Extension methods are a powerful feature in C# and .NET that allow you to add new methods to existing types without modifying them. Here's a simple example of using extension methods in a real-time application with .NET Core.  
  
Let's consider a scenario where you have a DateTime object, and you want to format it in a specific way for logging purposes throughout your application. Instead of repeating the formatting code everywhere, you can create an extension method for DateTime to handle the formatting consistently.  
  
RealTime Application use of Extension  
  
1. Request Validation:  
2. Logging:  
3. Response Formatting:  
4. Middleware:  
5. Authentication and Authorization:

**How Extension Methods Work**

* Extension methods are defined as **static methods** in a **static class**.
* The first parameter of the method specifies the type it extends, prefixed with the this keyword.
* Once defined, the method can be called as if it were an instance method of the extended type.

using System;

public static class StringExtensions

{

// Extension method to reverse a string

public static string ReverseString(this string str)

{

if (string.IsNullOrEmpty(str)) return str;

char[] charArray = str.ToCharArray();

Array.Reverse(charArray);

return new string(charArray);

}

}

class Program

{

static void Main(string[] args)

{

string original = "Hello, World!";

string reversed = original.ReverseString();

Console.WriteLine($"Original: {original}");

Console.WriteLine($"Reversed: {reversed}");

}

}

**Key Points About Extension Methods**

1. **Static Class Requirement**:
   * The class containing the extension method must be static.
2. **Calling Extension Methods**:
   * They can be called like instance methods on the type they extend.
   * The compiler resolves them as static method calls under the hood.
3. **Chaining**:
   * You can chain extension methods to create a fluent API.
4. **Namespace Inclusion**:
   * To use an extension method, the namespace of the static class must be included.
5. **Precedence**:
   * If a type has a method with the same name as an extension method, the instance method takes precedence.

**Advantages of Extension Methods**

* Non-intrusive: Add functionality without modifying existing code.
* Enhance built-in or third-party types.
* Useful for utility functions and fluent APIs.
* Promotes clean and reusable code.