via the administration interface of the connected Kentico application – the license is available through the shared database.

Connecting to the Kentico database

To be able to access the Kentico database, you need to specify the appropriate connection string in your application's **web.config** or **app.config** file.

Add the connection string into the *configuration/connectionStrings* section using an *<add>* element named **CMSConnectionString**.

```
 <configuration>
  <connectionStrings>

    ...

    <add name="CMSConnectionString" connectionString="Persist Security Info=False;
database=Kentico;server=myserver;user id=username;password=mypassword;Current
Language=English;Connection Timeout=120;" />

  </connectionStrings>
```

Tip: We recommend copying the exact connection string from the web.config file of your Kentico web project.

Integrating Kentico API libraries

Before you can use Kentico functionality in your application, you need to add the libraries containing the required code:

1. Right-click your application's project in the Visual Studio **Solution Explorer** and select **Manage NuGet Packages**.
2. Search for the **Kentico.Libraries** package.
3. Install the *Kentico.Libraries* version that matches the version of the connected Kentico database.
4. (Optional) If you wish to use Kentico controls or web form API in external web applications, you need to install the **Kentico.Libraries.Web.UI** package.

Maintaining Kentico.Libraries

If you upgrade or hotfix the related Kentico project and its database to a newer version, you also need to update the Kentico NuGet packages to a matching version in your external application.

Initializing the Kentico application

You need to initialize Kentico before making calls to its API from an external project.

Web applications

1. Edit your application's Global.asax file (create the file if necessary).
2. Execute the **CMSApplication.Init** method in the **Application_BeginRequest** method:

```
void Application_BeginRequest(object sender, EventArgs e)
{
        CMS.DataEngine.CMSApplication.Init();
}
```

If you wish to load and display content from the Kentico database in an external web application, you need to perform additional steps. See Displaying Kentico content in external applications.

Non-web applications (for example Console applications)

1. Edit your application's **Main** method.
2. Execute the **CMSApplication.Init** method at any point before the first call of the Kentico API.

You can now use the Kentico API in your external application. The Kentico API allows you to perform any action available in the standard administration interface.

Using custom code that requires discovery

For external projects that compile into a different output type than a DLL assembly (for example Console applications), do NOT directly add any custom classes that need to be detected during the API initialization. This includes classes registered using attributes, such as generated wrapper classes for objects, custom module classes, custom implementations of interfaces or providers, etc. The system is only able to discover such code within assemblies.

Instead, add the code into a *Class Library* project with the *AssemblyDiscoverable* attribute and reference the assembly from your project.

WCF services

If implementing custom WCF services within the Kentico web project, you need to initialize the Kentico application before calling the Kentico API within the service code.

Otherwise you may encounter errors if the first request that starts the application is destined for the WCF service. The standard initialization does not occur, because WCF requests are not processed by the ASP.NET HTTP runtime. For more information, see the WCF Services and ASP.NET article.

For example, you can execute the **CMSApplication.Init** method within the constructor of your WCF service class.

Additional configuration and tips

Sharing project files with the Kentico application

If you need to call API that works with a specific folder in the Kentico project, you can map the physical path used by the Kentico API in your external application to a specific Kentico project. For example, you can use this approach if you wish to externally update or rebuild the files of locally stored search indexes.

Set the **SystemContext.WebApplicationPhysicalPath** property to the path of your Kentico project's **CMS** folder. Map the path *before* you initialize the Kentico API (i.e. call *CMSApplication.Init*), for example in the **Application_Start** method of *Global.asax,* or in your application's **Main** method.

Example

```
// Maps the physical path used by the Kentico API in the external application to the
folder of the connected Kentico project
CMS.Base.SystemContext.WebApplicationPhysicalPath = "C:
\\inetpub\\wwwroot\\Kentico\\CMS";
```

Working with the user context

When working with the Kentico API in an external application, the default Kentico user context is not available. The standard API for getting the current user (***CMS.Membership.MembershipContext.AuthenticatedUser***) always returns the "public" user when called externally.

To use API that relies on the context of a specific user, enclose the code into a **using** statement with a declared **CMSActionContext** instance and the appropriate *UserInfo* object as a parameter.

Example

```
using CMS.Base;
using CMS.Membership;
using CMS.EventLog;

...

// Gets an object representing a specific Kentico user
UserInfo user = UserInfoProvider.GetUserInfo("Andy");

// Sets the context of the user
using (new CMSActionContext(user))
{
        // Logs an information event into the Kentico event log, with "Andy" as the
user who caused the event
        EventLogProvider.LogEvent(EventType.INFORMATION, "External Application",
"Event_Code", "Details");
}
```

Disabling the Kentico Virtual path provider

If you have the **Kentico.Libraries.Web.UI** package installed, Kentico uses a virtual path provider to load virtual objects, such as ASCX transformations or page layouts, from the database. When using the Kentico API externally in web applications, you may wish to disable the virtual path provider for the following reasons:

- To avoid conflicts if your application has its own virtual path provider implementation

- If your external application does not use Kentico virtual objects at all (disabling the virtual path provider reduces overhead)

To disable the Kentico virtual path provider:

1. Edit your external application's **web.config** or **app.config** file.
2. Add the **<appSettings>** element under the *<configuration>* section.
3. Add the **CMSUseVirtualPathProvider** key and set the value to *false*.

```
<configuration>

...

        <appSettings>
                <add key="CMSUseVirtualPathProvider" value="false" />
        </appSettings>

...

</configuration>
```

Note: Without the virtual path provider, you cannot use ASCX transformations (or other Kentico virtual objects) to display data in external web applications. See Displaying Kentico content in external applications for more information and workarounds.