

# Validating User Input in ASP.NET Web Pages

Visual Studio 2010

You can add input validation to ASP.NET Web pages using validation controls. Validation controls provide an easy-to-use mechanism for all common types of standard validation—for example, testing for valid dates or values within a range—plus ways to provide custom-written validation. In addition, validation controls allow you to customize how error information is displayed to the user.

Validation controls can be used with any controls you put on an ASP.NET Web page, including both HTML and Web server controls. For more information, see [ASP.NET Web Server Controls Overview](#).

## Security Note:

By default, ASP.NET Web pages automatically check for potentially malicious input. For more information, see [Script Exploits Overview](#).

## Using Validation Controls

You enable validation of user input by adding validation controls to your page as you would add other server controls. There are controls for different types of validation, such as range checking or pattern matching. For a complete list of validation types, see [Types of Validation for ASP.NET Server Controls](#). Each validation control references an input control (a server control) elsewhere on the page. When user input is being processed (for example, when a page is submitted), the validation control tests the user input and sets a property to indicate whether the entry passed the test. After all of the validation controls have been called, a property on the page is set indicating whether any validation check has failed.

Validation controls can be associated into validation groups so that validation controls belonging to a common group are validated together. You can use validation groups to selectively enable or disable validation for related controls on a page. Other validation operations, such as displaying a [ValidationSummary](#) control or calling the [GetValidators](#) method, can reference the validation group.

You can test the state of the page and of individual controls in your own code. For example, you would test the state of the validation controls before updating a data record with information entered by the user. If you detect an invalid state, you bypass the update. Typically, if any validation checks fail, you skip all of your own processing and return the page to the user. Validation controls that detect errors then produce an error message that appears on the page. You can display all validation errors in one place using a [ValidationSummary](#) control.

## Note:

Data-bound controls that update, insert, and delete data, such as the [GridView](#), [FormView](#), and [DetailsView](#) controls, automatically verify that validation checks have passed before performing a data update operation.

## When Validation Occurs

Validation controls perform input checking in server code. When the user submits a page to the server, the validation controls are invoked to check the user input, control by control. If a validation error is detected in any of the input

controls, the page itself is set to an invalid state so you can test for validity before your code runs. Validation occurs after page initialization (that is, after view state and postback data have been processed) but before any change or click event handlers are called.

If the user is working with a browser that supports ECMAScript (Javascript), the validation controls can also perform validation using client script. This can improve response time in the page because errors are detected immediately and error messages are displayed as soon as the user leaves the control containing the error. If client-side validation is available, you have greater control over the layout of error messages and can display an error summary in a message box. For more information, see [Client-Side Validation for ASP.NET Server Controls](#).

ASP.NET performs validation on the server even if the validation controls have already performed it on the client, so that you can test for validity within your server-based event handlers. In addition, re-testing on the server helps prevent users from being able to bypass validation by disabling or changing the client script check.

You can invoke validation in your own code by calling a validation control's **Validate** method. For more information, see [How to: Validate Programmatically for ASP.NET Server Controls](#).

## Validating for Multiple Conditions

Each validation control typically performs one test. However, you might want to check for multiple conditions. For example, you might want to specify both that a user entry is required and that the user entry is limited to accepting dates within a specific range.

You can attach more than one validation control to an input control on a page. In that case, the tests performed by the controls are resolved using a logical **AND** operator, which means that the data entered by the user must pass all of the tests in order to be considered valid.

In some instances, entries in several different formats might be valid. For example, if you are prompting for a phone number, you might allow users to enter a local number, a long-distance number, or an international number. Using multiple validation controls would not work in this instance because the user input must pass all tests to be valid. To perform this type of test—a logical **OR** operation where only one test must pass—use the [RegularExpressionValidator](#) validation control and specify multiple valid patterns within the control. Alternatively, you can use the [CustomValidator](#) validation control and write your own validation code.


## Displaying Error Information

Validation controls are not normally visible in the rendered page. However, if the control detects an error, it displays the error message text that you specify. The error message can be displayed in a variety of ways, as listed in the following table.

Display method	Description
Inline	Each validation control can individually display an error message in place (usually next to the control where the error occurred).
Summary	<p>Validation errors can be collected and displayed in one place—for example, at the top of the page. This strategy is often used in combination with displaying a message next to the input fields with errors. If the user is working in Internet Explorer 4.0 or later, the summary can be displayed in a message box.</p> <p>If you are using validation groups, you need a <a href="#">ValidationSummary</a> control for each separate group.</p>

In place and summary	The error message can be different in the summary and in place. You can use this option to show a shorter error message in place with more detail in the summary.
Custom	You can customize the error message display by capturing the error information and designing your own output.

If you use the in-place or summary display options, you can format the error message text using HTML.


 **Security Note:**

If you create custom error messages, make sure that you do not display information that might help a malicious user compromise your application. For more information, see [How to: Display Safe Error Messages](#).

## Validation Object Model

You can interact with validation controls by using the object model that is exposed by individual validation controls and by the page. Each validation control exposes its own **IsValid** property that you can test to determine whether a validation test has passed or failed for that control. The page also exposes an **IsValid** property that summarizes the **IsValid** state of all the validation controls on the page. This property allows you to perform a single test to determine whether you can proceed with your own processing.

The page also exposes a [Validators](#) collection containing a list of all the validation controls on the page. You can loop through this collection to examine the state of individual validation controls.

 **Note:**

There are a few differences in the object model for client-side validation. For more information, see [Client-Side Validation for ASP.NET Server Controls](#).

## Customizing Validation

You can customize the validation process in the following ways:

- You can specify the format, text, and location of error messages. In addition, you can specify whether the error messages appear individually or as a summary.
- You can create custom validation using [CustomValidator](#) control. The control calls your logic but otherwise functions like other validation controls in setting error state, displaying error messages, and so on. This provides an easy way to create custom validation logic while still using in the validation framework of the page.
- For client-side validation, you can intercept the validation call and substitute or add your own validation logic.

## See Also

Other Resources

---

# Community Additions

---