Forms allow you to gather structured data from your site visitors. A typical example can be a feedback form or a form in which visitors provide additional information about themselves. The data a visitor submits via forms then updates the contact that represents the visitor in the system. This way, visitors can provide email addresses that you can then use to send marketing emails.

To use the Kentico forms functionality on an MVC site, you need to:

- Create a form in the **Forms** application and generate its object class
- Handle submission of the form data in the MVC application

Data protection regulation compliance

To comply with GDPR and other data protection regulations, you can add consents to forms on MVC sites.

Compatibility with the Forms application's functionality

Kentico features related to forms are NOT supported by default for forms and form submissions that occur on MVC sites. For example:

- The Form builder and Layout functionality – you need to define the look of your form in the corresponding view.
- When creating form fields, only the general settings such as the **Field name**, **Data type**, and **Size** apply. Field appearance settings and features, such as form controls, are not supported.
- The default Kentico functionality for email notifications about form submissions.
- The Autoresponder functionality.

Adding a basic form to an MVC site

The following example demonstrates how to create a basic feedback form and integrate it into an MVC website.

Tip: To view the full code of a functional example directly in Visual Studio, download the Kentico MVC solution from GitHub and inspect the **LearningKit** project. You can also run the Learning Kit website after connecting the project to a Kentico database.

Creating a form in Kentico

Before you add the necessary functionality to your MVC project, you first need to create the form in your Kentico application and generate its object class:

1. Open the **Forms** application and click **New form**.
2. Fill in the following properties:
    - **Form display name**: Feedback form
    - **Form code name**: FeedbackForm
    - **Table name**: FeedbackForm

3. Click **Save**.
4. Switch to the **Fields** tab.
5. Create the following fields:
    - **Field name**: UserName
    - **Data type**: Text
    - **Size**: 200

    - **Field name**: UserLastName
    - **Data type**: Text
    - **Size**: 200

    - **Field name**: UserEmail
    - **Data type**: Text

- **Size**: 254

- **Field name**: UserFeedback
- **Data type**: Text
- **Size**: 2000

6. Switch to the **Code** tab.
7. Add the generated **FeedbackFormItem** class to your MVC project.

The *Feedback form* is now ready to collect submissions from your MVC application.

Connecting the form to an MVC application

To connect the form with your MVC application, you need to:

1. Create a new controller class in your MVC project or edit an existing one.
2. Implement two actions – one basic GET action to display the feedback form and a second POST action to handle the feedback form submission.
3. Perform the following steps within the POST action:
    a. Prepare an instance of the generated **FeedbackFormItem** class based on the submitted data.
    b. Call the **Insert** method of the *FeedbackFormItem* to submit the form data to the Kentico database.

Feedback form controller example

```
 »using System.Web.Mvc;

using CMS.OnlineForms.Types;

using LearningKit.Models.Form;

namespace LearningKit.Controllers
{
    public class FeedbackFormController : Controller
    {
        /// <summary>
        /// Basic action that displays the feedback form.
        /// </summary>
        public ActionResult Fill()
        {
            return View("FormFill");
        }

        /// <summary>
        /// Inserts the form data to the connected database when the feedback
form is submitted.
        /// Accepts parameters posted from the feedback form via the
FeedbackFormMessageModel.
        /// </summary>
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult SendFeedback(FeedbackFormMessageModel model)
        {
            // Validates the received form data based on the view model
            if (!ModelState.IsValid)
            {
                return View("FormFill", model);
            }

            // Inserts the collected form data to the connected database
            InsertFeedbackFormItem(model);

            return View("FormSendSuccess");
        }

        // Inserts the collected data to the connected database (helper method)
        private void InsertFeedbackFormItem(FeedbackFormMessageModel feedback)
        {
            var item = new FeedbackFormItem
            {
                UserName = feedback.FirstName,
                UserLastName = feedback.LastName,
                UserEmail = feedback.Email,
                UserFeedback = feedback.MessageText,
            };

            item.Insert();
        }
    }
}
```

4. We recommend creating a view model for your form submission action *(FeedbackFormMessageModel* in the example above). The view model allows you to:

   - Pass parameters from the feedback form (name, last name, email address, message text, etc.).
   - Use data annotations to define validation and formatting rules for the registration data. See System. ComponentModel.DataAnnotations on MSDN for more information about the available data annotation attributes.

Feedback form model example

```
 »using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace LearningKit.Models.Form
{
    public class FeedbackFormMessageModel
    {
        [DisplayName("Name")]
        [DataType(DataType.Text)]
        [MaxLength(200, ErrorMessage = "The name cannot be longer than 200 
characters.")]
        public string FirstName
        {
            get;
            set;
        }

        [DisplayName("Last name")]
        [DataType(DataType.Text)]
        [MaxLength(200, ErrorMessage = "The last name cannot be longer than 200 
characters.")]
        public string LastName
        {
            get;
            set;
        }

        [DisplayName("Email address")]
        [DataType(DataType.EmailAddress)]
        [Required(ErrorMessage = "The email address field is required.")]
        [EmailAddress(ErrorMessage = "Invalid email address format.")]
        [MaxLength(254, ErrorMessage = "The email address cannot be longer than 
254 characters.")]
        public string Email
        {
            get;
            set;
        }

        [DisplayName("Message")]
        [DataType(DataType.MultilineText)]
        [Required(ErrorMessage = "The message field is required.")]
        [MaxLength(2000, ErrorMessage = "The message cannot be longer than 2000 
characters.")]
        public string MessageText
        {
            get;
            set;
        }
    }
}
```

5. Design the user interface required for the feedback form on your website:

    - Create a view for the *Fill* action and display an appropriate form. We recommend using a strongly typed view based on your feedback form view model.

- Post the form data to the form submission action (*SendFeedback* in the example).

Visitors on your MVC site can now submit feedback through this form. Upon successful submission, the system creates a new item in the connected Kentico database. You can manage collected submissions on the corresponding form's **Recorded data** tab.