

Kentico CMS 5.5 API Examples

Data layer

SettingsProvider

- namespace CMS.SettingsProvider
- DataClassInfoProvider (DataClassInfo)
- QueryProvider (Query)
- SettingsHelper

DataEngine

- namespace CMS.DataEngine
- CMSTransactionScope
- ConnectionHelper
- GeneralConnection
- DataCacheHelper

Data layer
You may want to use it to explicitly use a connection or transaction within a given block of code. If you don't use it, every call will use it's own.

Connections & Transactions

Using connection

```
using (CMSTransactionScope cs = new CMSTransactionScope())
{
    ... do your actions
}
```

Using transaction

```
using (CMSTransactionScope tr = new CMSTransactionScope())
{
    ... do your actions in transaction
    // Commit the changes, you can also use the Complete() method
    tr.Commit();
}
```

Managing objects

Create new object

```
// Create UserInfo
UserInfo user = new UserInfo();

user.UserName = "alice";
user.FirstName = "Alice";
user.LastName = "Cooper";
user.Email = "alice.cooper@domain.com";
user.IsEditor = true;

// Save the user
UserInfoProvider.SetUserInfo(user);

// Set user password
UserInfoProvider.SetPassword(user, "pass");
```

Edit existing object

```
// Get UserInfo object by user name
UserInfo user = UserInfoProvider.GetUserInfo("alice");

// Edit some properties
user.UserNickName = "Ali";

// Save the changes
UserInfoProvider.SetUserInfo(user);
```

Delete object

```
// Delete user
UserInfoProvider.DeleteUser(user);
```

Get objects by specific parameters

```
// Get all users without e-mail address
DataSet data = UserInfoProvider.GetUsers("Email IS NULL", "LastName");
```

Managing M:N relationships

Add object to site

```
// Add user to the current site
UserInfoProvider.AddUserToSite(user, CMSContext.CurrentSite);
```

Remove object from site

```
// Remove user from the current site
UserInfoProvider.RemoveUserFromSite(user.UserID, CMSContext.CurrentSiteID);
```

Managing object attachments

Create object attachment

```
// Create metafile from uploaded file
MetaFileInfo file = new MetaFileInfo(Uploader.PostedFile, userID, "cms.user", "mycategory");
file.MetaFileSiteID = CMSContext.CurrentSiteID;

// Save the metafile
MetaFileInfoProvider.SetMetaFileInfo(file);
```

Get object attachments

```
// Get metafiles specified by objectID, objectType, category and eventually by where condition
DataSet files = MetaFileInfoProvider.GetMetaFiles(userID, "cms.user", "mycategory", null, "MetaFileName");
```

Edit existing object attachment

```
// Get metafile by its ID
MetaFileInfo file = MetaFileInfoProvider.GetMetaFileInfo(metaFileID);

file.MetaFileName = "test_attachment.jpg";

// Set the changes
MetaFileInfoProvider.SetMetaFileInfo(file);
```

Delete object attachment

```
// Delete metafile
MetaFileInfoProvider.DeleteMetaFileInfo(file);
```

Managing objects
You can apply the examples to **any object type**. Just use the right **info** and **provider** classes.

Managing documents
The approach to documents depends on whether you just want to **display** them on the live site or **edit** them. **Versioning** and **workflow** have their own manager classes with methods corresponding to user actions in the admin UI. The **TreeNode** class always represents the actual document data.

Managing documents

Get dataset of documents for live site

```
DataSet data = TreeHelper.GetDocuments("CorporateSite", "/Products/Notebooks/FS-V2030", "en-us", "..."); // Automatically handles preview vs. published version
```

Get single document for live site

```
TreeNode node = TreeHelper.GetDocument("CorporateSite", "/Products/Notebooks/FS-V2030", "en-us", "..."); // Automatically handles preview vs. published version
```

Get dataset of documents

```
DataSet data = DocumentHelper.GetDocuments("CorporateSite", "/Products/Notebooks/FS-V2030", "en-us", "..."); // Always gets the latest (edited) version
```

Get single document

```
TreeNode node = DocumentHelper.GetDocument("CorporateSite", "/Products/Notebooks/FS-V2030", "en-us", "..."); // Always gets the latest (edited) version
```

Create new document

```
// Prepare the TreeProvider (must be initialized with user)
UserInfo user = UserInfoProvider.GetUserInfo("administrator");
TreeProvider tree = new TreeProvider(user);

// Create new document (TreeNode = document)
TreeNode node = new TreeNode("CMS.News", tree);
node.DocumentName = "TestingNews";
node.DocumentCulture = CMSContext.PreferredCultureCode;
// Set custom fields
node.SetValue("NewsTitle", "Testing news");
node.SetValue("NewsText", "Testing site testing news text");

// Insert it under parent document specified by parentNodeID
DocumentHelper.InsertDocument(node, parentNodeID, tree);
```

Managing document attachments

Get s list of all attachments for the given document without binary data

```
DataSet attachments = DocumentHelper.GetAttachments(node, null, null, false, tree);
```

Get an attachment with binary data

```
AttachmentInfo attachInfo = DocumentHelper.GetAttachment(node, attachmentGUID, tree);
```

Add or update a field attachment of the document

```
AttachmentInfo attInfo = DocumentHelper.AddAttachment(node, "FileAttachment", file, tree);
```

Add or update grouped attachment of the document

```
AttachmentInfo attInfo = DocumentHelper.AddGroupedAttachment(node, attachmentGUID, groupGUID, file, tree, ...); // This applies to document attachments field type
```

Add or update unsorted attachment of the document

```
AttachmentInfo attachInfo = DocumentHelper.AddUnsortedAttachment(node, attachmentGUID, file, tree, ...); // These are the attachments on Attachments tab
```

Delete an attachment

```
DocumentHelper.DeleteAttachment(node, attachmentGUID, tree); // Any of the types above
```

Workflow

Approve the document

```
// Initialize workflow manager and approve the document
WorkflowManager wm = new WorkflowManager(tree);
WorkflowStepInfo newStep = wm.MoveToNextStep(node, comment); // = Approve
```

Reject the document

```
WorkflowStepInfo newStep = wm.MoveToPreviousStep(node, comment); // = Reject
```

Archive the document

```
WorkflowStepInfo newStep = wm.ArchiveDocument(node, comment); // Must be in Published
```

Approve the document to specific step (publish)

```
// Get current step information
WorkflowStepInfo currentStep = wm.GetStepInfo(node);

// Use the workflow step information to approve the document until it is published
while (currentStep.StepName.ToLower() != "published")
{
    currentStep = wm.MoveToNextStep(node, "");
    if (currentStep == null) break; // Stop in the case the workflow was removed
}
}
```

Update the document

```
node.SetValue("MyColumn", newValue);
// Updates the document data to the database
DocumentHelper.UpdateDocument(node, tree);
```

Create new culture version

```
node.DocumentCulture = "cs-cz";
// Creates new culture version of a document
DocumentHelper.InsertNewCultureVersion(node, tree);
```

Create linked document

```
// Insert the document as a link under the parent specified by node ID
DocumentHelper.InsertDocumentAsLink(node, parentID, tree);
```

Delete the document

```
// Delete document with all cultures, document history and products
DocumentHelper.DeleteDocument(node, tree, true, true, true);
```

Versioning

Check-out the document

```
// Initialize version manager and check-out the document
VersionManager vm = new VersionManager(tree);
vm.CheckOut(node); // Lock, also creates new minor version
```

Check-in the document

```
vm.CheckIn(node, versionNumber, versionComment); // Unlock
```

Undo the document check-out

```
vm.UndoCheckOut(node); // Undo check-out also drops latest version
```

Application layer

SiteProvider

- namespace CMS.SiteProvider
- BadWordInfoProvider (BadWordInfo)
- BadWordsHelper
- CustomTableItemProvider (CustomTableItem)
- CountryInfoProvider (CountryInfo)
- CultureInfoProvider (CultureInfo)
- StateInfoProvider (StateInfo)
- TimeZoneHelper
- TimeZoneInfoProvider (TimeZoneInfo)
- UICultureInfoProvider (UICultureInfo)
- CssStyleSheetInfoProvider (CssStyleSheetInfo)
- FormUserControlInfoProvider (FormUserControlInfo)
- InlineControlInfoProvider (InlineControlInfo)
- LayoutInfoProvider (LayoutInfo)
- AvatarInfoProvider (AvatarInfo)
- BadgeInfoProvider (BadgeInfo)
- OpenIDUserInfoProvider (OpenIDUserInfo)
- PermissionNameInfoProvider (PermissionNameInfo)
- RoleInfoProvider (RoleInfo)
- RolePermissionInfoProvider (RolePermissionInfo)
- RoleUIElementInfoProvider (RoleUIElementInfo)
- UIElementInfoProvider (UIElementInfo)
- UserInfoProvider (UserInfo)
- UserRoleInfoProvider (UserRoleInfo)
- UserSettingsInfoProvider (UserSettingsInfo)
- UserSiteInfoProvider (UserSiteInfo)
- AllowedChildClassInfoProvider (AllowedChildClassInfo)
- CategoryInfoProvider (CategoryInfo)
- ClassSiteInfoProvider (ClassSiteInfo)
- MetaFileInfoProvider (MetaFileInfo)
- MetaFileURLProvider
- ObjectRelationshipInfoProvider (ObjectRelationshipInfo)
- RelationshipNameInfoProvider (RelationshipNameInfo)
- ResourceInfoProvider (ResourceInfo)
- SettingsCategoryInfoProvider (SettingsCategoryInfo)
- AbuseReportInfoProvider (AbuseReportInfo)
- BannedIPInfoProvider (BannedIPInfo)
- FloodProtectionHelper
- SiteDomainAliasInfoProvider (SiteDomainAliasInfo)
- SiteInfoProvider (SiteInfo)
- WebTemplateInfoProvider (WebTemplateInfo)
- SiteContext
- SearchHelper
- SearchIndexCultureInfoProvider (SearchIndexCultureInfo)
- SearchIndexInfoProvider (SearchIndexInfo)
- SearchIndexSiteInfoProvider (SearchIndexSiteInfo)
- SearchTaskInfoProvider (SearchTaskInfo)
- TagGroupInfoProvider (TagGroupInfo)
- TagHelper
- TagInfoProvider (TagInfo)

FileManager

- namespace CMS.FileManager
- AttachmentInfo
- AttachmentManager
- AttachmentURLProvider

TreeEngine

- namespace CMS.TreeEngine
- DocumentCategoryInfoProvider (DocumentCategoryInfo)
- DocumentTagInfoProvider (DocumentTagInfo)
- DocumentAliasInfoProvider (DocumentAliasInfo)
- DocumentURLProvider
- TreeNode
- TreePathUtils
- TreeProvider
- AcInfo
- AcItemInfo
- AcProvider
- TreeSecurityProvider
- RelationshipProvider (RelationshipInfo)

WorkflowEngine

- namespace CMS.WorkflowEngine
- AttachmentHistoryInfoProvider (AttachmentHistoryInfo)
- VersionHistoryInfoProvider (VersionHistoryInfo)
- WorkflowHistoryInfoProvider (WorkflowHistoryInfo)
- WorkflowInfoProvider (WorkflowInfo)
- WorkflowScopeInfoProvider (WorkflowScopeInfo)
- WorkflowStepInfoProvider (WorkflowStepInfo)
- WorkflowStepRoleInfoProvider (WorkflowStepRoleInfo)
- DocumentHelper
- FileImport
- VersionAttachmentInfo
- VersionManager
- WorkflowManager

CMSStaging

- namespace CMS.Staging
- ServerInfoProvider (ServerInfo)
- TaskInfoProvider (TaskInfo)

FormEngine

- namespace CMS.FormEngine
- AlternativeFormInfoProvider (AlternativeFormInfo)
- BizFormInfoProvider (BizFormInfo)

LicenseProvider

- namespace CMS.LicenseProvider
- LicenseHelper
- LicenseKeyInfoProvider (LicenseKeyInfo)

WebFarmSync

- namespace CMS.WebFarmSync
- WebFarmServerInfoProvider (WebFarmServerInfo)
- WebFarmTaskInfoProvider (WebFarmTaskInfo)

GlobalHelper

- namespace CMS.GlobalHelper
- BrowserHelper
- CacheHelper
- CMSThread
- CookieHelper
- HTTPHelper
- LogContext
- PersistentStorageHelper
- QueryHelper
- RegistryHelper
- RequestStockHelper
- SessionHelper
- UIHelper
- URLHelper
- WindowHelper
- CustomData
- DataHelper
- MimeTypeHelper
- RegexHelper
- ValidationHelper
- Validator
- CultureHelper
- DateTimeHelper
- ResHelper
- CSSHelper
- DiscussionMacroHelper
- HTMLHelper
- ScriptHelper
- TextHelper
- ImageHelper
- MediaHelper

EventLog

- namespace CMS.EventLog
- EventLogProvider

Scheduler

- namespace CMS.Scheduler
- TaskInfoProvider (TaskInfo)

EmailEngine

- namespace CMS.EmailEngine
- EmailAttachment
- EmailMessage
- EmailSender
- EmailTemplateInfoProvider (EmailTemplateInfo)

Caching

```
DataSet data = null;
// Cache the data for 10 minutes with key "mykey" <somevalue>
using (CachedSection<DataSet> cs = new CachedSection<DataSet>(ref data, 10, true, null, "mykey", someValue))
{
    if (cs.LoadData)
    {
        data = ...; // Get data from database
    }

    cs.CacheDependency = ...; // Optional cache dependencies
    cs.Data = data; // Save data to cache
}
```

Caching of data
You may want to use the caching in your code to get **better performance**, here is a code snippet that can be used with any of your code.

Managing module objects
You can work with module objects the same way as with **general objects**, using their **info** and **provider** classes.

Modules

Community

- namespace CMS.Community
- CommunityContext
- FriendInfoProvider (FriendInfo)
- GroupInfoProvider (GroupInfo)
- GroupMemberInfoProvider (GroupMemberInfo)
- InvitationInfoProvider (InvitationInfo)

MediaLibrary

- namespace CMS.MediaLibrary
- MediaFileInfoProvider (MediaFileInfo)
- MediaFileURLProvider
- MediaLibraryContext
- MediaLibraryHelper
- MediaLibraryInfoProvider (MediaLibraryInfo)

Blogs

- namespace CMS.Blogs
- BlogCommentInfoProvider (BlogCommentInfo)
- BlogPostSubscriptionInfoProvider (BlogPostSubscriptionInfo)
- BlogHelper
- BlogCommentDetail

MessageBoard

- namespace CMS.MessageBoard
- BoardInfoProvider (BoardInfo)
- BoardMessageInfoProvider (BoardMessageInfo)
- BoardModeratorInfoProvider (BoardModeratorInfo)
- BoardRoleInfoProvider (BoardRoleInfo)
- BoardSubscriptionInfoProvider (BoardSubscriptionInfo)

Ecommerce

- namespace CMS.Ecommerce
- AddressInfoProvider (AddressInfo)
- CurrencyInfoProvider (CurrencyInfo)
- CustomerInfoProvider (CustomerInfo)
- DepartmentInfoProvider (DepartmentInfo)
- DepartmentTaxClassInfoProvider (DepartmentTaxClassInfo)
- DiscountCouponInfoProvider (DiscountCouponInfo)
- DiscountLevelDepartmentInfoProvider (DiscountLevelDepartmentInfo)
- DiscountLevelInfoProvider (DiscountLevelInfo)
- ExchangeRateInfoProvider (ExchangeRateInfo)
- ExchangeTableInfoProvider (ExchangeTableInfo)
- InternalStatusInfoProvider (InternalStatusInfo)
- ManufacturerInfoProvider (ManufacturerInfo)
- OptionCategoryInfoProvider (OptionCategoryInfo)
- OrderInfoProvider (OrderInfo)
- OrderItemInfoProvider (OrderItemInfo)
- OrderStatusInfoProvider (OrderStatusInfo)
- PaymentOptionInfoProvider (PaymentOptionInfo)
- PaymentShippingInfoProvider (PaymentShippingInfo)
- PublicStatusInfoProvider (PublicStatusInfo)
- ShippingOptionInfoProvider (ShippingOptionInfo)
- ShoppingCartInfoProvider (ShoppingCartInfo)
- ShoppingCartItemInfoProvider (ShoppingCartItem)
- SKUDiscountCouponInfoProvider (SKUDiscountCouponInfo)
- SKUInfoProvider (SKUInfo)
- SKUOptionCategoryInfoProvider (SKUOptionCategoryInfo)
- SKUClassInfoProvider (SKUClassInfo)
- SupplierInfoProvider (SupplierInfo)
- TaxClassInfoProvider (TaxClassInfo)
- UserDepartmentInfoProvider (UserDepartmentInfo)
- VolumeDiscountInfoProvider (VolumeDiscountInfo)
- ECommerceContext

Messaging

- namespace CMS.Messaging
- ContactListInfoProvider (ContactListInfo)
- IgnoreListInfoProvider (IgnoreListInfo)
- MessageInfoProvider (MessageInfo)
- MessagingHelper

Newsletter

- namespace CMS.Newsletter
- EmailQueueItem
- EmailQueueManager
- EmailTemplateProvider (EmailTemplate)
- IssueProvider (Issue)
- IssueHelper
- NewsletterProvider (Newsletter)
- SubscriberProvider (Subscriber)

EventManager

- namespace CMS.EventManager
- EventAttendeeInfoProvider (EventAttendeeInfo)
- EventProvider

Polls

- namespace CMS.Polls
- PollAnswerInfoProvider (PollAnswerInfo)
- PollInfoProvider (PollInfo)

Notifications

- namespace CMS.Notifications
- NotificationGatewayInfoProvider (NotificationGatewayInfo)
- NotificationSubscriptionInfoProvider (NotificationSubscriptionInfo)
- NotificationTemplateInfoProvider (NotificationTemplateInfo)
- NotificationTemplateTextInfoProvider (NotificationTemplateTextInfo)

Forums

- namespace CMS.Forums
- ForumContext
- ForumAttachmentInfoProvider (ForumAttachmentInfo)
- ForumGroupInfoProvider (ForumGroupInfo)
- ForumInfoProvider (ForumInfo)
- ForumModeratorInfoProvider (ForumModeratorInfo)
- ForumPostInfoProvider (ForumPostInfo)
- ForumRoleInfoProvider (ForumRoleInfo)
- ForumSubscriptionInfoProvider (ForumSubscriptionInfo)
- ForumUserFavoritesInfoProvider (ForumUserFavoritesInfo)

WebAnalytics

- namespace CMS.WebAnalytics
- AnalyticsHelper
- HitLogProvider
- HitsInfoProvider (HitsInfo)
- IP2CountryHelper
- StatisticsInfoProvider (StatisticsInfo)