This page contains best practices for customizing Kentico. The recommendations mostly focus on minimizing problems that can occur when upgrading to newer versions.

> If you are unfamiliar with the upgrade process, see Upgrading to Kentico 11.

Database structure and objects

> **Note**: In general, upgrades may change the structure of the default Kentico database tables and overwrite any default database objects.

Customizing the default Kentico database tables

- NEVER modify Kentico tables manually in the database (adding or removing of columns, changing column data types, etc.).

- Adding of custom columns is only possible for tables that store the data of **customizable** classes:
  1. Open the **Modules** application in the Kentico administration interface.
  2. Edit the appropriate module.
  3. Edit the **customizable** class on the **Classes** tab.
  4. Define any new database columns on the **Fields** tab. You cannot remove the default fields or change their database definitions (column names, data types, size etc.).
  5. Add a ***unique prefix*** as a naming convention in the **Field name** of any custom fields. The prefix prevents conflicts with fields that Kentico may add in the future.

     > When you save the custom class field, the system creates a custom column in the corresponding database table.
     >
     > During upgrades, the system merges the database definitions for customizable classes. The default fields may be updated and overwritten, but the custom fields are preserved.

- Customization of **database table indexes** is supported, but may require additional maintenance during upgrades:
  - You can drop the default indexes and create custom indexes for tables. You need to perform custom index operations manually in the database (via scripts or SQL Server Management Studio).
  - Use a unique naming convention for your custom indexes and maintain documentation to keep track of the changes.
  - The SQL script performed by upgrades may fail if the system attempts to create an index that is not compatible with your custom indexes. For example, if you drop a table's default clustered index and create your own, the SQL upgrade may fail if it attempts to update or recreate the default clustered index.
  - After applying an upgrade, review each table with custom indexes and make sure the indexes are still relevant and fulfill your requirements.

Adding custom database tables

If you need to store data in a custom table within the Kentico database, use the administration interface to create one of the following objects:

- Custom class (under a Custom module) – the best option for fully integrating the data into Kentico.
- Page type – intended for data that you wish to store as pages in the website's content tree.
- Form – suitable for data that you wish to collect from the site's visitors.
- Custom table – only use if you need to store and edit simple data structures without the need to create a custom module and define an editing interface. Otherwise, custom module classes are recommended instead.

See Defining website content structure if you need help deciding which approach is the best for you.

Note: If you wish to define custom fields that reference other Kentico objects (i.e. foreign keys), you need to use custom module classes. See Adding references between classes.

In all cases, the system automatically creates a database table based on the options that you configure. Any fields (columns) that you define for the table remain unchanged during upgrades. In rare cases, the upgrade may add new general fields required by the system (such additions should not break or affect the table's functionality).

Important

When creating objects representing custom database tables (such as custom classes, page types, forms, etc.), always use a ***unique prefix (namespace)*** in the code name of the object and the matching table name. This convention prevents conflicts with objects and tables that Kentico may add in the future.

Never use the **cms**, **com**, **om** or **content** prefixes for custom objects or database tables. The best option is to use your company name or an abbreviation as the prefix, for example: *ACME_MyTable*

Customizing Kentico database objects (stored procedures, views, functions, etc.)

- Do NOT directly modify the default Kentico database objects. Upgrades can overwrite any database object and the customizations will be lost.
- You can add custom database objects. Always use a ***unique prefix*** or other naming convention for custom database objects. The prefix prevents conflicts with objects that Kentico may add in the future.
- If you need to use a modified version of a Kentico database object, create your own custom object and copy the code from the original (use a unique naming convention). Then make any required modifications and ensure that your custom functionality uses the changed version of the database object.

If you need to create a custom **View** or **Stored procedure**, but do not have direct access to the database, you can use the **Database objects** application in the Kentico administration interface. This approach is equivalent to creating or scripting the object manually in the database.

Note: Do NOT use the *Database objects* application to edit the default views or procedures.

Default data

Each installation of Kentico automatically includes a set of default objects. These objects serve as examples of Kentico functionality and in some cases are important components of the system.

The following is a list of object types whose ***default*** objects are "owned by the system":

Default objects that are all overwritten during every upgrade:

- Web parts
- Web part layouts
- Widgets
- Reports (for example used by Online marketing, Web analytics and E-commerce)
- Time zones

Default objects that can be individually overwritten by any upgrade or hotfix:

- Form controls

- UI elements (managed when editing *Modules* on the *User interface* tab)

Object types with a specific subset of default objects that are "owned by the system":

- Page templates – all default templates in the **UI templates** category are critical system objects.
- Page types – the following page types are integral parts of the system:
  - **Root** (CMS.Root)
  - **File** (CMS.File)
  - **Page (menu item)** (CMS.MenuItem)
  - **Chat - Transformations** (Chat.Transformations)

> All default **transformations** and **queries** stored under the given page types are also considered system objects. You can however add new queries or transformations under the system page types without problems.

To avoid problems during upgrades, handle the default objects listed above according to the following rules:

- Do NOT edit or delete any of the listed default objects. Many of the objects are crucial components that are required for the system and administration interface to work correctly.
- If you need to modify one of the default objects, create your own copy of the original object instead (in many cases you can use the **clone** feature). Then make the changes to your custom object and use it for the required purpose.
- If you cannot avoid making changes directly to one of the default Kentico objects, you need to assume that any upgrade or hotfix will ***overwrite*** your customizations. Maintain detailed documentation of all customizations so you can manually replicate them after an upgrade.
- You can create your own objects of the listed types without any problems. Upgrades only overwrite the default objects. For example, upgrades overwrite changes that you make to the default web parts, but any custom web parts that you create will not be affected.
- If you need to modify the user interface of the default Kentico applications, do NOT use the *Customize* option for default UI elements. Such changes can be lost during upgrades. Instead, add your own custom UI elements and then use the resulting interface instead of the original:
  1. Create a custom module in the **Modules** application.
  2. Add UI elements on the module's **User interface** tab.
     - All custom UI elements must be assigned to a custom module.
     - You can copy the settings of the original UI elements that you want to customize and then make your changes.
     - Either add the custom UI elements under existing elements or create an entire custom application in the **CMS -> Administration -> Custom** section of the UI element tree.
     - See the Creating custom modules documentation for more information.

> **Other default objects**
>
> In rare cases, specific upgrades or hotfixes may also overwrite, delete or convert other types of objects. To minimize the risk of losing changes made to default objects, we recommend taking the following steps, even for object types that are not listed above (for example Email templates):
>
> - Maintain documentation of all changes that you make to default objects
> - Carefully check the instructions for each upgrade or hotfix version before applying

Physical files and code changes

Use the following general rules when working with files in the Kentico web project:

- Avoid modifications of the default Kentico files whenever possible. Adding of new custom files is preferable and easier to maintain through upgrades. Kentico provides a customization model that allows you to implement various scenarios by adding custom classes (for example Event handling or Provider customization).
- Do NOT use Kentico API that is marked as *obsolete*. Such API will most likely be removed in the next version of Kentico, and your code will stop working.
- If you need to execute SQL queries from code, avoid hardcoding of the query text. Hardcoded queries can cause errors if the underlying database structure changes. If possible, use the Kentico data engine API to generate your queries (ObjectQuery, DocumentQuery, DataQuery).
- Maintain documentation to keep track of your custom files and modifications made to the default files.

> **Tip**: A good way to keep track of custom files and file modifications is to create a parallel web project next to your Kentico project. Start with an empty web application project and add (as links) any new or customized files from the main project. Recreate the Kentico folder structure as necessary to organize the files.

Adding custom files

Adding of custom files into the Kentico project is a necessary part of many customization scenarios. For example, you need to add files when:

- Defining custom objects or functionality (such as web parts, scheduled tasks, smart search indexes, custom macro method containers, etc.)
- Performing initialization code (such as assigning handlers to events)

We recommend adding custom files into the project's directories based on the Export folder structure. These locations allow the system to automatically export files along with related database objects.

When adding code files (classes), you can choose between two different approaches:

- ***(Preferred)*** Create the files as part of a custom project (library). Then add the project to the Kentico solution and connect it through references.
  – OR –
- Add the files directly into the Kentico project. Use the **App_Code** folder for web site projects.

> **Advantages of custom code projects**
>
> Using separate projects (libraries) for custom code has the following advantages:
>
> - Cleaner separation of custom code from the default code of the Kentico web project.
> - Compilation performance – the code of a separate project is compiled into a DLL and does not require runtime compilation that slows down the web project.
> - Better accessibility of your custom classes from external applications or projects (for example projects running automated tests).
> - Easier re-usability across multiple projects.

> **Maintaining references between Kentico and custom projects**
>
> When performing upgrades, references from the Kentico projects (CMS, CMSApp) to your custom projects are preserved. Upgrades merge any new content in the corresponding *.csproj* files with your custom content (such as references).
>
> However, references from custom projects to Kentico libraries (DLLs) need to be added again after performing an upgrade. The change in the versions of the Kentico DLLs breaks the original references.

Custom files that you add are not directly affected by upgrades. However, you still need to check each custom file after performing an upgrade:

- Make sure the function performed by the file is still relevant for the new Kentico version (read the release notes and upgrade instructions).
- Update any Kentico API that you call in code files to be compatible with the latest version. We recommend using the code upgrade tool and/or the list of API changes.

Modifying default files in the Kentico project

If you cannot avoid customizing one or more of the default files in the **CMS** web site project or **CMSApp** web application project:

- Add comments and/or regions to clearly identify your custom code within the default Kentico code. This makes it easier to keep track of your changes and allows you to use the code upgrade tool more efficiently.
- During an upgrade, the system automatically detects modified files and creates new versions of them with the *new* extension. You need to manually merge your customizations into the new version of the file (use Visual Studio or another comparison and merging tool).
- You need to manually ensure that any Kentico API you call in your custom code sections is up-to-date. We recommend using the code upgrade tool and/or the list of API changes.

The following are guidelines for common customization scenarios related to the default files:

- **Web.config** -- you can directly edit your project's *web.config* file and add any required configuration options. Upgrades attempt to automatically merge your web.config customizations with any changes introduced by the new version. If the automatic merge fails due to incompatible configuration, a separate web.config file with the *.new* extension is created and you need to combine the file content manually.

- **Global.asax** – do NOT edit or add content to the *Global.asax* file in the Kentico project. If you wish run code at specific points within the application life cycle, use the customization model to assign handlers of the appropriate Kentico events *(ApplicationEvents, RequestEvents).* See Reference - Global system events.

- **Administration interface customizations** – if you need to customize files that affect the default Kentico administration interface, you can edit them directly within the project:
    - UniGrid XML definitions used for listing pages - typically stored as *~/App_Data/CMSModules/<module name>/UI/Grids/<object type>/default.xml*
    - Web form pages used for UI elements with manually defined content - typically stored in *~/CMSModules/...*

    > **Note**: Avoid customizations of the default administration interface pages when possible. The recommended approach is to create your own UI elements and then use the resulting interface instead of the original (see the Default data section).

Changing the project structure

Upgrades must be applied to complete Kentico projects that use the standard folder structure (including the solution file, *GlobalAssemblyInfo.cs*, the *CMS* and *Lib* sub-folders). For example, you cannot directly upgrade web site deployments that only consist of the CMS folder.

If you wish to run a Kentico project with an incomplete or modified folder structure, you need to use the following development and upgrade process:

1. Maintain a development version of the project with the original complete folder structure.
2. Prepare your modified version of the project and use it to deploy your production website.
3. Apply upgrades to the development version of the project. After an upgrade is successfully completed, prepare a new version of the modified project and redeploy your website.

Using jQuery

Do NOT use the default jQuery provided in Kentico projects for your own client scripts. Newer versions of Kentico may contain a different version of jQuery, so your scripts may not work correctly after upgrading.

Best practice: Add, link and maintain your own jQuery library within the Kentico project. This gives you full control over the jQuery version and helps protect your custom scripts from breaking when upgrading Kentico.

Note: The Kentico jQuery is registered in No-Conflict mode under the **$cmsj** alias.