You can extend system objects in Kentico by adding your own fields (properties).

1. Create a custom data class containing the required fields.
2. Connect the class to the appropriate system object. All standard system objects implement the **IRelatedData** interface, which allows you to connect any type of class:
    - Through the **RelatedData** property
    - Dynamically by handling the **OnLoadRelatedData** event

If the connected class implements the **IDataContainer** interface, you can then access the data stored in the custom fields as part of the system object via the API (**GetValue** and **SetValue** methods) or macro expressions.

Example

The following sections demonstrate how to extend the **SiteInfo** object, which represents sites in Kentico. The example adds two custom properties for sites:

- **SiteOwner** (string)
- **SiteValidUntil** (DateTime)

Defining the custom data class

1. Open your Kentico web project in Visual Studio.
2. Create a new class named **SiteRegistrationData.cs**.
    - 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).
3. Write the code of the class.
    - The class must implement the **IDataContainer** interface.
    - Define your custom properties and all required *IDataContainer* members (as shown in the code below).

```
using System;
using System.Collections.Generic;

using CMS.Base;

public class SiteRegistrationData : IDataContainer
{
    #region "Variables"

    private string mSiteOwner = null;
    private DateTime mSiteValidUntil = DateTime.MinValue;

    #endregion

    #region "Properties"

    /// <summary>
    /// Gets or sets the name of the site owner.
    /// </summary>
    public string SiteOwner
    {
        get
        {
            return mSiteOwner;
        }
        set
        {
            mSiteOwner = value;
```

```
        }
    }

    /// <summary>
    /// Gets or sets the date until which the site is valid.
    /// </summary>
    public DateTime SiteValidUntil
    {
        get
        {
            return mSiteValidUntil;
        }
        set
        {
            mSiteValidUntil = value;
        }
    }

    #endregion

    #region "IDataContainer members"

    /// <summary>
    /// Gets a list of column names.
    /// </summary>
    public List<string> ColumnNames
    {
        get
        {
            return new List<string>() { "SiteOwner", "SiteValidUntil" };
        }
    }

    /// <summary>
    /// Returns true if the class contains the specified column.
    /// </summary>
    /// <param name="columnName"></param>
    public bool ContainsColumn(string columnName)
    {
        switch (columnName.ToLower())
        {
            case "siteowner":
            case "sitevalidduntil":
                return true;
            default:
                return false;
        }
    }

    /// <summary>
    /// Gets the value of the specified column.
    /// </summary>
    /// <param name="columnName">Column name</param>
    public object GetValue(string columnName)
    {
        switch (columnName.ToLower())
        {
            case "siteowner":
                return mSiteOwner;
```

```
            case "sitevaliduntil":
                return mSiteValidUntil;

            default:
                return null;
        }
    }

    /// <summary>
    /// Sets the value of the specified column.
    /// </summary>
    /// <param name="columnName">Column name</param>
    /// <param name="value">New value</param>
    public bool SetValue(string columnName, object value)
    {
        switch (columnName.ToLower())
        {
            case "siteowner":
                mSiteOwner = (string)value;
                return true;

            case "sitevaliduntil":
                mSiteValidUntil = (DateTime)value;
                return true;

            default:
                return false;
        }
    }

    /// <summary>
    /// Returns a boolean value indicating whether the class contains the specified
column.
    /// Passes on the specified column's value through the second parameter.
    /// </summary>
    /// <param name="columnName">Column name</param>
    /// <param name="value">Return value</param>
    public bool TryGetValue(string columnName, out object value)
    {
        switch (columnName.ToLower())
        {
            case "siteowner":
                value = mSiteOwner;
                return true;

            case "sitevaliduntil":
                value = mSiteValidUntil;
                return true;

            default:
                value = null;
                return false;
        }
    }

    /// <summary>
    /// Gets or sets the value of the column.
    /// </summary>
    /// <param name="columnName">Column name</param>
    public object this[string columnName]
```

```
    {
        get
        {
            return GetValue(columnName);
        }
        set
        {
            SetValue(columnName, value);
        }
    }

    #endregion

}
```

Connecting the custom data class to the system object

You need to bind the custom *SiteRegistrationData* class to the *SiteInfo* object. The example uses the **OnLoadRelatedData** event, which you can handle for specific object types, including *SiteInfo* objects. This type of binding is dynamic, which means that the system loads the data only when it is requested.

1. Create a custom module class in the same location as the *SiteRegistrationData* class.
2. Override the module's **OnInit** method and assign a handler to the **SiteInfo.TYPEINFO.OnLoadRelatedData** event.
3. Define the handler method for the **OnLoadRelatedData** event:
    - The handler must return an instance of your custom data class (with appropriate values assigned to the class's fields).

```
using System;

using CMS;
using CMS.DataEngine;
using CMS.SiteProvider;

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

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

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

                // Assigns a handler to the OnLoadRelatedData event of the SiteInfo
object type
                SiteInfo.TYPEINFO.OnLoadRelatedData += SiteInfo_OnLoadRelatedData;
        }

        // Handler method for the OnLoadRelatedData event of the SiteInfo class
        // Gets the related data from the custom storage class (must implement the
IDataContainer interface)
        static object SiteInfo_OnLoadRelatedData(BaseInfo infoObj)
        {
                SiteRegistrationData siteData = new SiteRegistrationData();

                siteData.SiteOwner = "John Smith";
                siteData.SiteValidUntil = DateTime.Now.AddDays(1);

                return siteData;
        }
}
```

Result

You can now work with the custom fields of site objects using the API.

For example, log in to the Kentico administration interface and open your website in the **Pages** application. Add the following code into the ASCX layout of one of your website's pages:

```
<asp:Label runat="server" id="lblSiteInfo" />
<script runat="server">
        protected void Page_PreRender(object sender, EventArgs e)
        {
                CMS.SiteProvider.SiteInfo currentSite = CMS.SiteProvider.SiteContext.
CurrentSite;

                this.lblSiteInfo.Text = String.Format("Site '{0}' is valid until {1}
and owned by {2}.", currentSite.DisplayName, currentSite.GetValue("SiteValidUntil"),
currentSite.GetValue("SiteOwner"));
        }
</script>
```

The **GetValue** method allows you to retrieve the data stored in the custom properties, just like with native fields of the *SiteInfo* object.

The label control displays information on the page in the following format:

Site 'Corporate Site' is valid until 1/15/2014 1:00:00 PM and owned by John Smith.

Accessing properties through macros

You can also load the values of custom properties using macro expressions. For example:

1. In the **Pages** application, edit a page on the **Design** tab.
2. Add the **Static text** web part to the page.
3. Copy the following text into the web part's **Text** property.

```
<ul>
        <li>Site: {% CurrentSite.SiteDisplayName %}</li>
        <li>Valid until: {% CurrentSite.SiteValidUntil %}</li>
        <li>Owned by: {% CurrentSite.SiteOwner %}</li>
</ul>
```

The web part displays a list of information about the current website, including the values of the custom fields.