100 XP

Refactor code using GitHub Copilot Inline Chat

6 minutes

GitHub Copilot's Inline Chat feature allows you to refactor code directly in the code editor. Inline chat is useful when you want to make changes to your code without having to switch to the Chat view.

Understand your code

Before you refactor your code, it's important to understand the code you're working with. You should understand the purpose of the code, how it works, and any dependencies it may have. If you're not familiar with the code, take some time to read through it and understand its structure and logic. Using GitHub Copilot to analyze the code can help you identify areas for improvement and suggest refactoring options.

You can use the Inline Chat feature to ask questions about the code, get explanations of specific parts, or request suggestions for improvements. For example, you might ask GitHub Copilot to explain how a particular function works or suggest ways to optimize it. You can also use the Inline Chat feature to ask for help with specific coding tasks, such as refactoring a function or improving code readability.

Here are some examples of prompts that ask GitHub Copilot for help with code:

plaintext

```
/explain Explain how authentication is implemented in this code
/explain Can this code be updated to improve security? Explain the con-
cepts and show some examples
```

If the explanation is long or complex, you can move to the Chat view to continue the conversation. You can also use the Chat view to ask follow-up questions or request additional information.

Use Inline Chat to refactor code

Use the Inline Chat feature when you want to refactor a section of code that performs a specific task. For more complex changes that involve updating multiple files or projects, use one of the Chat view modes.

To refactor code using Inline Chat, follow these steps:

1. Open the file that contains the code you want to optimize.
2. Select the code that you want to optimize.
3. Open an Inline Chat session.

   Use the **Ctrl+I** keyboard shortcut to open the Inline Chat session. Alternatively, you can select **Editor Inline Chat** from one of the GitHub Copilot menus.
4. Enter a prompt that asks for an optimization.

   For example: **Can this code be improved?**
5. Review the suggested code update.

   Copilot applies the suggested code update directly in the editor, where you can review it in-place, with the full context of the surrounding code.
6. Select **Accept** or **Close** to accept or discard the suggested code update.

   As with all Copilot suggestions, check that the revised code runs without errors and produces the correct result.

The following example demonstrates how to use Inline Chat to refactor a C# method that processes data.

The original method performs data cleansing, constructs a data table, and then prints the contents of the table. Combining several operations into one method makes it harder to maintain and test your code.

You can refactor the original method by asking Inline Chat to create separate methods for each of the tasks performed in the original method.

Here's the original code:

C#

```
using System;
using System.Data;

public class DataProcessor
{
```

```
    public void ProcessData(string item, string price)
    {
        // Cleanse the data
        item = item.Trim();    // Remove leading and trailing whitespace
        price = price.Trim();  // Remove leading and trailing whitespace
        double priceValue = double.Parse(price);   // Convert price to a
double
        // More cleansing operations can be added here

        // Create and print a DataTable
        DataTable table = new DataTable();
        table.Columns.Add("Item", typeof(string));
        table.Columns.Add("Price", typeof(double));
        table.Rows.Add(item, priceValue);
        foreach (DataRow row in table.Rows)
        {
            Console.WriteLine($"Item: {row["Item"]}, Price:
{row["Price"]}");
        }
    }
}

// Example usage
class Program
{
    static void Main()
    {
        string item = "  Apple  ";
        string price = "  1.50  ";
        var processor = new DataProcessor();
        processor.ProcessData(item, price);
    }
}
```

To refactor the ProcessData method, follow these steps:

1. Select the ProcessData method.
2. Open inline chat.
3. Enter a prompt that asks Copilot to create separate methods for each task, and then press **Enter**.

   For example: **I want to refactor the ProcessData method. I want a method for data cleansing, a method for printing, and a method for data processing.**
4. Review the suggested code update.

   For this example, Copilot suggests a refactored version of the original method that separates the cleansing and printing tasks into separate methods.

C#

```
using System;
using System.Data;

public class DataProcessor
{
    public void ProcessData(string item, string price)
    {
        (string cleansedItem, double cleansedPrice) =
CleanseData(item, price);
        DataTable table = new DataTable();
        table.Columns.Add("Item", typeof(string));
        table.Columns.Add("Price", typeof(double));
        table.Rows.Add(cleansedItem, cleansedPrice);
        PrintData(table);
    }

    private (string, double) CleanseData(string item, string price)
    {
        string trimmedItem = item.Trim();
        string trimmedPrice = price.Trim();
        double priceValue = double.Parse(trimmedPrice);
        return (trimmedItem, priceValue);
    }

    private void PrintData(DataTable table)
    {
        foreach (DataRow row in table.Rows)
        {
            Console.WriteLine($"Item: {row["Item"]}, Price:
{row["Price"]}");
        }
    }
}

// Example usage
class Program
{
    static void Main()
    {
        string item = "  Apple  ";
        string price = "  1.50  ";
        var processor = new DataProcessor();
        processor.ProcessData(item, price);
    }
}
```

In this example, refactoring the original code improves modularity and makes the code easier to understand.

5. Select **Accept** or **Close** to accept or discard the suggested code update.

In this example, the suggested code looks good, so you would select **Accept** to apply the refactored code to your project.

As with all Copilot suggestions, check that the revised code runs without errors and produces the correct result.

6. Test the refactored code to ensure it runs without errors and generates the expected result.

Summary

GitHub Copilot's Inline Chat feature allows you to refactor code directly in the code editor. Inline chat is useful when you want to make changes to your code without having to switch to the Chat view. You can use Inline Chat to ask for help with specific coding tasks, such as refactoring a function or improving code readability. You can also use Inline Chat to evaluate your existing code before refactoring.

Next unit: Refactor code using GitHub Copilot Chat modes

Previous Next