

## Lesson 4-8: Use the RequiredFieldValidator control

You now know how to work with the controls needed to create almost any online form. One thing you'll often want to do with online forms is to validate the user's input to make sure they are entering everything correctly.

Although you could validate input using a combination of C# and JavaScript, ASP.NET provides a series of controls that will automatically generate the validation code for you. The *RequiredFieldValidator* is one of these.

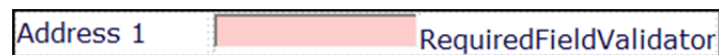
1 Open *ShiningStone* from your sample files folder.

2 Open *buy.aspx* in *Design* view.

This is the form a visitor to the site would use to buy a product. It isn't finished yet, and one thing it needs is validation.

3 Add a *RequiredFieldValidator* control to the page.

Drag a *RequiredFieldValidator* control from the *Validation* category of the *Toolbox* so that it is just after the textbox following *Address 1*.



A *RequiredFieldValidator* simply makes sure that the user has entered something into the control it validates. In this case you're going to make it validate that text has been entered into the *TextBoxAddress1* control.

4 Set the properties of the new *RequiredFieldValidator* control.

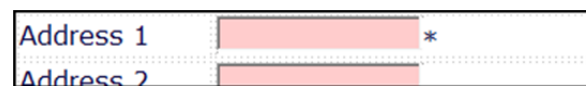
1. Set the *ID* property to: **RequiredFieldValidatorAddress1**

2. Set the *Text* property to: \*

This is the message that will appear if the control fails to validate, ie if the user doesn't fill in the field.

3. Set the *ControlToValidate* property to: **TextBoxAddress1**

The *ControlToValidate* property tells the validator which control it is going to check.

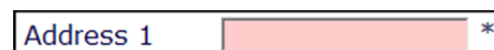


5 See *RequiredFieldValidator* in action.

1. View *buy.aspx* in your browser.

2. Click *Submit Order* without entering anything into any of the text boxes.

You will see that the page doesn't post back and your \* text appears where you placed your *RequiredFieldValidator*.



3. Type something into the *Address 1* text box.

### note

#### Validation groups

You already know that an ASP.NET page is implemented using one (and only one) form that typically includes all page content.

This can cause a problem.

Imagine you had a log-in section at the top of your web page containing two fields: *User Name* and *Password*.

When the user tried to log in, (in this example), validation would fail because *Address 1* was blank.

Validation groups solve this problem.

All validation controls have a *ValidationGroup* property which is blank by default. This means that, by default, all controls belong to the same Validation Group.

If you wished to associate some form controls with a submit button (in this example the *User Name* and *Password*) you would set the *ValidationGroup* property of these controls to the same value. When the user clicked the submit button control, validation rules would then be applied only to the *User Name* and *Password*.

4. Click *Submit Order*.

This time the page posts back without problems.

5. Close the web browser.

## 6 Validate using C#.

Although the messages you've seen displayed so far are very fast and helpful, the web server implements them using JavaScript. A malicious user could bypass your validation by simply disabling JavaScript.

Fortunately, you can use C# to very easily check that the user hasn't somehow bypassed your validation rules.

1. Open the code-behind file of *buy.aspx*.
2. Add the following code to the *ButtonSubmitOrder\_Click* event handler:

**if (Page.IsValid) Response.Write("OK!");**

```
protected void ButtonSubmitOrder_Click(object sender, EventArgs e)
{
    if (Page.IsValid) Response.Write("OK!");
}
```

This will re-check the user input against all validators on the page and only write *OK!* if all validation rules succeed. This is a very useful security enhancement.

This code uses an *if* statement. You'll learn all about them in: *Lesson 7-1: Use the if statement.*

3. View the page in your browser.
4. Enter some text into the *Address1* text box.

5. Click *Submit Order*.

You'll see that the page submits and the C# test succeeds. If you'd disabled JavaScript to try to bypass the validation, the C# test wouldn't succeed and you wouldn't see *OK!*.

## 7 Close Visual Studio.