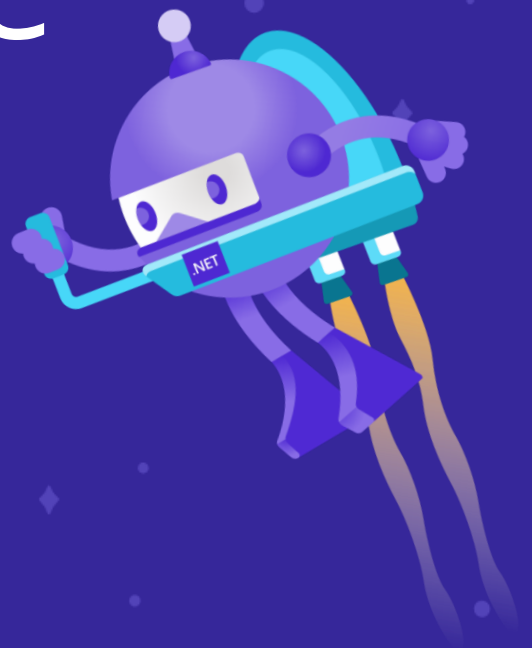
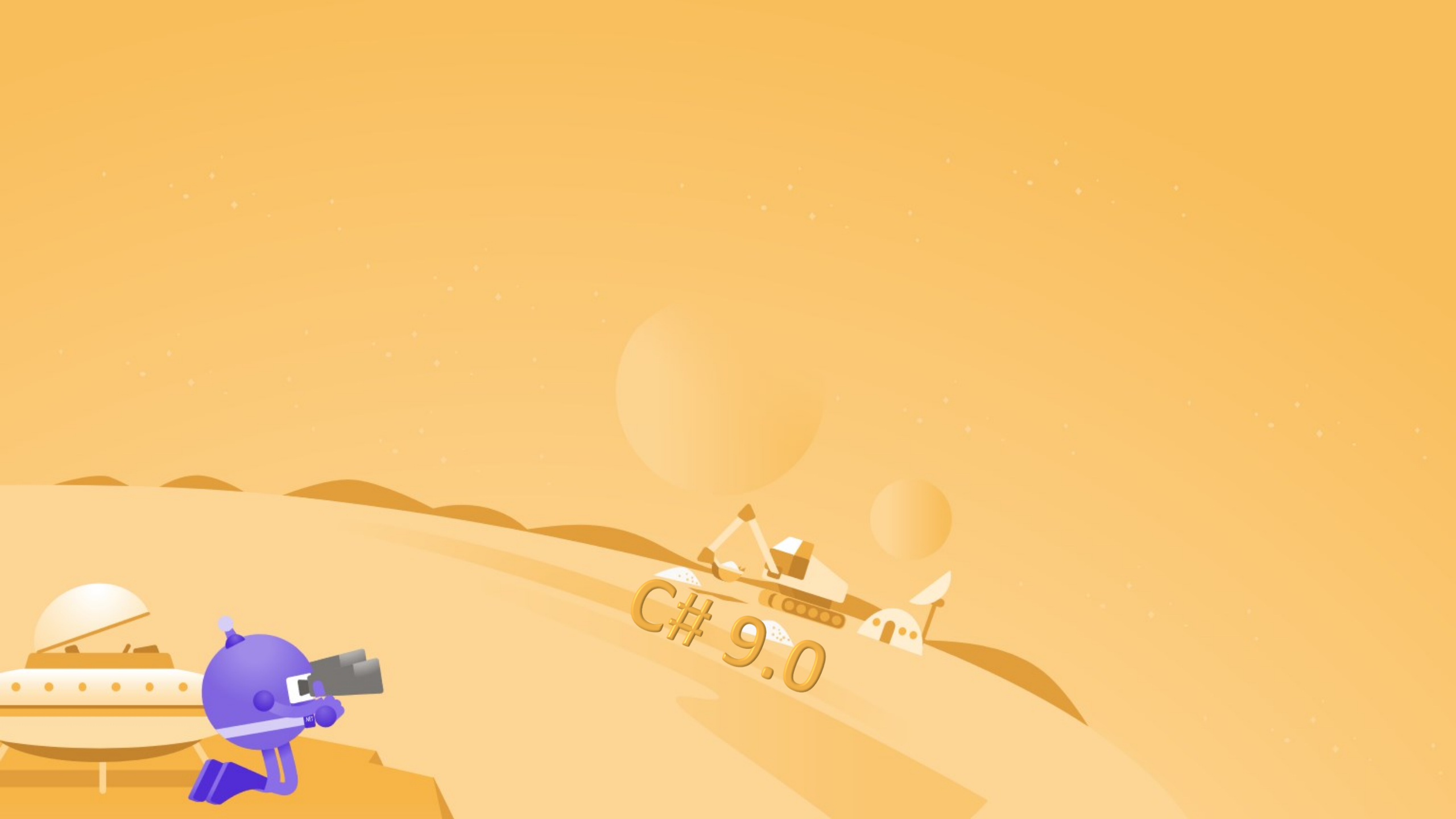
Built-in string data type methods in C#

Billy Vanegas




C# 9.0

Helper Methods in C#

- IndexOf()

- Reports the zero-based index of the first occurrence of a specified Unicode character or string within this instance.
- The method returns -1 if the character or string is not found in this instance.

Step 1 - Write code to find an opening and closing parenthesis embedded in a string

```
string message = "Find what is (inside the parentheses)";

int openingPosition = message.IndexOf('(');
int closingPosition = message.IndexOf(')');

Console.WriteLine(openingPosition);
Console.WriteLine(closingPosition);
```

```
13
36
```

Helper Methods in C#

- Substring()

- Retrieves a substring from this instance.

Step 2 - Add code to retrieve the value between two parenthesis characters

```
string message = "Find what is (inside the parentheses)";

int openingPosition = message.IndexOf('(');
int closingPosition = message.IndexOf(')');

// Console.WriteLine(openingPosition);
// Console.WriteLine(closingPosition);

int length = closingPosition - openingPosition;
Console.WriteLine(message.Substring(openingPosition, length));
```

(inside the parentheses

Step 3 - Update the code to modify the starting position of the sub string

```
string message = "Find what is (inside the parentheses)";

int openingPosition = message.IndexOf('(');
int closingPosition = message.IndexOf(')');

openingPosition  += 1;

int length = closingPosition - openingPosition;
Console.WriteLine(message.Substring(openingPosition, length));
```

inside the parentheses

Avoid Magic Values

- Hardcoded strings
- Hardcoded numeric

- Instead use Constants

Using Constant to avoid unexpected values

```
string message = "What is the value <span>between the tags</span>?";

const string openSpan = "<span>";
const string closeSpan = "</span>";

int openingPosition = message.IndexOf(openSpan);
int closingPosition = message.IndexOf(closeSpan);

openingPosition  += openSpan.Length;

int length = closingPosition - openingPosition;
Console.WriteLine(message.Substring(openingPosition, length));
```

between the tags

Helper Methods in C#

- LastIndexOf()

- Reports the zero-based index position of the last occurrence of a specified Unicode character or string within this instance.
- The method returns -1 if the character or string is not found in this instance.

Step 4 - Write code to retrieve the last occurrence of a sub string

```
string message = "(What if) I am (only interested) in the last (set of parentheses)?";
int openingPosition = message.LastIndexOf('(');

openingPosition  += 1;
int closingPosition = message.LastIndexOf(')');


int length = closingPosition - openingPosition;
Console.WriteLine(message.Substring(openingPosition, length));
```

set of parentheses

Step 5 - Update the code example to retrieve any value between one or more sets of parentheses in a string

```
string message = "(What if) I am (only interested) in the last (set of parentheses)?";
while(true) {
    int openingPosition = message.IndexOf('(');
    if (openingPosition == -1) break;
    openingPosition  += 1;
    int closingPosition = message.IndexOf(')');
    int length = closingPosition - openingPosition;
    Console.WriteLine(message.Substring(openingPosition, length));
    //Note how we use the overload of Substring to return only the remaining
    // unprocessed message:
    message = message.Substring(closingPosition +1)
}
```

What if
more than
set of parentheses

Helper Methods in C#

- IndexOfAny()

- Reports the index of the first occurrence in this instance of any character in a specified array of Unicode characters.
- The method returns -1 if the characters in the array are not found in this instance.

Step 6 - Update the code example to work with different types of symbol sets

```
string message = "(What if) I have [different symbols] but every {open symbol} needs a [matching closing symbol]?";
char[] openSymbols = { '[', '{', '(' };
int closingPosition = 0;
while (true)
{
    int openingPosition = message.IndexOfAny(openSymbols, closingPosition);
    if (openingPosition == -1) break;
    string currentSymbol = message.Substring(openingPosition, 1);
    char matchingSymbol = ' ';
    switch (currentSymbol)    {
        case "[":
            matchingSymbol = ']';
            break;
        case "{":
            matchingSymbol = '}';
            break;
        case "(":
            matchingSymbol = ')';
            break;    }
    openingPosition += 1;
    closingPosition = message.IndexOf(matchingSymbol, openingPosition);
    int length = closingPosition - openingPosition;
    Console.WriteLine(message.Substring(openingPosition, length));
}
```

What if
different symbols
open symbol
matching closing symbol

Recap

- IndexOf() gives us the first position of a character or string inside of another string.
- IndexOf() returns -1 if it can't find a match.
- Substring() returns just the specified portion of a string, using a starting position and optional length.
- LastIndexOf() returns the last position of a character or string inside of another string.
- IndexOfAny() returns the first position of an array of char that occurs inside of another string.
- There's often more than one way to solve a problem. We used two separate techniques to find all instances of a given character or string.
- Avoid hardcoded magic values. Instead, define a const variable. A constant variable's value can't be changed after initialization.

Helper Methods in C#

- Remove()

- Returns a new string in which a specified number of characters from the current string are deleted.

Step 1 - Write code to remove characters in specific locations from a string

```
string data = "12345John Smith          5000  3 ";
string updateData = data.Remove(5,20);
Console.WriteLine(updateData);
```

```
123455000 3
```

Step 2 - Write code to remove characters no matter where they appear in a string

```
string message = "This--is--ex-amp-le--da-ta";
message = message.Replace("--"," ");
message = message.Replace("-"," ");
Console.WriteLine(message);
```

This is example data

Recap

- The Remove() method works like the Substring() method, except that it deletes the specified characters in the string.
- The Replace() method swaps all instances of a string with a new string.

Challenge No. 1

In this challenge, you'll work with a string that contains a fragment of HTML. You'll extract data from the HTML fragment, replace some of its content, and remove other parts of its content to achieve the desired output.

- Step 1: Delete all of the code in the .NET Editor from the earlier exercise.
- Step 2: Write code in the .NET Editor to extract data, replace data, and remove data from an input string.

Thanks!