# Rational.ToString Method

名前空間: WS.Theia.ExtremelyPrecise

アセンブリ: ExtremelyPrecise.dll

現在の BigInteger オブジェクトの数値を等価の文字列形式に変換します。

# オーバーロード

|  |  |
| --- | --- |
| ToString() | 現在の Rational オブジェクトの数値を等価の文字列形式に変換します。 |
| ToString(IFormatProvider) | 指定されたカルチャ固有の書式情報を使用して、現在の Rational オブジェクトの数値をそれと等価の文字列形式に変換します。 |
| ToString(String) | 指定された書式を使用して、現在の Rational オブジェクトの数値をそれと等価な文字列形式に変換します。 |
| ToString(String,IFormatProvider) | 指定された書式とカルチャ固有の書式情報を使用して、現在の Rational オブジェクトの数値をそれと等価の文字列形式に変換します。 |

# ToString()

現在の Rational オブジェクトの数値を等価の文字列形式に変換します。

public override string ToString();

## 戻り値

String

現在の Rational 値の文字列形式。

## 例

次の例ではToString()メソッドを使ってRationalオブジェクトを文字列に変換しています。Rationalの標準書式指定子を使用した結果です。例ではEN-USカルチャの書式指定規則が適用されています。

// Initialize a Rational value.  
Rational value = Rational.Add(UInt64.MaxValue, 1024);  
  
// Display value using the default ToString method.  
Console.WriteLine(value.ToString());   
// Display value using some standard format specifiers.  
Console.WriteLine(value.ToString("G"));  
Console.WriteLine(value.ToString("C"));  
Console.WriteLine(value.ToString("D"));  
Console.WriteLine(value.ToString("F"));  
Console.WriteLine(value.ToString("N"));  
// The example displays the following output on a system whose current   
// culture is en-US:  
// 18446744073709552639  
// 18446744073709552639  
// $18,446,744,073,709,552,639.00  
// 18446744073709552639  
// 18446744073709552639.00  
// 18,446,744,073,709,552,639.00

## 注釈

ToString()メソッドはラウンドトリップ形式で既定のカルチャで変換を行います。特定のカルチャ、特定の形式で変換をしたい場合は、以下のオーバーロードを使用してください。

|  |  |  |
| --- | --- | --- |
| 形式 | カルチャ | オーバーロード |
| ラウンドトリップ形式 | 指定したカルチャ | ToString(IFormatProvider) |
| 指定した形式 | 既定のカルチャ | ToString(String) |
| 指定した形式 | 指定したカルチャ | ToString(String,IFormatProvider) |

# ToString(IFormatProvider)

指定されたカルチャ固有の書式情報を使用して、現在の Rational オブジェクトの数値をそれと等価の文字列形式に変換します。

public string ToString(IFormatProvider provider);

## パラメーター

provider　 IFormatProvider  
カルチャ固有の書式情報を提供するオブジェクト。

## 戻り値

String

現在の Rational 値の文字列形式を、provider パラメーターで指定されている形式で表現した値。

## 例

次の例では、NumberFormatInfoで負の符号としてチルダ（~）を設定したカスタムカルチャ用意しToString(IFormatProvider)メソッドを使ってRationalオブジェクトを文字列に変換しています。Rationalの標準書式指定子を使用した結果です。

Rational number = 9867857831128;  
number = Math.Pow(number, 3) \* Rational.MinusOne;  
  
NumberFormatInfo bigIntegerProvider = new NumberFormatInfo();  
bigIntegerProvider.NegativeSign = "~";  
  
Console.WriteLine(number.ToString(bigIntegerProvider));

## 注釈

ToString(IFormatProvider)メソッドはラウンドトリップ形式で既定のカルチャで変換を行います。特定のカルチャ、特定の形式で変換をしたい場合は、以下のオーバーロードを使用してください。

|  |  |  |
| --- | --- | --- |
| 形式 | カルチャ | オーバーロード |
| ラウンドトリップ形式 | 既定のカルチャ | ToString() |
| 指定した形式 | 既定のカルチャ | ToString(String) |
| 指定した形式 | 指定したカルチャ | ToString(String,IFormatProvider) |

providerパラメーターは、IFormatProviderインタフェースのGetFormatメソッドが実装されたオブジェクトを設定することができます。GetFormatメソッドの戻り値はNumberFormatInfoオブジェクトです。Parse(String,IFormatProvider)メソッドには主に3種類方法でproviderを渡すことができます。

* CultureInfoが提供する書式を表すオブジェクト。（GetFormatメソッドがそのカルチャに合わせたNumberFormatInfoオブジェクトを返します。）
* 直接インスタンス化したNumberFormatInfoオブジェクト。（GetFormatがNumberFormatInfoオブジェクト自信を返します）
* IFormatProviderを実装したカスタムオブジェクト。（そのオブジェクトのGetFormatメソッドがNumberFormatInfoオブジェクトをインスタンス化して返します。）

# ToString(String)

現在の Rational オブジェクトの数値を等価の文字列形式に変換します。

public string ToString(string format);

## 引数

Format　String

標準またはカスタムの数値書式指定文字列。

## 戻り値

String

現在の Rational 値の文字列形式を、format パラメーターで指定されている形式で表現した値。

## 例

次の例ではRationalを初期化し、いくつかのカスタム書式文字列を使用して変換しています。例ではEN-USカルチャの書式指定規則が適用されています。

Rational value = Rational.Parse("-903145792771643190182");  
string[] specifiers = { "C", "D", "D25", "E", "E4", "e8", "F0",   
 "G", "N0", "P", "R", "0,0.000",   
 "#,#.00#;(#,#.00#)" };  
  
foreach (string specifier in specifiers)  
 Console.WriteLine("{0}: {1}", specifier, value.ToString(specifier));  
  
// The example displays the following output:  
// C: ($903,145,792,771,643,190,182.00)  
// D: -903145792771643190182  
// D25: -0000903145792771643190182  
// E: -9.031457E+020  
// E4: -9.0314E+020  
// e8: -9.03145792e+020  
// F0: -903145792771643190182  
// G: -903145792771643190182  
// N0: -903,145,792,771,643,190,182  
// P: -90,314,579,277,164,319,018,200.00 %  
// R: -903145792771643190182  
// 0,0.000: -903,145,792,771,643,190,182.000  
// #,#.00#;(#,#.00#): (903,145,792,771,643,190,182.00)

## 注釈

ToString(String)メソッドはラウンドトリップ形式、既定のカルチャで変換を行います。特定のカルチャ、特定の形式で変換したい場合は以下のオーバーロードを使用してください。

|  |  |  |
| --- | --- | --- |
| 形式 | カルチャ | オーバーロード |
| ラウンドトリップ形式 | 既定のカルチャ | ToString() |
| ラウンドトリップ形式 | 指定したカルチャ | ToString(IFormatProvider) |
| 指定した形式 | 指定したカルチャ | ToString(String,IFormatProvider) |

# ToString(String,IFormatProvider)

現在の Rational オブジェクトの数値を等価の文字列形式に変換します。

public string ToString(String format,IFormatProvider provider);

## パラメーター

format　String  
標準またはカスタムの数値書式指定文字列。

provider 　IFormatProvider  
カルチャ固有の書式情報を提供するオブジェクト。

## 戻り値

String

format パラメーターと providerパラメーターで指定されている現在の Rational 値の文字列表現。

## 例

次の例ではRational値を初期化して標準書式指定文字列を使用して変換しています。NumberFormatInfoオブジェクトで負の符号としてチルダ（~）を定義しています。

// Redefine the negative sign as the tilde for the invariant culture.  
NumberFormatInfo bigIntegerFormatter = new NumberFormatInfo();  
bigIntegerFormatter.NegativeSign = "~";  
  
Rational value = Rational.Parse("-903145792771643190182");  
string[] specifiers = { "C", "D", "D25", "E", "E4", "e8", "F0",   
 "G", "N0", "P", "R", "0,0.000",   
 "#,#.00#;(#,#.00#)" };  
  
foreach (string specifier in specifiers)  
 Console.WriteLine("{0}: {1}", specifier, value.ToString(specifier,   
 bigIntegerFormatter));  
  
// The example displays the following output:  
// C: (☼903,145,792,771,643,190,182.00)  
// D: ~903145792771643190182  
// D25: ~0000903145792771643190182  
// E: ~9.031457E+020  
// E4: ~9.0314E+020  
// e8: ~9.03145792e+020  
// F0: ~903145792771643190182  
// G: ~903145792771643190182  
// N0: ~903,145,792,771,643,190,182  
// P: ~90,314,579,277,164,319,018,200.00 %  
// R: ~903145792771643190182  
// 0,0.000: ~903,145,792,771,643,190,182.000  
// #,#.00#;(#,#.00#): (903,145,792,771,643,190,182.00)

## 注釈

ToString(String,IFormatProvider)メソッドはラウンドトリップ形式で既定のカルチャで変換を行います。特定のカルチャ、特定の形式で変換をしたい場合は、以下のオーバーロードを使用してください。

|  |  |  |
| --- | --- | --- |
| 形式 | カルチャ | オーバーロード |
| ラウンドトリップ形式 | 既定のカルチャ | ToString() |
| ラウンドトリップ形式 | 指定したカルチャ | ToString(IFormatProvider) |
| 指定した形式 | 既定のカルチャ | ToString(String) |

providerパラメーターは、IFormatProviderインタフェースのGetFormatメソッドが実装されたオブジェクトを設定することができます。GetFormatメソッドの戻り値はNumberFormatInfoオブジェクトです。Parse(String,IFormatProvider)メソッドには主に3種類方法でproviderを渡すことができます。

* CultureInfoが提供する書式を表すオブジェクト。（GetFormatメソッドがそのカルチャに合わせたNumberFormatInfoオブジェクトを返します。）
* 直接インスタンス化したNumberFormatInfoオブジェクト。（GetFormatがNumberFormatInfoオブジェクト自信を返します）

IFormatProviderを実装したカスタムオブジェクト。（そのオブジェクトのGetFormatメソッドがNumberFormatInfoオブジェクトをインスタンス化して返します。）

# 適用対象

### .NET Core

2.0

### .NET Framework

4.6.1

### .NET Standard

2.0

### UWP

10.0.16299

### Xamarin.Android

8.0

### Xamarin.iOS

10.14

### Xamarin.Mac

3.8