

## **REFACTORING IN C#**

Refactoring is the process of changing the code structure after we complete the writing of the code to increase readability and easy maintenance of the code. Refactoring is done by changing the internal structure of the code without changing the external behavior of the code block.

In 'C#' following Refactoring commands are present under the refactoring menu :-

- Extract Method
- Rename
- Encapsulate Field
- Extract Interface
- Promote Local Variable to Parameter
- Remove Parameters
- Reorder Parameters

### **Extract Method**

Extract Method is a refactoring command that is used to create a new method from an existing code fragment. By using this command you can create a new method by extracting a selected portion of the code block. The new method contains the selected code in its block. The selected code segment from the parent block is replaced by the function call to this new function. The new method is inserted below the source code block in the same class. The signature of the new method is determined by the Extract Method. For example if the new method doesn't have any reference to the instance data then this method is declared as "static". This signature is automatically generated by the refactor engine based on the selected code block.

#### **Advantages:**

- 1- Reduces duplication of code.
- 2- Easy for the maintenance of the code.

### **Rename**

Rename is a refactoring command that facilitates to rename fields, local variables, methods, namespaces, properties and types. It can be used to change names in the comment and in the strings. We can apply the renaming operation on the following code symbols:

- Fields
- Local variable
- Method
- Namespace
- Property
- Type

**NOTE-1:** When an extension method is renamed then all instances of the method which are in the same scope are renamed to the new method name. It does not matter whether the method is used as static or as instance method.

**NOTE-2:** When renaming the namespace the visual studio also renames the default namespace property of the Application page.

**NOTE-3:** When renaming a method if we check the "Rename overloads" option in the rename dialog box then the methods of the base class which are overloaded in the child class are also renamed.

**NOTE-4:** While renaming a member which is implemented/overrides, visual studio display a dialog box saying that the renaming operation causes cascading updation.

### **Encapsulate Field**

Encapsulate Field is a refactoring command that is used to create a property from an existing field. It can update the code to have the references to the new property if the "Update References" option is checked in the Encapsulate Field dialog box. It changes the access specifier of the field from public to private and then generates get and set accessors for that field. If that field is marked as read only then only get accessor is created.

**NOTE-1:** The encapsulate field operation is done only for the field on which the cursor is present.

**NOTE-2:** Under the "Update References" option there are two options like: External and All

If we choose the external option then only the external references to the field for which we are creating the property are changed, not the internal reference ones.

If we choose the all option the all the references to the field are changed.

### **Extract Interface**

Extract Interface is a refactoring command used to create new interfaces from an existing class, structure or interface. The new interface is created as a new file and the cursor is positioned at the beginning of that new file. We can provide the name for the new interface and can also select the members from which we want to create the interface by using the "Extract Interface dialog box". If we extract an interface from a class or structure then the refactor engine automatically create a reference to the new interface.

### **Promote Local Variable to Parameter**

It a refactoring command using which we can convert a local variable of a function to its parameter. The local variable must be initialize with some value or expression. When we promote a local variable to a parameter then the new parameter of the function is added to the end of the parameter list of the function. Any call to the function is modified to reflect the change.

## **Remove Parameters**

It is a refactoring command which is used to remove a parameter from a methods, indexes, or delegates. It can make changes at the declaration of the member and any other places where the member is called.

**NOTE-1:** If a parameter being removed is modified during the call to a method, then the parameter is not modified anymore.

**NOTE-2:** Removal of a parameter from a method does not remove the reference of that parameter from the body of that method. Hence it will cause a build error. You have to manually remove these references.

**NOTE-3:** You cannot remove the first parameter of an extension method

## **Reorder Parameters**

It is a refactoring command which is used to change the sequence of the parameter list a methods, indexes or delegates. It can make changes at the declaration of the member and any other places where the member is called.

**NOTE-1:** You cannot reorder the first parameter of an extension method.

## **Advance Features of Refactoring**

### **1. Preview Changes Dialog Box**

Most of the refactoring operations provide a way to preview your changes in the code before committing them. For this there is an option in the refactoring dialog box which is appearing for every type of refactoring . If you select this option then when you do any refactoring operation then it displays the changes in a dialog box. In the preview dialog box there are two views. The upper view display the current code that is going to be changed and the bottom view display the changed code. It has also two buttons - Apply: For accepting the changes, Cancel: For discarding the changes.

### **2. Multi-Project Refactoring**

Visual Studio supports multi-project refactoring for projects that are in the same solution. It can apply the refactoring operation on all the files having correct references across all projects of the same language.

### **3. Error-Tolerant Refactoring**

The refactoring operation can be applied on a project which cannot be build. It can be done because the refactoring operation is done to change the structure of the code not the behavior of the code. But if the project has incorrect references then the refactoring operation may not be applied correctly.

### **References to Bing more**

- <http://msdn.microsoft.com/en-us/library/719exd8s.aspx>
- [http://msdn.microsoft.com/en-us/library/719exd8s\(VS.100\).aspx](http://msdn.microsoft.com/en-us/library/719exd8s(VS.100).aspx)
- <http://www.devsource.com/c/a/Using-VS/Refactoring-in-Visual-C-2005/>

### **External tools for Refactoring**

[http://devexpress.com/Products/Visual\\_Studio\\_Add-in/Refactoring/](http://devexpress.com/Products/Visual_Studio_Add-in/Refactoring/)

\*\*\*\*