# Rational.Equals Method

名前空間: WS.Theia.ExtremelyPrecise

アセンブリ: ExtremelyPrecise.dll

2つの値が等しいかどうかを示す値を返します。

# オーバーロード

|  |  |
| --- | --- |
| Equals(Decimal) | 現在のインスタンスの値と 10進数の値が等しいかどうかを示す値を返します。 |
| Equals(Double) | 現在のインスタンスの値と倍精度浮動小数点数の値が等しいかどうかを示す値を返します。 |
| Equals(Int64) | 現在のインスタンスの値と符号付き 64 ビット整数の値が等しいかどうかを示す値を返します。 |
| Equals(Object) | 現在のインスタンスの値と指定されたオブジェクトの値が等しいかどうかを示す値を返します。 |
| Equals(Rational) | 現在のインスタンスの値と Rational の値が等しいかどうかを示す値を返します。 |
| Equals(UInt64) | 現在のインスタンスの値と符号無し 64 ビット整数の値が等しいかどうかを示す値を返します。 |
| Equals(Rational,Rational) | 2 つの Rational オブジェクトの値が等しいかどうかを示す値を返します。 |

# Equals(Decimal)

現在のインスタンスの値と 10進数の値が等しいかどうかを示す値を返します。

public bool Equals(decimal other);

## パラメーター

other　Decimal  
比較する 10 進数。

## 戻り値

Boolean  
10進数の値と現在のインスタンスが等しい場合はtrue。それ以外の場合はfalse。

## 例

次の例ではRationalオブジェクトと10進数値をEquals(Decimal)メソッドで比較します。Rationalに、渡された数値を持つ10進数との比較である為、値が等しいと判定します。

Rational rationalValue;   
decimal decimalValue = 16.2m;  
rationalValue = new Rational(decimalValue);  
Console.WriteLine("{0} {1} = {2} {3} : {4}",   
 rationalValue.GetType().Name, rationalValue,  
 decimalValue.GetType().Name, decimalValue,   
 rationalValue.Equals(decimalValue));  
// The example displays the following output:  
// Rational 16.2 = Decimal 16.2 : True

## 注釈

等しいかどうかだけではなく、2つの値の相対的な大小を取得したい場合はRational.CompareTo(Decimal)メソッドを使用してください。

# Equals(Double)

現在のインスタンスの値と 10進数の値が等しいかどうかを示す値を返します。

public bool Equals(double other);

## パラメーター

other　Double  
比較する 10 進数。

## 戻り値

Boolean  
10進数の値と現在のインスタンスが等しい場合はtrue。それ以外の場合はfalse。

## 例

次の例ではRationalオブジェクトと倍精度浮動小数点をEquals(Double)メソッドで比較します。Rationalに、渡された数値を持つ倍精度浮動小数点との比較である為、値が等しいと判定します。

Rational rationalValue;   
  
float floatValue = 16.25f;  
rationalValue = new Rational(floatValue);  
Console.WriteLine("{0} {1} = {2} {3} : {4}",   
 rationalValue.GetType().Name, rationalValue,  
 floatValue.GetType().Name, floatValue,   
 rationalValue.Equals(floatValue));  
  
double doubleValue = 16.25d;  
rationalValue = new Rational(doubleValue);  
Console.WriteLine("{0} {1} = {2} {3} : {4}",   
 rationalValue.GetType().Name, rationalValue,  
 doubleValue.GetType().Name, doubleValue,   
 rationalValue.Equals(doubleValue));  
// The example displays the following output:  
// Rational 16.25 = Single 16.25 : True  
// Rational 16.25 = Double 16.25 : True

## 注釈

otherがSingleの場合は暗黙的にDoubleに変換して、このメソッドが呼び出されます。

等しいかどうかだけではなく、2つの値の相対的な大小を取得したい場合はRational.CompareTo(Decimal)メソッドを使用してください。

# Equals(Int64)

現在のインスタンスの値と符号付き 64 ビット整数の値が等しいかどうかを示す値を返します。

public bool Equals(long other);

## パラメーター

other　Int64  
比較する 符号付き 64 ビット整数。

## 戻り値

Boolean  
符号付き 64 ビット整数の値と現在のインスタンスが等しい場合はtrue。それ以外の場合はfalse。

## 例

次の例ではRationalオブジェクトと符号付き 64 ビット整数をEquals(Int64)メソッドで比較します。Rationalに、渡された数値を持つ符号付き 64 ビット整数との比較である為、値が等しいと判定します。

Rational rationalValue;   
  
byte byteValue = 16;  
rationalValue = new Rational(byteValue);  
Console.WriteLine("{0} {1} = {2} {3} : {4}",   
 rationalValue.GetType().Name, rationalValue,  
 byteValue.GetType().Name, byteValue,   
 rationalValue.Equals(byteValue));  
   
sbyte sbyteValue = -16;  
rationalValue = new Rational(sbyteValue);  
Console.WriteLine("{0} {1} = {2} {3} : {4}",   
 rationalValue.GetType().Name, rationalValue,  
 sbyteValue.GetType().Name, sbyteValue,  
 rationalValue.Equals(sbyteValue));  
  
short shortValue = 1233;  
rationalValue = new Rational(shortValue);  
Console.WriteLine("{0} {1} = {2} {3} : {4}",   
 rationalValue.GetType().Name, rationalValue,  
 shortValue.GetType().Name, shortValue,   
 rationalValue.Equals(shortValue));  
   
ushort ushortValue = 64000;  
rationalValue = new Rational(ushortValue);  
Console.WriteLine("{0} {1} = {2} {3} : {4}",   
 rationalValue.GetType().Name, rationalValue,  
 ushortValue.GetType().Name, ushortValue,   
 rationalValue.Equals(ushortValue));  
  
int intValue = -1603854;  
rationalValue = new Rational(intValue);  
Console.WriteLine("{0} {1} = {2} {3} : {4}",   
 rationalValue.GetType().Name, rationalValue,  
 intValue.GetType().Name, intValue,   
 rationalValue.Equals(intValue));  
  
uint uintValue = 1223300;  
rationalValue = new Rational(uintValue);  
Console.WriteLine("{0} {1} = {2} {3} : {4}",   
 rationalValue.GetType().Name, rationalValue,  
 uintValue.GetType().Name, uintValue,   
 rationalValue.Equals(uintValue));  
  
long longValue = -123822229012;  
rationalValue = new Rational(longValue);  
Console.WriteLine("{0} {1} = {2} {3} : {4}",   
 rationalValue.GetType().Name, rationalValue,  
 longValue.GetType().Name, longValue,   
 rationalValue.Equals(longValue));  
// The example displays the following output:  
// Rational 16 = Byte 16 : True  
// Rational -16 = SByte -16 : True  
// Rational 1233 = Int16 1233 : True  
// Rational 64000 = UInt16 64000 : True  
// Rational -1603854 = Int32 -1603854 : True  
// Rational 1223300 = UInt32 1223300 : True  
// Rational -123822229012 = Int64 -123822229012 : True

## 注釈

otherがByte、Int16、Int32、SByte、Uint16、又はUint32の場合は暗黙的にInt64に変換して、このメソッドが呼び出されます。

等しいかどうかだけではなく、2つの値の相対的な大小を取得したい場合はRational.CompareTo(Int64)メソッドを使用してください。

# Equals(Object)

現在のインスタンスの値と指定されたオブジェクトの値が等しいかどうかを示す値を返します。

public bool Equals(object obj);

## パラメーター

other　Object  
比較対象のオブジェクト。

## 戻り値

Boolean  
obj 引数が 数値 で、その値が現在の Rational インスタンスの値と等しい場合は true。それ以外の場合は false。

## 例

次の例ではObject配列とRational配列の各要素を比較しています。各要素は数値としての値は同じですが、Rationalオブジェクトの場合のみ等しいとみなされます。

using System;  
using System.Numerics;  
  
public class Example  
{  
 public static void Main()  
 {  
 object[] obj = { 0, 10, 100, new Rational(1000), -10 };  
 Rational [] rt = { Rational.Zero, new Rational (10),  
 new Rational (100), new Rational (1000),  
 new Rational (-10) };  
 for (int ctr = 0; ctr < rt.Length; ctr++)  
 Console.WriteLine(rt[ctr].Equals(obj[ctr]));  
 }  
}  
// The example displays the following output:  
// False  
// False  
// False  
// True  
// False

## 注釈

objパラメーターがRationalオブジェクトでない場合、Equals(Object)メソッドはfalseを返します。objがRationalオブジェクトかつ値が等しい場合にのみtrueを返します。

等しいかどうかだけではなく、2つの値の相対的な大小を取得したい場合はRational.CompareTo(Object)メソッドを使用してください。

# Equals(Rational)

現在のインスタンスの値と Rational の値が等しいかどうかを示す値を返します。

public bool Equals(WS.Theia.ExtremelyPrecise.Rational other);

## パラメーター

other　 WS.Theia.ExtremelyPrecise.Rational  
比較する Rational 値。

## 戻り値

Boolean  
Rational の値と現在のインスタンスの値が等しい場合は true。それ以外の場合は false。

## 例

次の例では地球からいくつかの星までの距離を比較しています。この場合は、Equalsの各オーバーロードを使用して等しいか比較しています。

const long LIGHT\_YEAR = 5878625373183;  
  
Rational altairDistance = 17 \* LIGHT\_YEAR;  
Rational epsilonIndiDistance = 12 \* LIGHT\_YEAR;  
Rational ursaeMajoris47Distance = 46 \* LIGHT\_YEAR;  
long tauCetiDistance = 12 \* LIGHT\_YEAR;  
ulong procyon2Distance = 12 \* LIGHT\_YEAR;  
object wolf424ABDistance = 14 \* LIGHT\_YEAR;  
  
Console.WriteLine("Approx. equal distances from Epsilon Indi to:");  
Console.WriteLine(" Altair: {0}",   
 epsilonIndiDistance.Equals(altairDistance));  
Console.WriteLine(" Ursae Majoris 47: {0}",   
 epsilonIndiDistance.Equals(ursaeMajoris47Distance));  
Console.WriteLine(" TauCeti: {0}",   
 epsilonIndiDistance.Equals(tauCetiDistance));  
Console.WriteLine(" Procyon 2: {0}",   
 epsilonIndiDistance.Equals(procyon2Distance));  
Console.WriteLine(" Wolf 424 AB: {0}",   
 epsilonIndiDistance.Equals(wolf424ABDistance));  
// The example displays the following output:  
// Approx. equal distances from Epsilon Indi to:  
// Altair: False  
// Ursae Majoris 47: False  
// TauCeti: True  
// Procyon 2: True  
// Wolf 424 AB: False

## 注釈

このメソッドはIEquatable<T>インターフェスの実装です。Equals(Object)と異なりotherパラメーターをキャストしない為、Equals(Object)より若干パフォーマンスが優れています。

等しいかどうかだけではなく、2つの値の相対的な大小を取得したい場合はRational.CompareTo(Rational)メソッドを使用してください。

# Equals(UInt64)

⚠重要  
この API は CLS 準拠ではありません。

現在のインスタンスの値と 符号なし64ビット整数の値が等しいかどうかを示す値を返します。

public bool Equals(ulong other);

## パラメーター

other　Decimal  
比較する符号なし64ビット整数。

## 戻り値

Boolean  
符号なし64ビット整数の値と現在のインスタンスが等しい場合はtrue。それ以外の場合はfalse。

## 例

次の例では地球からいくつかの星までの距離を比較しています。この場合は、Equalsの各オーバーロードを使用して等しいか比較しています。

const long LIGHT\_YEAR = 5878625373183;  
Rational altairDistance = 17 \* LIGHT\_YEAR;  
Rational epsilonIndiDistance = 12 \* LIGHT\_YEAR;  
Rational ursaeMajoris47Distance = 46 \* LIGHT\_YEAR;  
long tauCetiDistance = 12 \* LIGHT\_YEAR;  
ulong procyon2Distance = 12 \* LIGHT\_YEAR;  
object wolf424ABDistance = 14 \* LIGHT\_YEAR;  
  
Console.WriteLine("Approx. equal distances from Epsilon Indi to:");  
Console.WriteLine(" Altair: {0}",   
 epsilonIndiDistance.Equals(altairDistance));  
Console.WriteLine(" Ursae Majoris 47: {0}",   
 epsilonIndiDistance.Equals(ursaeMajoris47Distance));  
Console.WriteLine(" TauCeti: {0}",   
 epsilonIndiDistance.Equals(tauCetiDistance));  
Console.WriteLine(" Procyon 2: {0}",   
 epsilonIndiDistance.Equals(procyon2Distance));  
Console.WriteLine(" Wolf 424 AB: {0}",   
 epsilonIndiDistance.Equals(wolf424ABDistance));  
// The example displays the following output:  
// Approx. equal distances from Epsilon Indi to:  
// Altair: False  
// Ursae Majoris 47: False  
// TauCeti: True  
// Procyon 2: True  
// Wolf 424 AB: False

## 注釈

等しいかどうかだけではなく、2つの値の相対的な大小を取得したい場合はRational.CompareTo(UInt64)メソッドを使用してください。

# Equals(Rational,Rational)

名前空間: WS.Theia.ExtremelyPrecise

アセンブリ: ExtremelyPrecise.dll

2 つの Rational オブジェクトの値が等しいかどうかを示す値を返します。

public static WS.Theia.ExtremelyPrecise.Rational Equals(WS.Theia.ExtremelyPrecise.Rational left, WS.Theia.ExtremelyPrecise.Rational right);

## パラメーター

left　Rational  
比較する最初の値。

right　Rational  
比較する 2 番目の値。

## 戻り値

Rational  
left パラメーターと right パラメーターが同じ値の場合は true。それ以外の場合は false。

# 注釈

演算子のオーバーロードや、カスタム演算子をサポートしない言語用の、Rational値を比較する代替メソッド。Rational値を比較して変数に割り当てる時は次の例の様に使用する。

// The statement:  
// bool comp = Int64.MaxValue == Int32.MaxValue;  
// produces compiler error CS0220: The operation overflows at compile time in checked mode.  
// The alternative:  
bool comp = Rational.Equals(Int64.MaxValue, Int32.MaxValue);

# 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