Using Forms in Razor Pages

Forms are used for transferring data from the browser to the web server for further processing, such as saving it to a database, constructing an email, or simply subjecting the data to some kind of algorithm and then displaying the result.

The HTML form element

The HTML `<form>` element is used to create a form on a web page. The `form` element has a number of attributes, the most commonly used of which are `method` and `action`. The `method` attribute determines the HTTP verb to use when the form is submitted. By default, the GET verb is used and the form values are appended to the receiving page's URL as query string values. If the action attribute is omitted, the form will be submitted to the current URL i.e. the page that the form is in.

Usually, you submit forms using the POST verb which removes the form values from the URL and allows more data to be sent in the request as query strings are limited by most browsers. Therefore you should provide a `method` attribute with the value `post`:

```
<form method="post">
...
</form>
```

Capturing user input

The primary role of the form is to capture input provided by the user for transfer to the web server. A collection of form controls, represented by the `input`, `select` and `textarea` elements are designed to accept user input for submission.

The `input` element's display and behaviour is controlled by its `type` parameter. If omitted, the `type` defaults to `text` and the control renders as a single line textbox:

There are a range of other `input` types whose behaviour and appearance differs based on the `type` value, and the browser:

Type	Example	Description
`checkbox`	☐	Renders as a check box
`color`	■	Renders a color picker
`date`	mm/dd/yyyy	Renders a date control
`dateTime`		Obsolete, replaced by `datetime-local`. Was never implemented by browser vendors.
`datetime-local` [1]	mm/dd/yyyy --:-- --	Creates a control that accepts the date and time and displays it in the browser's local format

Type	Example	Description
`email`		A text box that accepts valid email addresses only. Validation is performed by the browser
`file`	Choose File No file chosen	Renders a file selector
`hidden`		Nothing is rendered. Used to pass form values that do not need to be displayed
`image`		Renders a submit button using the specified image
`month` [1]	--------- ----	Renders a control designed to accept a month and year
`number`		Some browsers render a spinner control and refuse to accept non-numeric values
`password`		Values entered by the user are obscured for security purposes
`radio`	○	Renders as a radio button
`range`		Browsers render a slider control
`search`		A text box designed to accept search terms. Some browsers may provide additional features such as a content reset icon
`submit`	Submit	Renders a standard submit button with the text "Submit"
`tel`		A textbox designed to accept telephone numbers. Browsers do not validate for any specific format
`time` [1]	--:-- --	A control that accepts a time value in HH:mm format
`url`		A text input that validates for a URL
`week` [1]	Week --, ----	An input that accepts a week number and a year

1. These input types only enjoy **partial support across the latest browsers (https://caniuse.com/?search=date%20and%20time)**. None of them are supported by IE 11.

The two other most commonly used elements for capturing user input are the `textarea`, rendering a multi-line textbox, and the `select` element, which is used to encapsulate multiple `option` elements, providing the user with a mechanism for choosing one or more of a fixed list of options.

Accessing User Input

User input is only available to server-side code if the form control has a value applied to the `name` attribute. There are several ways to reference posted form values:

- Accessing the `Request.Form` collection via a string-based index, using the `name` attribute of the form control as the index value.
- Leveraging **Model Binding (/razor-pages/model-binding)** to map form fields to **handler method (/razor-pages/handler-methods)** parameters.
- Leveraging Model Binding to map form fields to public properties on a **PageModel (/razor-pages/pagemodel)** class.

Request.Form

⚠ This approach is not recommended, although it offers a level of familiarity to developers who are migrating from other frameworks (such as PHP, classic ASP or ASP.NET Web Pages) where `Request.Form` is the only native way to access posted form values.

Items in the `Request.Form` collection are accessible via their string-based index. The value of the string maps to the `name` attribute given to the relevant form field. The form below has one input that accepts values named `emailaddress`:

```
<form method="post">
    <input type="email" name="emailaddress">
    <input type="submit">
</form>
```

You can access the value in the `OnPost` handler method as follows:

```
public void OnPost()
{
    var emailAddress = Request.Form["emailaddress"];
    // do something with emailAddress
}
```

The string index is case-insensitive, but it must match the name of the input. The value returned from the `Request.Form` collection is always a string.

Further Information

- PageModel (/razor-pages/pagemodel)
- Handler Methods (/razor-pages/handler-methods)

Leveraging Model Binding

The recommended method for working with form values is to use **model binding (/razor-pages/model-binding)**. Model binding is a process that maps form values to server-side code automatically, and converts the strings coming in from the `Request.Form` collection to the type represented by the server-side target. Targets can be handler method parameters or public properties on a PageModel.

Handler method parameters

The following example shows how to revise the `OnPost` handler method so that the `emailAddress` input value is bound to a handler method parameter:

```
public void OnPost(string emailAddress)
{
    // do something with emailAddress
}
```

And here is how the handler code would be modified to work with a public property:

```
[BindProperty]
public string EmailAddress { get; set; }
public void OnPost()
{
    // do something with EmailAddress
}
```

The property to be included in model binding must be decorated with the `BindProperty` attribute.

Further information

- Model Binding (/razor-pages/model-binding)

Tag Helpers

The `form`, `input`, `select` and `textarea` elements are all targets of **Tag helpers (/razor-pages/tag-helpers)**, components that extend the HTML element to provide custom attributes which are used to control the HTML generation.

The most important attribute is the `asp-for` attribute that takes the name of a PageModel property. This results in the correct `name` attribute being generated so that form values are bound correctly to the model when the form is posted back to the server.

In the previous example, the `EmailAddress` property is passed to the `input` tag helper as follows:

```
<input asp-for="EmailAddress" />
```

The resulting HTML is as follows:

```
<input type="text" id="EmailAddress" name="EmailAddress" value="" />
```

Further information

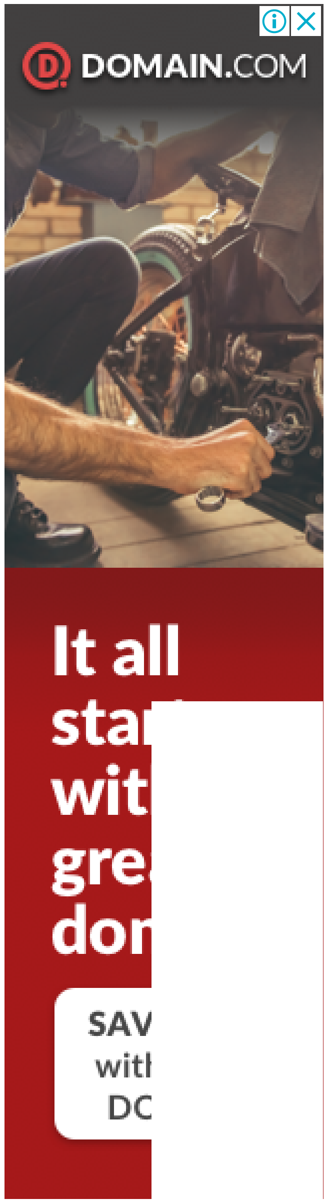
- Tag helpers (/razor-pages/tag-helpers)

Request Verification

The Razor Pages framework includes security as a feature. When you add a `<form>` element with a `method` attribute set to `post`, an additional hidden form field is generated for the purposes of validating that the form post originated from the same site. This process is known as Request Verification. Although not advisable, you can turn this feature off. You can read more about why this safety check is included and how to manage it here (/security/request-verification).

Last updated: 28/10/2020 16:34:48

On this page



Latest Updates

The Razor _Layout.cshtml file (https://www.learnrazorpages.com/razor-pages/files/layout)

An Introduction To ASP.NET Core Razor Pages (https://www.learnrazorpages.com/)

Publishing and deploying a Razor Pages application to IIS on Windows (https://www.learnrazorpages.com/publishing/publish-to-iis)

Implementing a Custom Model Binder In Razor Pages (https://www.learnrazorpages.com/advanced/custom-model-binder)

Checkboxes in a Razor Pages Form (https://www.learnrazorpages.com/razor-pages/forms/checkboxes)

User Input Validation in Razor Pages (https://www.learnrazorpages.com/razor-pages/validation)