Efficient String Manipulation Using StringBuilders

Steve Gordon

.NET Engineer and Microsoft MVP

@stevejgordon www.stevejgordon.co.uk

Overview

- Getting started with the StringBuilder type
- Apply benchmarks to measure performance
- Learn about the implementation details
- Optimize use of StringBuilders
- Learn about regex multiline mode
- Append and remove data using StringBuilders

Reminder:

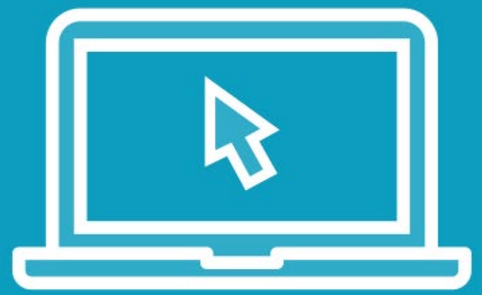
Strings are immutable. Operations which appear to modify them allocate new string instances.

Efficient String Manipulation

Consider optimizing string manipulation on hot paths of heavy-load applications.

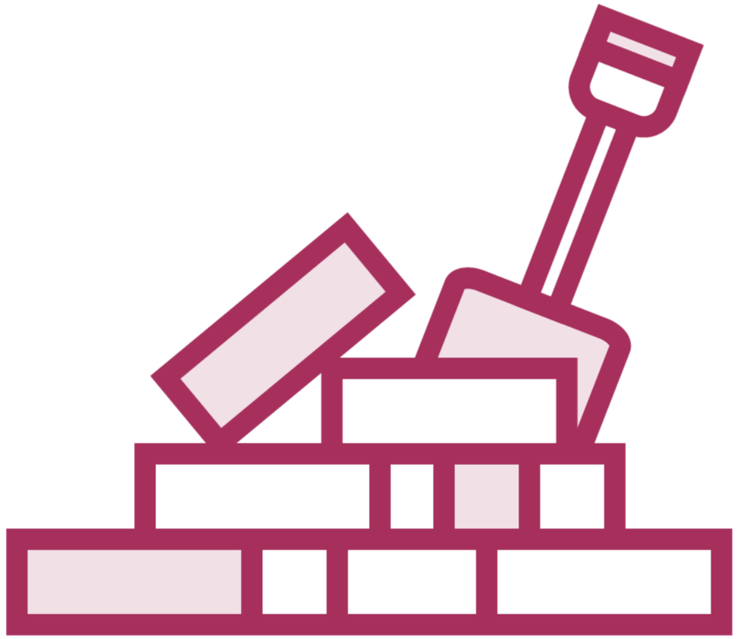
Profile your application under expected load before, during and after making changes.

Demo



Use a StringBuilder to concatenate strings

StringBuilder

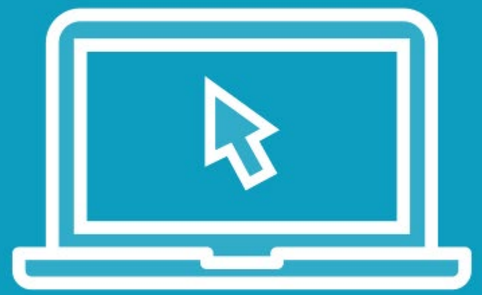


Modify text data without allocating additional string objects

Provides a way to work a with mutable sequence of characters

Improves performance when concatenating many strings, such as within unbounded loops

Demo



Benchmark performance of StringBuilders

- Reimplement SalesDataWriter to compare allocations

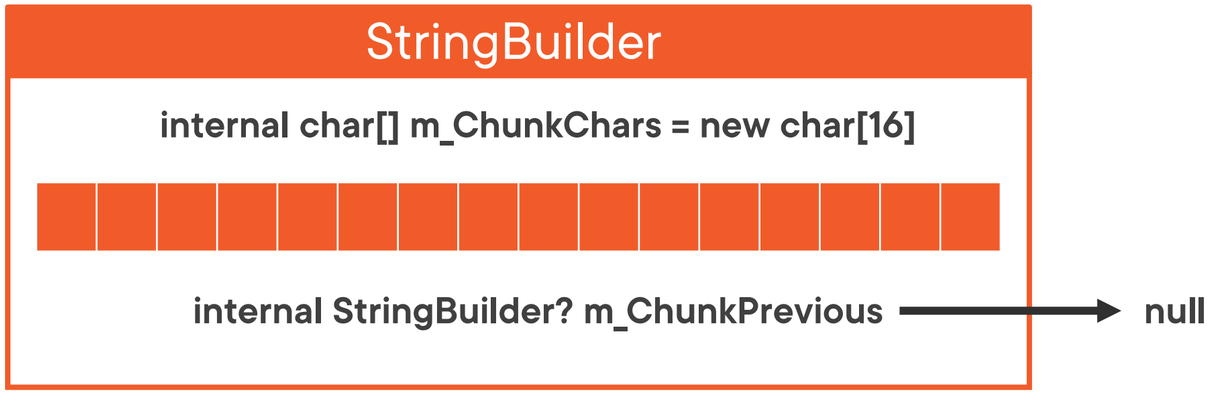
StringBuilder Implementation Details

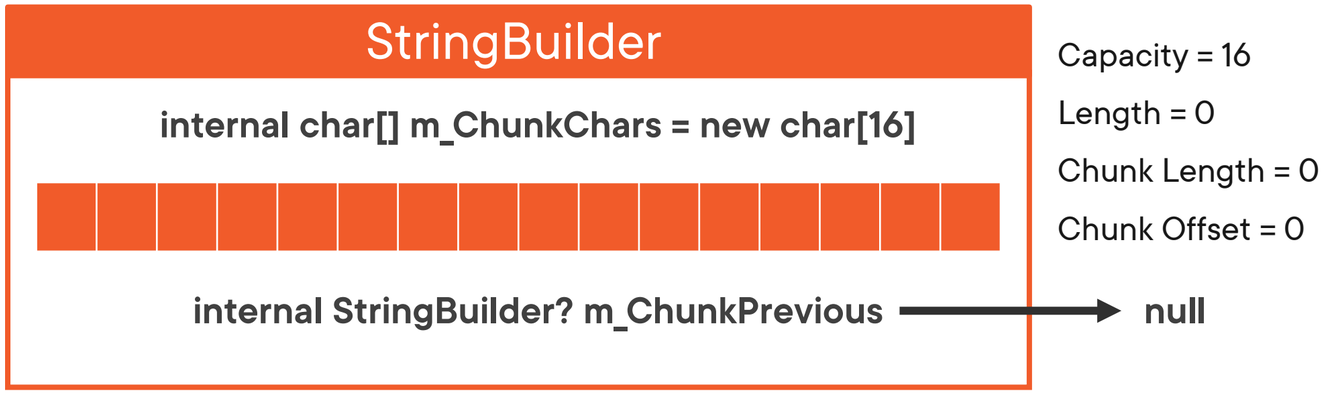
StringBuilder Internals

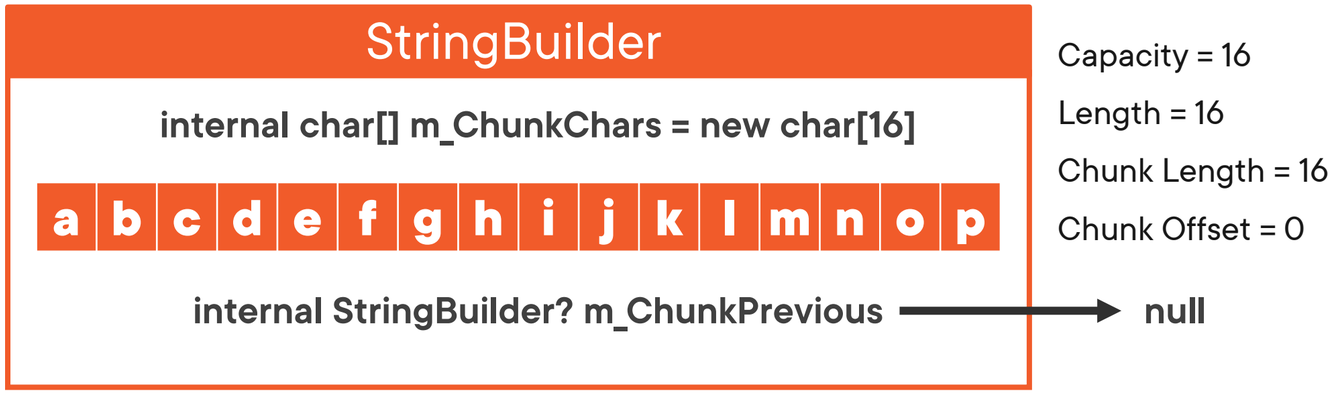
The StringBuilder class provides a way to mutate text data

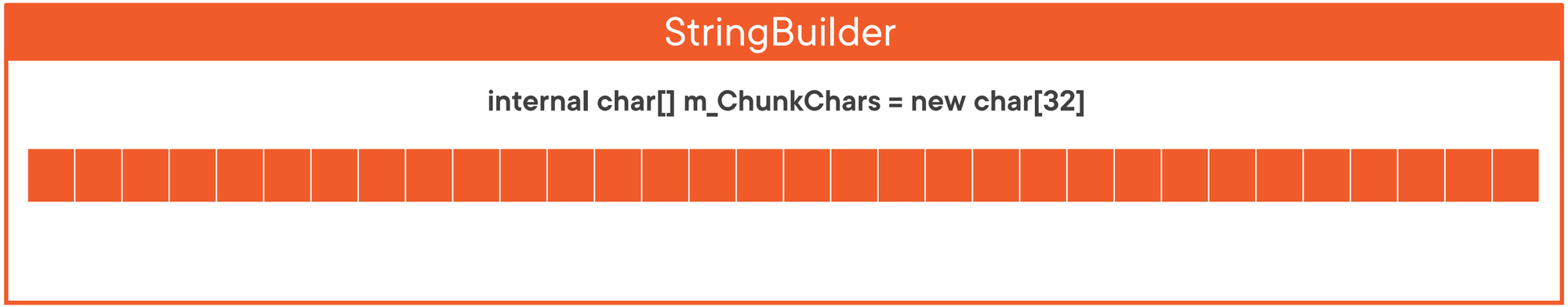
Maintains an internal buffer of characters

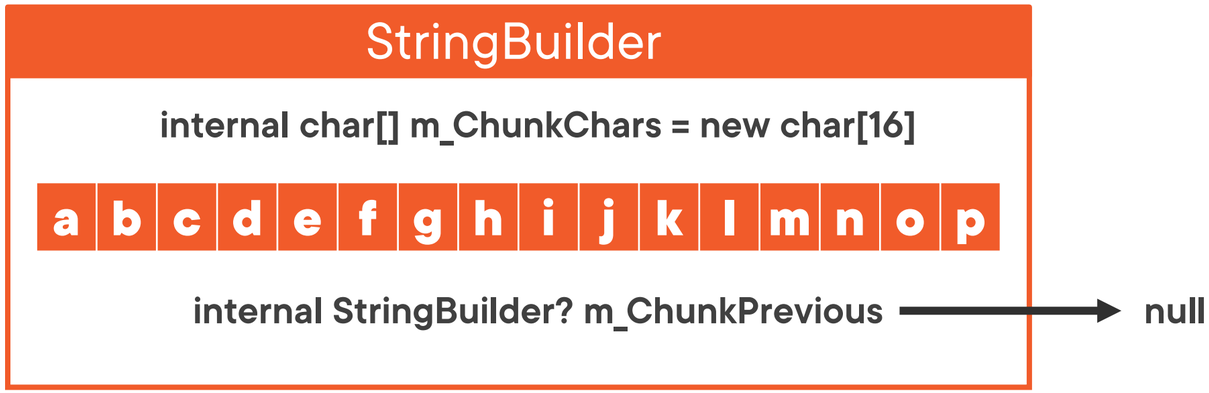
Can ‘grow’ by referencing a new StringBuilder with an increased (doubled) buffer capacity

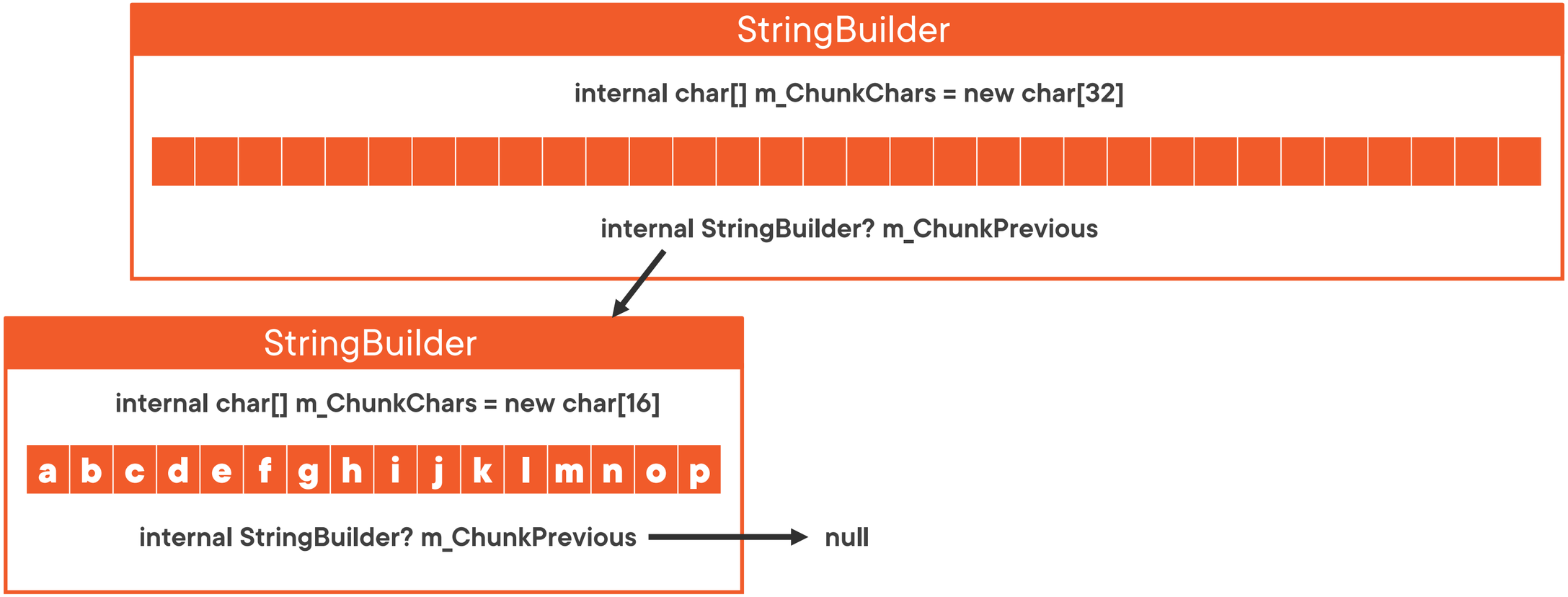

StringBuilder
internal char[] m_ChunkChars = new char[16]
internal StringBuilder? m_ChunkPrevious
null

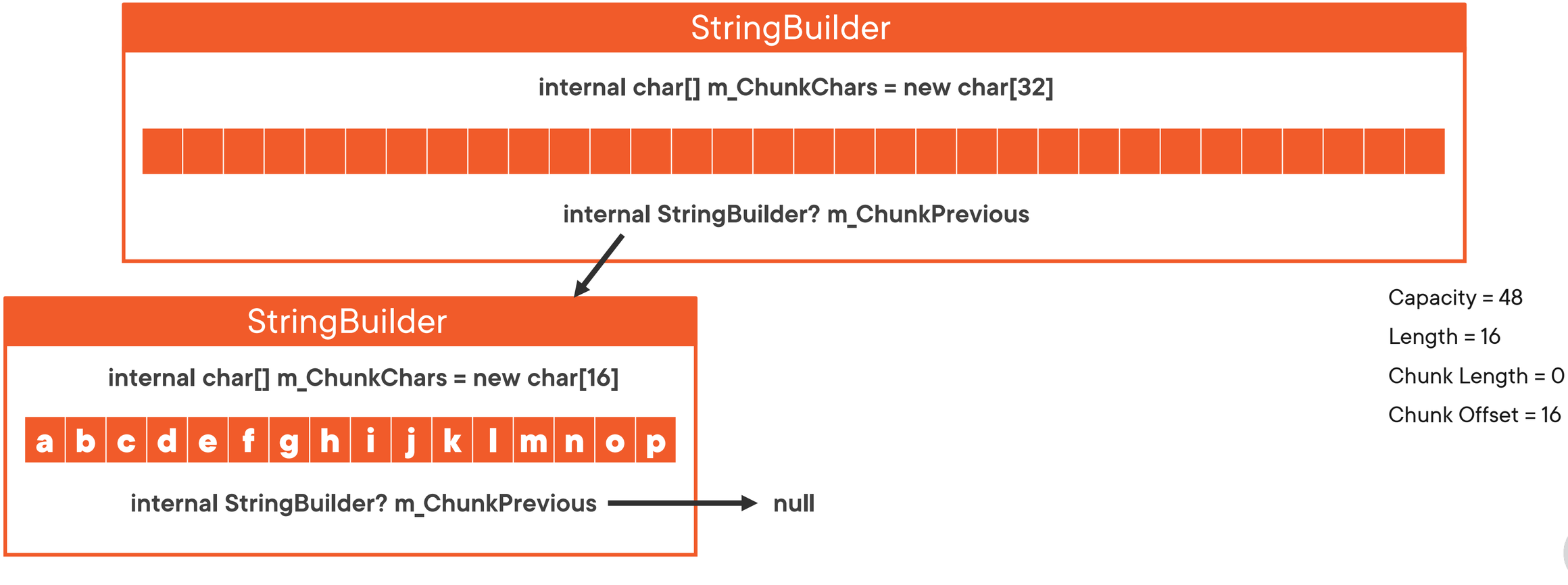

StringBuilder
internal char[] m_ChunkChars = new char[16]
internal StringBuilder? m_ChunkPrevious
null
Capacity = 16
Length = 0
Chunk Length = 0
Chunk Offset = 0

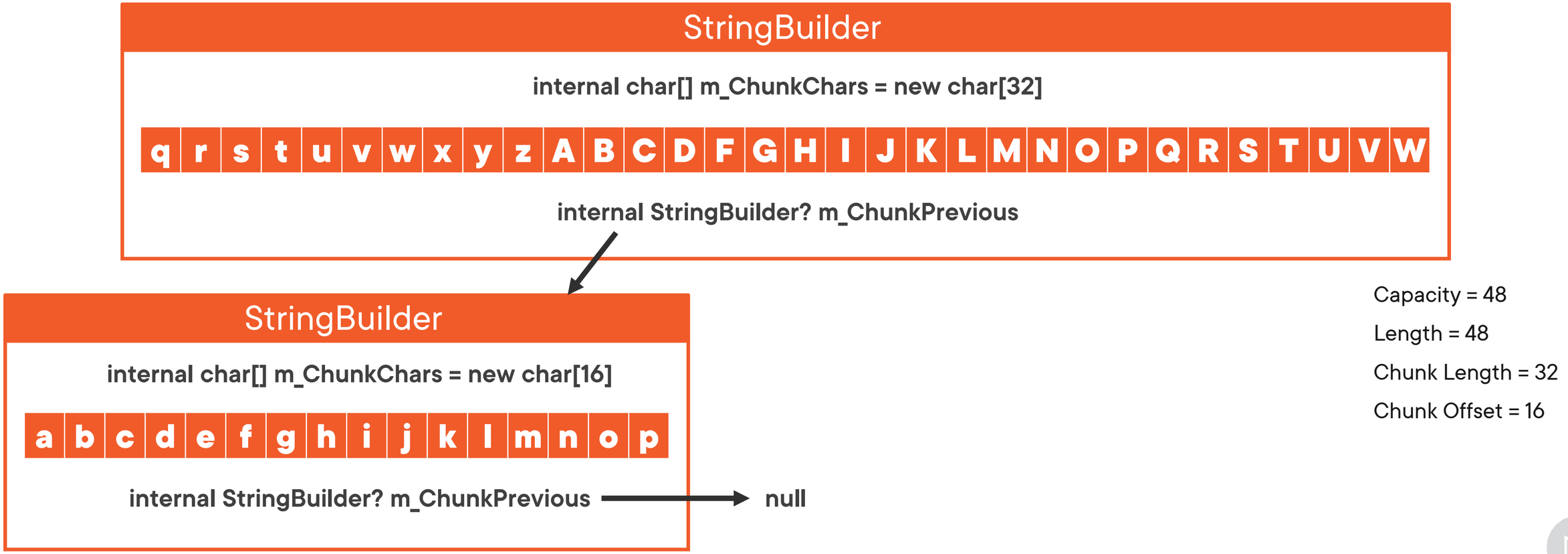

StringBuilder
internal char[] m_ChunkChars = new char[16]
a b c d e f g h i j k l m n o p
internal StringBuilder? m_ChunkPrevious
null
Capacity = 16
Length = 16
Chunk Length = 16
Chunk Offset = 0

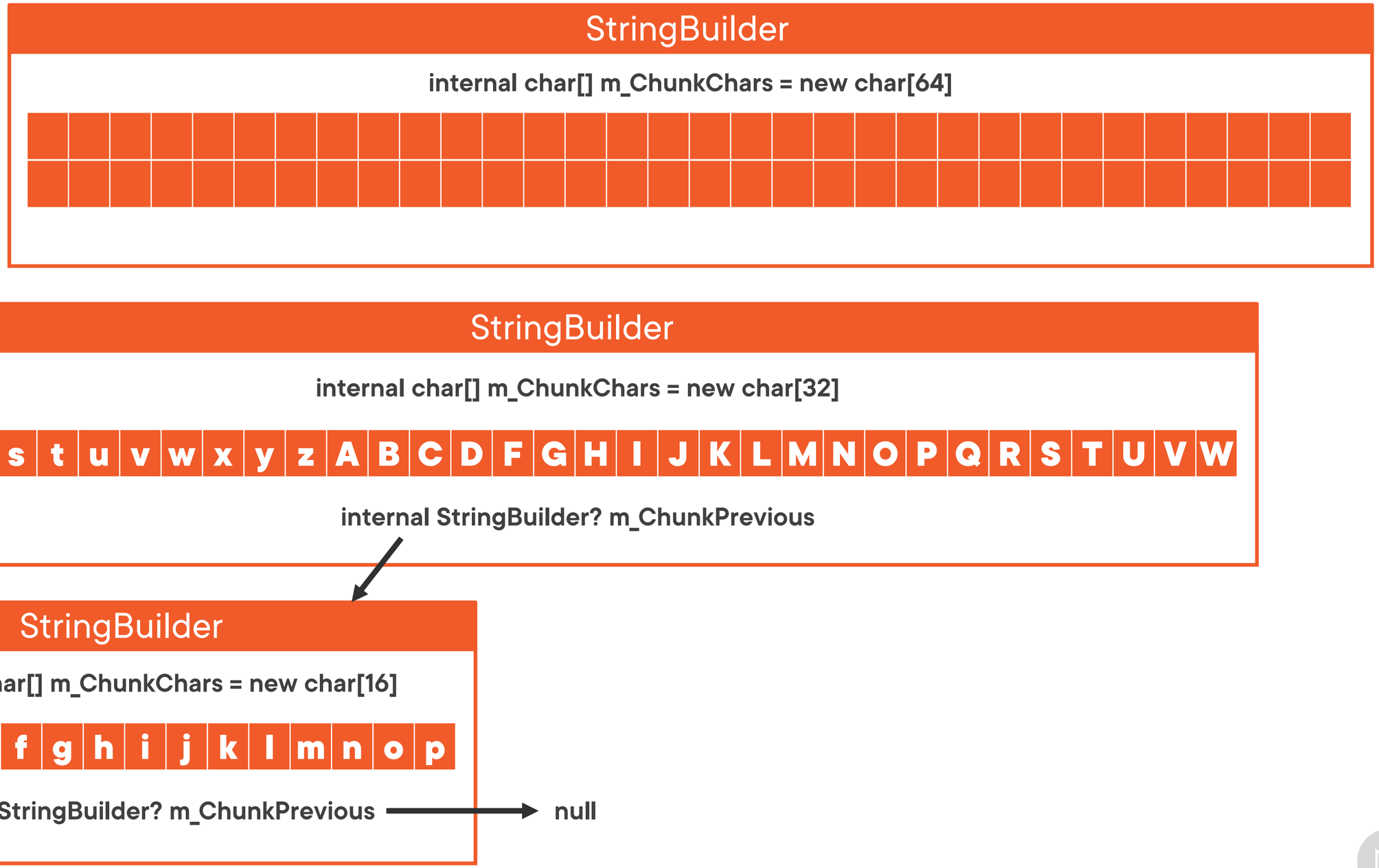

StringBuilder
internal char[] m_ChunkChars = new char[32]

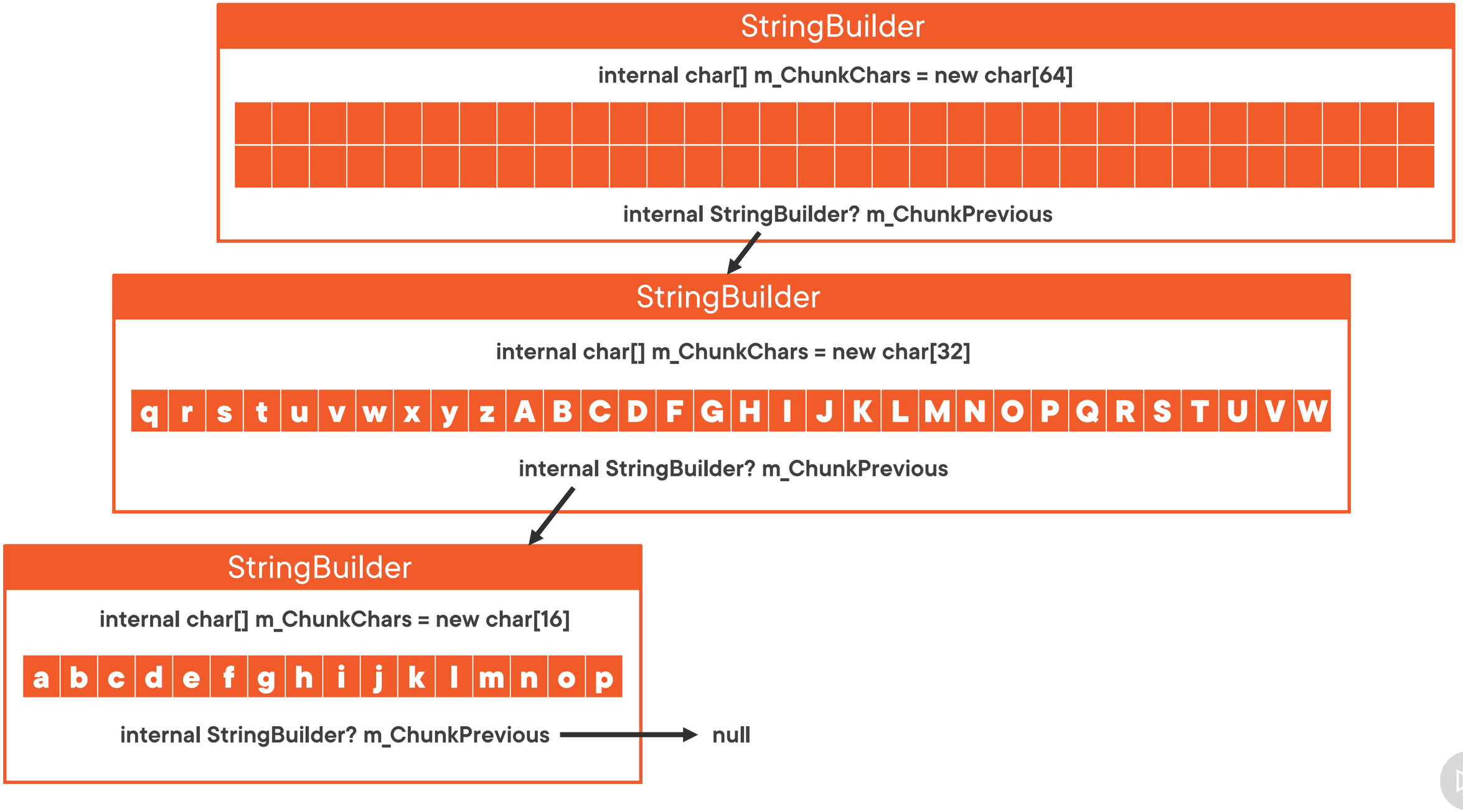

StringBuilder
internal char[] m_ChunkChars = new char[16]
a b c d e f g h i j k l m n o p
internal StringBuilder? m_ChunkPrevious
null


StringBuilder
internal char[] m_ChunkChars = new char[32]
internal StringBuilder? m_ChunkPrevious
StringBuilder
internal char[] m_ChunkChars = new char[16]
a b c d e f g h i j k l m n o p
internal StringBuilder? m_ChunkPrevious
null


StringBuilder
internal char[] m_ChunkChars = new char[32]
internal StringBuilder? m_ChunkPrevious
StringBuilder
internal char[] m_ChunkChars = new char[16]
a b c d e f g h i j k l m n o p
internal StringBuilder? m_ChunkPrevious
null
Capacity = 48
Length = 16
Chunk Length = 0
Chunk Offset = 16


StringBuilder
internal char[] m_ChunkChars = new char[32]
q r s t u v w x y z A B C D F G H I J K L M N O P Q R S T U V W
internal StringBuilder? m_ChunkPrevious
StringBuilder
internal char[] m_ChunkChars = new char[16]
a b c d e f g h i j k l m n o p
internal StringBuilder? m_ChunkPrevious
null
Capacity = 48
Length = 48
Chunk Length = 32
Chunk Offset = 16


StringBuilder
internal char[] m_ChunkChars = new char[64]
StringBuilder
internal char[] m_ChunkChars = new char[32]
q r s t u v w x y z A B C D F G H I J K L M N O P Q R S T U V W
internal StringBuilder? m_ChunkPrevious
StringBuilder
internal char[] m_ChunkChars = new char[16]
a b c d e f g h i j k l m n o p
internal StringBuilder? m_ChunkPrevious
null

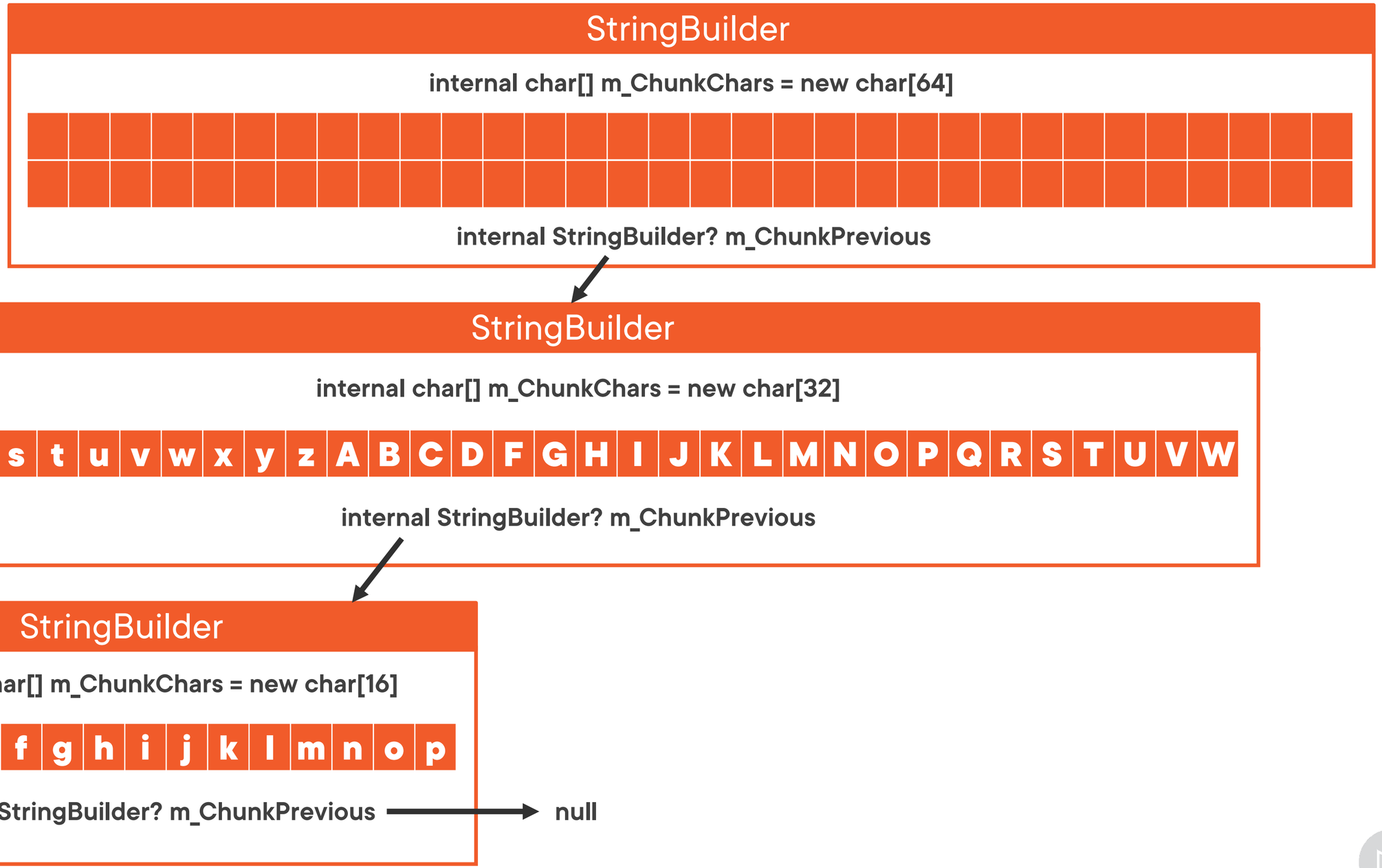

StringBuilder
internal char[] m_ChunkChars = new char[64]
internal StringBuilder? m_ChunkPrevious
StringBuilder
internal char[] m_ChunkChars = new char[32]
q r s t u v w x y z A B C D F G H I J K L M N O P Q R S T U V W
internal StringBuilder? m_ChunkPrevious
StringBuilder
internal char[] m_ChunkChars = new char[16]
a b c d e f g h i j k l m n o p
internal StringBuilder? m_ChunkPrevious
null

Capacity = 112

Length = 48

Chunk Length = 0

Chunk Offset = 48

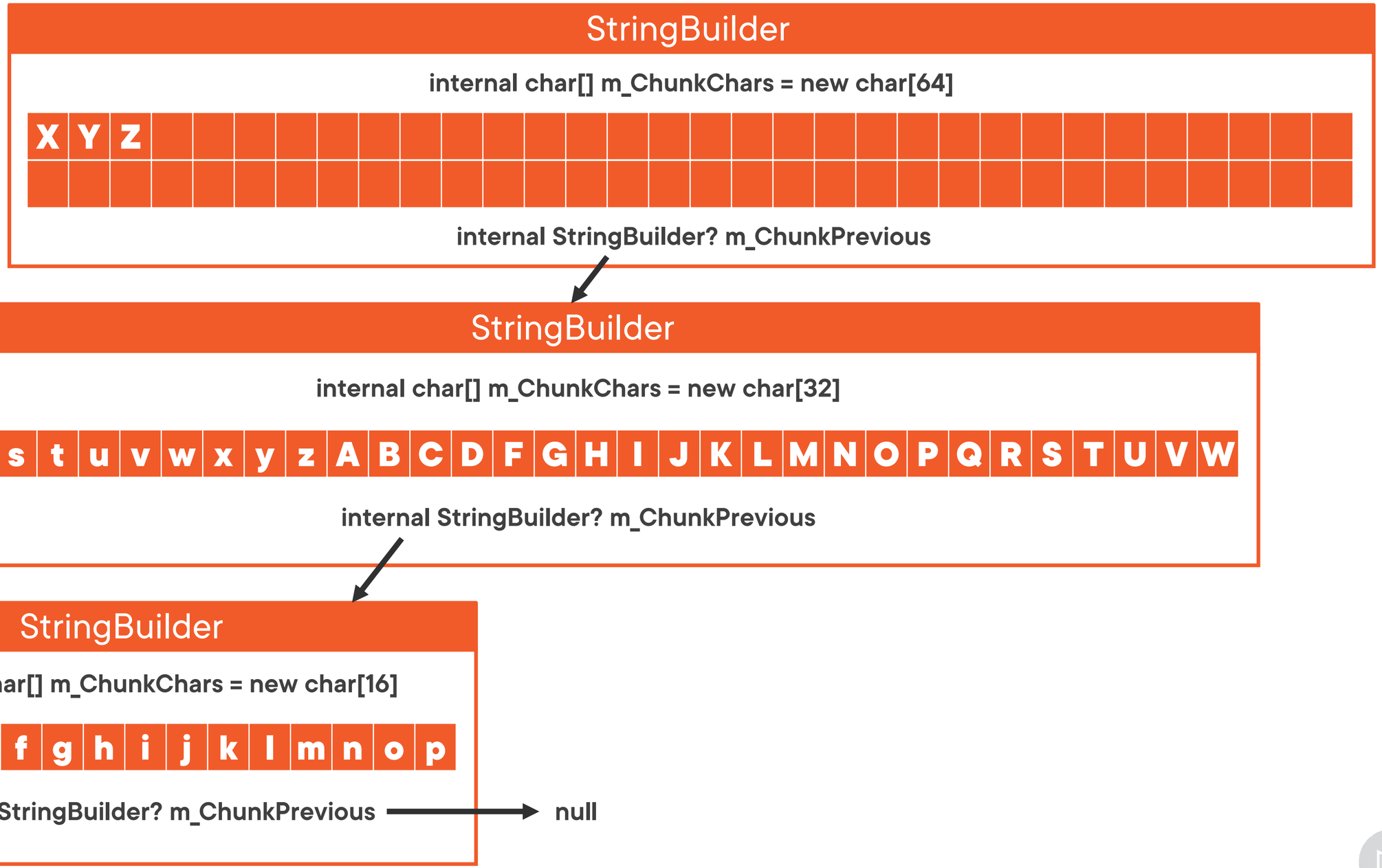


Capacity = 112

Length = 52

Chunk Length = 3

Chunk Offset = 48



```
var myString = stringBuilder.ToString()
```

Create the Final String

This traverses the linked list of StringBuilders to return the final string from the occupied characters in all chunks.

```
var length = stringBuilder.Length
```

Length Property

Get the current length of the StringBuilder including all previous chunks. Calculated using the chunk offset and the chunk length.

```
stringBuilder.MaxCapacity = 256;
```

MaxCapacity Property

The maximum capacity the StringBuilder is allowed to have.

For performance reasons, the implementation is a little strange. When appending lots of small strings, it may grow beyond MaxCapacity. Eventually it will throw a ArgumentOutOfRangeException if you attempt to append data which would enlarge the instance beyond its configured capacity.

```
var stringBuilder = new StringBuilder(2048);
```

Create a StringBuilder with an Initial Capacity

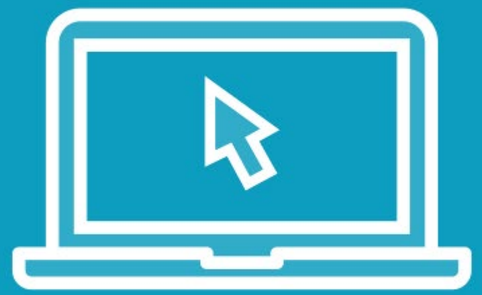
This code creates a StringBuilder with an internal char array size of 2048

```
stringBuilder.Capacity = 4096;
```

Capacity Property

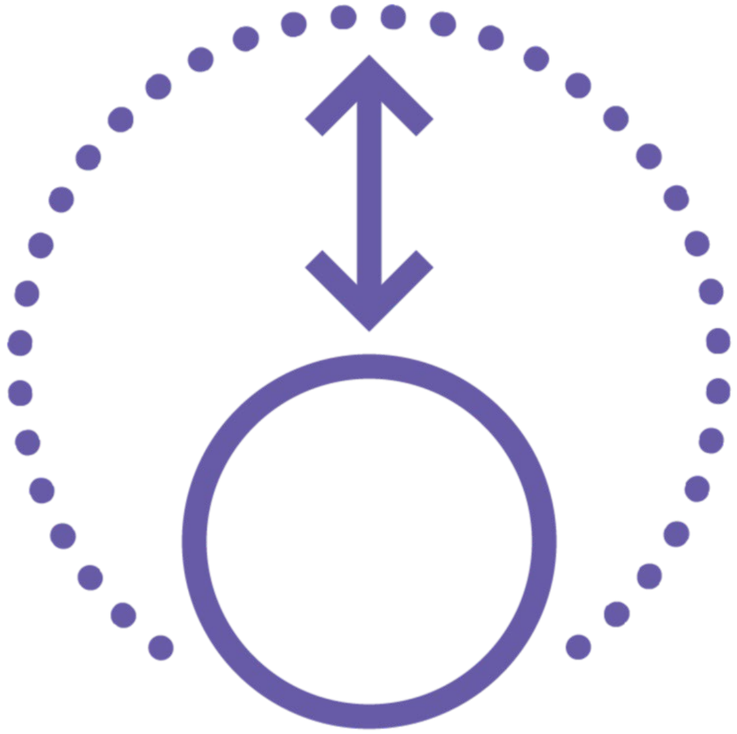
Sets the number of characters that can be contained in the memory allocated by the StringBuilder instance (chunk).
This creates a new array of the required capacity and copies any existing characters from the existing array.

Demo



Provide a capacity to StringBuilder instances

Specifying a Capacity



When capacity is reached, the StringBuilder must ‘grow’ to accommodate more data

This process causes additional allocations

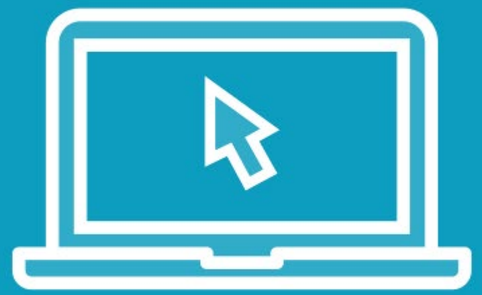
Sometimes we want to avoid this performance overhead

We can achieve this by providing an initial capacity to correctly size the character buffer

Warning!

It’s always recommended that you benchmark you code under realistic load to understand its performance characteristics.

Demo



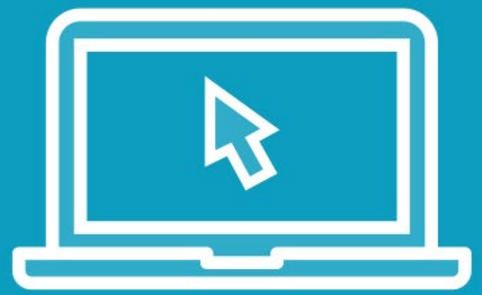
Learn to use multiline mode during regex matching

- Accessing multiple matches from a string

Requirements

- Process logs from a proxy server.
- Identify the IP addresses of clients that attempted to connect using HTTP rather than HTTPS.
- Collect each server hostname.
- The report should be organized by client IP address, listing unique hostnames they attempted insecure connections to.

Demo



Append and remove data with StringBuilders

Review

- Applied practical techniques while building the sample data processing application
- Learned about the string type
- Discussed character encoding
- Introduced regular expressions
- Working with strings
- Parsed and processed strings
- Applied regular expressions
- Compared and sorted strings
- Regex security considerations

Review

- Searched strings
- Modified strings
- Applied regex lookarounds
- Combined and formatted strings
- Applied StringBuilders for efficient string manipulation

Don’t expect to use all of the techniques in every application.

String Manipulation in C#: Best Practices

Steve Gordon

app.pluralsight.com/library/courses/string-manipulation-c-sharp-best-practices

Steve Gordon

MICROSOFT DEVELOPER TECHNOLOGIES MVP

@stevejgordon www.stevejgordon.co.uk