

SLIDE 1

Input validation.

What is it?

Input validation is the process of checking the data entered by the user before that data is saved to the database.

It can be used to give users guides on how to enter valid data.

Validating user input before it is written to the database can improve the quality of the data stored in your tables, make your program more user friendly, and increase the speed at which users can enter valid data.

SLIDE 2

Almost every field in your database requires some type of input validation.

Before designing your data input form, you should consider the following questions regarding your data fields:

1. What characters are valid or invalid for the field?
Do you want only numeric data? Do you want uppercase letters only?
2. For numeric fields, is there a high/low range limit?
If you have an order form, the minimum order might be 10 widgets. If the user input 5 widgets, you would not have a valid order.
3. Is there a specific list of valid values for the field?
If you have a customer type field, and you only deal with Retail, Wholesale, and Distributor customers, then providing a list for the user would almost guarantee a valid input.
4. Must data be entered into a field?
Are there any required fields on the form? Almost every input form has at least one required field.
5. Is this a conditional field?
For instance, if the user indicates on a form that the customer wishes to have his purchase shipped instead of picked up, the shipping address would be a conditional field.

SLIDE 3

Even if you take all these ideas into consideration, you cannot assume that the user will input exactly what you, or more importantly, what your database is expecting.

This is why input validation is essential.

SLIDE 4

There are 2 types of input validation.

First is Field Level validation.

This kind of validation occurs at the keypress event.

This means that as the user types in the input, the validation is occurring.

Some types of Field Level Validation include

1. Keyboard filtering
2. Input masking
3. Validation lists

I will explain these in detail in a moment.

The second type of Input Validation is Form Level Validation.

This type is performed at the time the user clicks an OK, SAVE, or UPDATE control. The program will alert the user if any data is not valid, or if anything is missing.

Some types of Form Level Validation include:

1. High/Low ranges
2. Min/Max Field Lengths
3. Required Fields
4. Conditional Fields

SLIDE 5

Our first Field Level Validation is Keyboard Filtering.

Using a Text Box as your input control, you can automatically prohibit the user from inputting invalid data.

For instance, in the first section of code, we are limiting the input in txtNumber control to the numbers 012345678 and 9. If the user tries to enter anything that is not one of these numbers, the keypress returns an KeyAscii value of zero, which is the same as if the user never touched the keyboard.

In the if statement, the part “If ASCII > 26” means that you are also allowing the user to have access to keys such as Escape, Return, and Backspace. The ASCII values below 26 represent the keys like the Escape, Return, Arrows, Page Up/Page Down, Backspace, etc.

In the second section of code, we are using this same idea, but limiting the input to the alphabet.

In the third section of code, we allow uppercase letters and numbers.

SLIDE 6

Input Masking allows the form to display data in a familiar or more readable format. Some examples would be social security number, invoice number, birthdate, etc. I have included phone number and currency on my sample form.

For Input Masking, we use a control called MaskedTextBox, which looks like this.

To format a phone number, in the properties window we set the MASK property to (###) ###-#### which formats our data on the form to look like a standard phone number. The database sees this as ten digits with no spaces.

We have also changed the PromptInclude property to False. You should always set this to false when your MaskedTextBox is a bound control.

With the MASK property, the user actually sees the format of the field as he is entering numbers.

For the mskDollars control, I used this code so that only numbers would be accepted as valid input.

In the properties window, the MASK property was left blank, and the FORMAT property was set to \$#,##0.00;(\$#,##0.00). The FORMAT property does not display the format in the field until the user has entered the data and moved on to the next field.

(I didn't understand why, but the book I was using said you would get a database error when you tried to update this record without setting this to false.)

SLIDE 7

Validation Lists

Here you use the familiar ComboBox control.

These are very helpful if you have a set number of selections that would represent valid input from your user.

This is like our earlier example of having 3 customer types – Retail, Wholesale, and Distributor. You are almost guaranteed that the user will give you valid data when presented with one of these.

They work best when your list of possibilities is relatively small.

I have coded the selections at Form_Load, but they can also be filled at run time.

SLIDE 8

Now we have reached FORM LEVEL VALIDATION

Sometimes you will have a field with a maximum and minimum acceptable value.

In my code example here, I have specified that acceptable values in a specific TextBox are between 1 and 100.

If the user inputs a number outside this range, and tries to update the database, they will get a warning message box, be reminded of the acceptable range, and are instructed to try again.

SLIDE 9

This code works like the High/Low Values example, except that it tests the input's length to see if it is within predetermined parameters. Here I set the minimum length at 3 characters and the maximum at 10 characters. If the user inputs a name with too few or too many characters, they will get a warning message box, be reminded of the acceptable length, and are instructed to try again.

SLIDE 10

Most forms you will encounter will have at least one required field.

Here is some code that will alert the user if they attempt to update the database without a value in the txtRequired field.

Notice the use of the TRIM function, which removes any leading or trailing spaces. This assures that the user cannot skip this validation step by typing a few spaces into this field.

SLIDE 11

Finally, we have our Conditional Field.

Let's assume that you are completing an order form for a customer, and they asked that you ship the product to them instead of picking it up themselves.

If you click the SHIP TO ADDRESS" OptionButton, without typing an address into the ADDRESS, CITY, STATE, and ZIP fields, the user will be alerted.

Even if only one of the fields is left blank.