C# XML Documentation Example

C# XML documentation example

XML documentation in C# is used to provide structured comments for classes, methods, properties, and other elements in the code. These comments can later be processed to generate API documentation. Below is an example:

Example: XML Documentation in C#

csharp

```csharp
using System;

namespace ExampleNamespace
{
    /// <summary>
    /// Represents a basic calculator for arithmetic operations.
    /// </summary>
    public class Calculator
    {
        /// <summary>
        /// Adds two numbers.
        /// </summary>
        /// <param name="a">The first number.</param>
        /// <param name="b">The second number.</param>
        /// <returns>The sum of <paramref name="a"/> and <paramref name="b"/>.</returns>
        public int Add(int a, int b)
        {
            return a + b;
        }

        /// <summary>
        /// Subtracts one number from another.
        /// </summary>
        /// <param name="a">The number to subtract from.</param>
        /// <param name="b">The number to subtract.</param>
        /// <returns>The result of <paramref name="a"/> minus <paramref name="b"/>.</returns>
        public int Subtract(int a, int b)
        {
            return a - b;
        }

        /// <summary>
        /// Multiplies two numbers.
        /// </summary>
        /// <param name="a">The first number.</param>
        /// <param name="b">The second number.</param>
        /// <returns>The product of <paramref name="a"/> and <paramref name="b"/>.</returns>
        public int Multiply(int a, int b)
        {
            return a * b;
        }

        /// <summary>
        /// Divides one number by another.
        /// </summary>
        /// <param name="a">The numerator.</param>
        /// <param name="b">The denominator.</param>
        /// <returns>The quotient of <paramref name="a"/> divided by <paramref name="b"/>.</returns>
        /// <exception cref="DivideByZeroException">Thrown when <paramref name="b"/> is zero.</exception>
        public int Divide(int a, int b)
        {
            if (b == 0)
                throw new DivideByZeroException("Denominator cannot be zero.");
            return a / b;
        }
    }

    class Program
    {
        static void Main()
        {
            Calculator calc = new Calculator();
            Console.WriteLine("Addition: " + calc.Add(10, 5));
            Console.WriteLine("Subtraction: " + calc.Subtract(10, 5));
            Console.WriteLine("Multiplication: " + calc.Multiply(10, 5));
            Console.WriteLine("Division: " + calc.Divide(10, 5));
        }
    }
}
```

Explanation:

- `` `<summary>` ``: Provides a brief description of the class, method, or property.
- `` `<param>` ``: Describes a parameter for a method or constructor.
- `` `<returns>` ``: Specifies the return value of the method.
- `` `<exception>` ``: Indicates possible exceptions the method might throw.
- `` `<paramref>` ``: Highlights the parameter within documentation for better readability.

Generating Documentation:

- Enable XML documentation file generation in Visual Studio:
    1. Right-click the project in **Solution Explorer** > **Properties**.
    2. Go to the **Build** tab.
    3. Check the **XML documentation file** option.

This will generate an XML file containing the comments, which can be used to create detailed documentation.

same code with javadoc+java

Here's the same functionality as the C# example, written in Java using Javadoc for documentation:

Example: Javadoc in Java

```java
java


/**
 * Represents a basic calculator for arithmetic operations.
 */
public class Calculator {

    /**
     * Adds two numbers.
     *
     * @param a the first number
     * @param b the second number
     * @return the sum of {@code a} and {@code b}
     */
    public int add(int a, int b) {
        return a + b;
    }

    /**
     * Subtracts one number from another.
     *
     * @param a the number to subtract from
     * @param b the number to subtract
     * @return the result of {@code a} minus {@code b}
     */
    public int subtract(int a, int b) {
        return a - b;
    }

    /**
     * Multiplies two numbers.
     *
     * @param a the first number
     * @param b the second number
     * @return the product of {@code a} and {@code b}
     */
    public int multiply(int a, int b) {
        return a * b;
    }

    /**
     * Divides one number by another.
     *
     * @param a the numerator
     * @param b the denominator
     * @return the quotient of {@code a} divided by {@code b}
     * @throws ArithmeticException if {@code b} is zero
     */
    public int divide(int a, int b) {
        if (b == 0) {
            throw new ArithmeticException("Denominator cannot be zero.");
        }
        return a / b;
    }

    public static void main(String[] args) {
        Calculator calc = new Calculator();
        System.out.println("Addition: " + calc.add(10, 5));
        System.out.println("Subtraction: " + calc.subtract(10, 5));
        System.out.println("Multiplication: " + calc.multiply(10, 5));
        System.out.println("Division: " + calc.divide(10, 5));
    }
}
```

Explanation:

1. **Class Documentation (`` `/**` ``):**
    - Provides an overview of the class.

2. **Method Documentation (`` `@param` ``, `` `@return` ``, `` `@throws` ``)**:
    - `` `@param` ``: Describes the method parameters.
    - `` `@return` ``: Explains the return value of the method.
    - `` `@throws` ``: Specifies exceptions that might be thrown by the method.
3. **Inline Code Tags**:
    - `` `{@code}` ``: Used to format code elements inline in the documentation.

Generating Javadoc:

1. In an IDE like IntelliJ IDEA or Eclipse, you can generate Javadoc by selecting the **Tools** menu and choosing **Generate Javadoc**.
2. On the command line:
    - Use the `` `javadoc` `` tool.
    - Example: `` `javadoc -d docs Calculator.java` ``
    - This generates the documentation in the `` `docs` `` directory.

The generated Javadoc can be viewed in a browser and provides a structured, easy-to-read documentation of the code.