Namespace NetAutoGUI

Classes

BitmapData
  Managed Bitmap, there is no need to dispose it explicitly.

BitmapDataExtensions

GUI
  Entry of controllers

KeyboardExtensions

Location
  A location

Rectangle
  A rectangle

RectangleWithConfidence
  Rectangle with confidence

ScreenshotExtensions

Size
  Size

Window
  A window on desktop

WindowExtensions

Interfaces

IApplicationController
  A controller for control processes and windows

IDialogController
  A controller for display dialogs

IKeyboardController
  Keyboard controller, used for simulating keyboard events

IMouseController
  Mouse controller, used for simulating mouse events

[IScreenshotController]
Controller for screenshot

[IServiceLoader]
Service loader for different OS.

[IWindowController]

Enums

[ImageType]
Image type

[MouseButtonType]
Mouse button type

[PauseMethod]

[VirtualKeyCode]

Class BitmapData

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

Managed Bitmap, there is no need to dispose it explicitly.

```
public record BitmapData : IEquatable<BitmapData>
```

Inheritance

object ← BitmapData

Implements

IEquatable<BitmapData>

Inherited Members

object.Equals(object), object.Equals(object, object), object.GetHashCode(), object.GetType(), object.MemberwiseClone(), object.ReferenceEquals(object, object), object.ToString()

Extension Methods

BitmapDataExtensions.ToMat(BitmapData, ImreadModes)

Constructors

BitmapData(byte[], int, int)

Managed Bitmap, there is no need to dispose it explicitly.

```
public BitmapData(byte[] Data, int Width, int Height)
```

Parameters

`Data` byte[]

bitmap format data of an image

`Width` int

Width of the image

`Height` int

Heigh of the image

Properties

Data

bitmap format data of an image

```
public byte[] Data { get; init; }
```

Property Value

byte[]

Height

Heigh of the image

```
public int Height { get; init; }
```

Property Value

int

LoadFromFileFunc

```
public static Func<string, BitmapData> LoadFromFileFunc { get; set; }
```

Property Value

Func<string, BitmapData>

Width

Width of the image

```
public int Width { get; init; }
```

Property Value

int

Methods

FromFile(string)

```
public static BitmapData FromFile(string imageFile)
```

Parameters

`imageFile` string

Returns

BitmapData

Save(Stream, ImageType)

Save the image into a stream

```
public void Save(Stream outStream, ImageType imgType)
```

Parameters

`outStream` Stream

the output stream

`imgType` ImageType

saved image format

Save(string, ImageType?)

Save the image into a local file.

```
public void Save(string filename, ImageType? imgType = null)
```

Parameters

`filename` string

file name

`imgType` ImageType?

saved image format

Class BitmapDataExtensions

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

```
public static class BitmapDataExtensions
```

Inheritance

object ← BitmapDataExtensions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

ToMat(BitmapData, ImreadModes)

Convert the BitmapData into a Mat of OpenCVSharp

```
public static Mat ToMat(this BitmapData bitmapData, ImreadModes flags = (ImreadModes)-1)
```

Parameters

`bitmapData` BitmapData

the bitmap data

`flags` ImreadModes

the flags

Returns

Mat

the converted Mat(It must be disposed after used)

Class GUI

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

Entry of controllers

```
public static class GUI
```

Inheritance

object ← GUI

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Fields

Application

```
public static readonly IApplicationController Application
```

Field Value

IApplicationController

Dialog

```
public static readonly IDialogController Dialog
```

Field Value

IDialogController

Keyboard

```
public static readonly IKeyboardController Keyboard
```

Field Value

IKeyboardController

Mouse

```
public static readonly IMouseController Mouse
```

Field Value

IMouseController

Screenshot

```
public static readonly IScreenshotController Screenshot
```

Field Value

IScreenshotController

Properties

PauseMethod

```
public static PauseMethod PauseMethod { get; set; }
```

Property Value

PauseMethod

Methods

Pause(double)

```
public static void Pause(double seconds)
```

Parameters

`seconds` double

WaitFor(Func<bool>, double)

```
public static void WaitFor(Func<bool> condition, double seconds = 1)
```

Parameters

`condition` Func<bool>

`seconds` double

Interface IApplicationController

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

A controller for control processes and windows

```
public interface IApplicationController
```

Methods

FindWindow(Func<Window, bool>)

Find a window using the given criteria

```
Window? FindWindow(Func<Window, bool> predict)
```

Parameters

`predict` Func<Window, bool>

the criteria

Returns

Window

the first found window

FindWindowById(long)

Find a window by its id/handler

```
Window? FindWindowById(long id)
```

Parameters

`id` long

Returns

Window

the found window

FindWindowByTitle(string)

Find the first window with the given title.

```
Window? FindWindowByTitle(string title)
```

Parameters

`title` string

the title of the window

Returns

Window

The first found window

FindWindowLikeTitle(string)

Find a window with a given title using wildcard

```
Window? FindWindowLikeTitle(string wildcard)
```

Parameters

`wildcard` string

the wildcard expression. it supports * and ?. For example: *notepad*, n?te

Returns

Window

the first found window

GetAllWindows()

Get all opened windows.

```
Window[] GetAllWindows()
```

Returns

Window[]

all opend windows

IsApplicationRunning(string, string?)

Check if there is any processes running with the given process name

```
bool IsApplicationRunning(string processName, string? arguments = null)
```

Parameters

`processName` string

the process's name.

`arguments` string

arguments

Returns

bool

true: the application is running; false: the application is not running

KillProcesses(string)

Kill all processes with the given name

```
void KillProcesses(string processName)
```

Parameters

`processName` string

the process's name.

LaunchApplication(string, string?)

Launch an application

```
Process LaunchApplication(string appPath, string? arguments = null)
```

Parameters

`appPath` string

the path to the application

`arguments` string

arguments passed to the application

Returns

Process

the Process object associated with the started application

OpenFileWithDefaultApp(string)

Open the given file with the default application

```
void OpenFileWithDefaultApp(string filePath)
```

Parameters

`filePath` string

the file path to be opened

WaitForApplication(string, double)

Wait for the first process with the give name running.

```
Process WaitForApplication(string processName, double timeoutSeconds = 5)
```

Parameters

`processName` string

the process's name.

`timeoutSeconds` double

timeout in second

Returns

Process

the first found process

Exceptions

TimeoutException

thrown when time is up

WaitForApplicationAsync(string, double, CancellationToken)

Wait for the first process with the give name running.

```
Task<Process> WaitForApplicationAsync(string processName, double timeoutSeconds = 5,
CancellationToken cancellationToken = default)
```

Parameters

`processName` string

the process's name.

`timeoutSeconds` double

timeout in second

`cancellationToken` CancellationToken

cancellationToken

Returns

Task<Process>

the first found process

Exceptions

TimeoutException

thrown when time is up

WaitForWindow(Func<Window, bool>, string, double)

Wait for a window using the given criteria

```
Window WaitForWindow(Func<Window, bool> predict, string errorMessageWhenTimeout, double
timeoutSeconds = 5)
```

Parameters

`predict` Func<Window, bool>

the condition

errorMessageWhenTimeout string

errorMessageWhenTimeout

timeoutSeconds double

timeout in second

Returns

Window

the first found window

Exceptions

TimeoutException

thrown when time is up

WaitForWindowAsync(Func<Window, bool>, string, double, CancellationToken)

Wait for a window using the given criteria

```
Task<Window> WaitForWindowAsync(Func<Window, bool> predict, string
errorMessageWhenTimeout, double timeoutSeconds = 5, CancellationToken cancellationToken
= default)
```

Parameters

predict Func<Window, bool>

the condition

errorMessageWhenTimeout string

errorMessageWhenTimeout

timeoutSeconds double

timeout in second

`cancellationToken` CancellationToken

cancellationToken

Returns

Task<Window>

the first found window

Exceptions

TimeoutException

thrown when time is up

WaitForWindowByTitle(string, double)

Wait for the window with the given title

```
Window WaitForWindowByTitle(string title, double timeoutSeconds = 5)
```

Parameters

`title` string

title

`timeoutSeconds` double

timeout in second

Returns

Window

The first found window

Exceptions

TimeoutException

thrown when time is up

WaitForWindowByTitleAsync(string, double, CancellationToken)

Wait for the window with the given title

```
Task<Window> WaitForWindowByTitleAsync(string title, double timeoutSeconds = 5,
CancellationToken cancellationToken = default)
```

Parameters

`title` string

title

`timeoutSeconds` double

timeout in second

`cancellationToken` CancellationToken

cancellationToken

Returns

Task<Window>

The first found window

Exceptions

TimeoutException

thrown when time is up

WaitForWindowLikeTitle(string, double)

Wait for a window using the given wildcard title

```
Window WaitForWindowLikeTitle(string wildcard, double timeoutSeconds = 5)
```

Parameters

`wildcard` string

the wildcard expression. it supports * and ?. For example: *notepad*, n?te

`timeoutSeconds` double

timeout in second

Returns

Window

the first found window

Exceptions

TimeoutException

thrown when time is up

WaitForWindowLikeTitleAsync(string, double, CancellationToken)

Wait for a window using the given wildcard title

```
Task<Window> WaitForWindowLikeTitleAsync(string wildcard, double timeoutSeconds = 5,
CancellationToken cancellationToken = default)
```

Parameters

`wildcard` string

the wildcard expression. it supports * and ?. For example: *notepad*, n?te

`timeoutSeconds` double

timeout in second

`cancellationToken` CancellationToken

cancellationToken

Returns

Task<Window>

the first found window

Exceptions

TimeoutException

thrown when time is up

Interface IDialogController

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

A controller for display dialogs

```
public interface IDialogController
```

Properties

Parent

The parent window handler for dialogs of the controller. If no parent is specified, the current process's active window will be used as parent.

```
long? Parent { get; set; }
```

Property Value

long?

Methods

Alert(string, string)

Pop up an alert dialog.

```
void Alert(string text, string title = "Alert")
```

Parameters

`text` string

text

`title` string

title

Confirm(string, string)

Popup a confirmation dialog

```
bool Confirm(string text, string title = "Confirm")
```

Parameters

`text` string

text

`title` string

title

Returns

bool

true: [Ok] button is pressed; false: [Cancel] button is pressed.

Password(string, string?, string?)

Popup an entry dialog for password

```
string? Password(string title = "", string? okText = null, string? cancelText = null)
```

Parameters

`title` string

title

`okText` string

text of [OK] button, defaulted to be [OK]

`cancelText` string

text of [Cancel] button, defaulted to be [Cancel]

Returns

string

The password entered

Prompt(string, string?, string?)

Popup an entry dialog

```
string? Prompt(string title = "", string? okText = null, string? cancelText = null)
```

Parameters

`title` string

title

`okText` string

text of [OK] button, defaulted to be [OK]

`cancelText` string

text of [Cancel] button, defaulted to be [Cancel]

Returns

string

The text entered

SelectFileForLoad(string)

Pop up a loading file dialog

```
string? SelectFileForLoad(string filters = "")
```

Parameters

`filters` string

The file filters. Example: "txt files (.*txt)*|.txt|All files (.)|."

Returns

string

the selected file path

SelectFileForSave(string)

Pop up a saving file dialog.

```
string? SelectFileForSave(string filters = "")
```

Parameters

`filters` string

The file filters. Example: "txt files (.*txt)*|.txt|All files (.)|."

Returns

string

the selected file path

SelectFolder()

Pop up a folder selection dialog.

```
string? SelectFolder()
```

Returns

string

the selected path

YesNoBox(string, string)

Popup a Yes/No dialog

```
bool YesNoBox(string text, string title = "Ask")
```

Parameters

`text` string

text

`title` string

title

Returns

bool

true: [Yes] button is pressed; false: [No] button is pressed.

Interface IKeyboardController

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

Keyboard controller, used for simulating keyboard events

```
public interface IKeyboardController
```

Extension Methods

KeyboardExtensions.Ctrl_A(IKeyboardController) ,
KeyboardExtensions.Ctrl_C(IKeyboardController) ,
KeyboardExtensions.Ctrl_V(IKeyboardController)

Methods

Hold(VirtualKeyCode)

Press down a key until the return the Dispose() method of the returned IDisposable is invoked.

```
IDisposable Hold(VirtualKeyCode key)
```

Parameters

`key` VirtualKeyCode

Returns

IDisposable

HotKey(params VirtualKeyCode[])

pressed down keys in order, and then released in reverse order

```
void HotKey(params VirtualKeyCode[] keys)
```

Parameters

`keys` VirtualKeyCode[]

KeyDown(VirtualKeyCode)

Press down a key

```
void KeyDown(VirtualKeyCode key)
```

Parameters

`key` VirtualKeyCode

key

KeyUp(VirtualKeyCode)

Press up a key

```
void KeyUp(VirtualKeyCode key)
```

Parameters

`key` VirtualKeyCode

key

Press(params VirtualKeyCode[])

Press a keys combination

```
void Press(params VirtualKeyCode[] keys)
```

Parameters

`keys` VirtualKeyCode[]

keys

Write(char)

Write a character from keyboard

```
void Write(char c)
```

Parameters

c char

the character

Write(string)

Write a string from keyboard

```
void Write(string s)
```

Parameters

s string

the string

Write(string, double)

Write a string from keyboard, wait a specific interval between each character

```
void Write(string s, double intervalInSeconds)
```

Parameters

s string

the string

`intervalInSeconds` double

interval of wait in seconds

Interface IMouseController

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

Mouse controller, used for simulating mouse events

```
public interface IMouseController
```

Methods

Click(int?, int?, MouseButtonType, int, double)

Simulate a single mouse click.

```
void Click(int? x = null, int? y = null, MouseButtonType button = MouseButtonType.Left,
int clicks = 1, double intervalInSeconds = 0)
```

Parameters

`x` int?

mouse x. The default value is current mouse x.

`y` int?

mouse y. The default value is current mouse y.

`button` MouseButtonType

which mouse button to click

`clicks` int

click count

`intervalInSeconds` double

interval in seconds between clicks

DoubleClick(int?, int?, MouseButtonType, double)

Simulate a double mouse click.

```
void DoubleClick(int? x = null, int? y = null, MouseButtonType button =
MouseButtonType.Left, double intervalInSeconds = 0)
```

Parameters

`x` int?

move mouse to (x,y), then click the button

`y` int?

move mouse to (x,y), then click the button

`button` MouseButtonType

which mouse button to click

`intervalInSeconds` double

interval in seconds

MouseDown(int?, int?, MouseButtonType)

Simulate a mouse down

```
void MouseDown(int? x = null, int? y = null, MouseButtonType button
= MouseButtonType.Left)
```

Parameters

`x` int?

x

`y` int?

y

`button` MouseButtonType

which button

MouseUp(int?, int?, MouseButtonType)

Simulate a mouse up

```
void MouseUp(int? x = null, int? y = null, MouseButtonType button = MouseButtonType.Left)
```

Parameters

`x` int?

x

`y` int?

y

`button` MouseButtonType

which button

Move(int, int)

move the mouse cursor over a few pixels relative to its current position

```
void Move(int offsetX, int offsetY)
```

Parameters

`offsetX` int

`offsetY` int

MoveTo(int, int)

Move the mouse cursor to the specific location

```
void MoveTo(int x, int y)
```

Parameters

x int

y int

Position()

Get current location of the mouse cursor

```
Location Position()
```

Returns

Location

Scroll(int)

Scroll the mouse wheel

```
void Scroll(int value)
```

Parameters

value int

positive value is for scrolling up, negative is value for scrolling down

Interface IScreenshotController

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

Controller for screenshot

```
public interface IScreenshotController
```

Extension Methods
ScreenshotExtensions.ClickOnScreen(IScreenshotController, string, double, double) ,
ScreenshotExtensions.Highlight(IScreenshotController, BitmapData, double) ,
ScreenshotExtensions.LocateAllOnScreen(IScreenshotController, BitmapData, double) ,
ScreenshotExtensions.LocateOnScreen(IScreenshotController, BitmapData, double) ,
ScreenshotExtensions.WaitOnScreen(IScreenshotController, BitmapData, double, double) ,
ScreenshotExtensions.WaitOnScreen(IScreenshotController, string, double, double) ,
ScreenshotExtensions.WaitOnScreenAsync(IScreenshotController, BitmapData, double, double, CancellationToken)

Methods

Highlight(params Rectangle[])

Highlight several areas

```
void Highlight(params Rectangle[] rectangles)
```

Parameters

`rectangles` Rectangle[]

multiple areas to highlight

LocateAll(BitmapData, BitmapData, double)

Locates all occurrences of a given bitmap within a base image with a specified confidence level.

```
Rectangle[] LocateAll(BitmapData basePicture, BitmapData bitmapToBeFound, double
confidence = 0.99)
```

Parameters

`basePicture` BitmapData

The base image where the search is performed

`bitmapToBeFound` BitmapData

The image to locate within the base image

`confidence` double

The confidence level required for a match, ranging from 0.0 to 1.0. A value closer to 1.0 ensures higher accuracy but may result in fewer matches.

Returns

Rectangle[]

An array of Rectangle objects, each representing a located instance of `bitmapToBeFound` within `basePicture`.

LocateAllWithConfidence(BitmapData, BitmapData, double)

Locates all occurrences of a given bitmap within a base image with a specified confidence level.

```
RectangleWithConfidence[] LocateAllWithConfidence(BitmapData basePicture, BitmapData
bitmapToBeFound, double confidence = 0.99)
```

Parameters

`basePicture` BitmapData

The base image where the search is performed

`bitmapToBeFound` BitmapData

The image to locate within the base image `confidence` double

The confidence level required for a match, ranging from 0.0 to 1.0. A value closer to 1.0 ensures higher accuracy but may result in fewer matches.

Returns

RectangleWithConfidence[]

An array of RectangleWithConfidence objects, each representing a located instance of `bitmapToBeFound` within `basePicture`, along with the confidence score.

Screenshot()

Take a screenshot. If there are multiple monitors, they will be displayed into a single image with system's multiple displays' arrangement. On Windows, please invoke GUIWindows.Initialize() at the beginning of application's entry, for example Main() or Program.cs

```
BitmapData Screenshot()
```

Returns

BitmapData

Screenshot(Window)

Take a screenshot of a window.

```
BitmapData Screenshot(Window window)
```

Parameters

`window` Window

Returns

BitmapData

Interface IServiceLoader

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

Service loader for different OS.

```
public interface IServiceLoader
```

Methods

LoadApplicationController()

Load ApplicationController

```
IApplicationController LoadApplicationController()
```

Returns

IApplicationController

LoadDialogController()

Load DialogController

```
IDialogController LoadDialogController()
```

Returns

IDialogController

LoadKeyboardController()

Load KeyboardController

```
IKeyboardController LoadKeyboardController()
```

Returns

IKeyboardController

LoadMouseController()

Load MouseController

```
IMouseController LoadMouseController()
```

Returns

IMouseController

LoadScreenshotController()

Load ScreenshotController

```
IScreenshotController LoadScreenshotController()
```

Returns

IScreenshotController

LoadWindowController()

```
IWindowController LoadWindowController()
```

Returns

IWindowController

Interface IWindowController

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

```
public interface IWindowController
```

Methods

Close(Window)

```
void Close(Window window)
```

Parameters

`window` Window

GetBoundary(Window)

```
Rectangle GetBoundary(Window window)
```

Parameters

`window` Window

Returns

Rectangle

GetTitle(Window)

```
string GetTitle(Window window)
```

Parameters

`window` Window

Returns

string

PressKey(Window, VirtualKeyCode)

```
void PressKey(Window window, VirtualKeyCode keyCode)
```

Parameters

`window` Window

`keyCode` VirtualKeyCode

Enum ImageType

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

Image type

```
public enum ImageType
```

Fields

`Jpg = 1`

`Png = 2`

`WebP = 0`

Class KeyboardExtensions

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

```
public static class KeyboardExtensions
```

Inheritance

object ← KeyboardExtensions

Inherited Members

object.Equals(object), object.Equals(object, object), object.GetHashCode(), object.GetType(), object.MemberwiseClone(), object.ReferenceEquals(object, object), object.ToString()

Methods

Ctrl_A(IKeyboardController)

Press Ctrl+A

```
public static void Ctrl_A(this IKeyboardController kb)
```

Parameters

`kb` IKeyboardController

Ctrl_C(IKeyboardController)

Press Ctrl+C.

```
public static void Ctrl_C(this IKeyboardController kb)
```

Parameters

`kb` IKeyboardController

Ctrl_V(IKeyboardController)

Press Ctrl+V

```
public static void Ctrl_V(this IKeyboardController kb)
```

Parameters

`kb` IKeyboardController

Class Location

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

A location

```
public record Location : IEquatable<Location>
```

Inheritance

object ← Location

Implements

IEquatable<Location>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Location(int, int)

A location

```
public Location(int X, int Y)
```

Parameters

X int

x

Y int

y

Properties

X

x

```
public int X { get; init; }
```

Property Value

int

Y

y

```
public int Y { get; init; }
```

Property Value

int

Methods

Deconstruct(out int, out int)

```
public void Deconstruct(out int x, out int y)
```

Parameters

x int

y int

Operators

implicit operator Vector2(Location)

```
public static implicit operator Vector2(Location loc)
```

Parameters

`loc` Location

Returns

Vector2

implicit operator Location(Vector2)

```
public static implicit operator Location(Vector2 vec2)
```

Parameters

`vec2` Vector2

Returns

Location

Enum MouseButtonType

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

Mouse button type

```
public enum MouseButtonType
```

Fields

`Left = 0`

`Middle = 1`

`Right = 2`

Enum PauseMethod

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

```
public enum PauseMethod
```

Fields

`Sleep = 0`

Use Thread.Sleep(), which is CPU-friendly; however, it may cause dead-lock when being used in multiple-thread context, and async methods.

`SpinWait = 1`

Use SpinWait, which causes high CPU usage; however, it's fool-proof when being used in multiple-thread context, and async methods. It's the default value. Warning: Avoid using it for waiting too long.

Class Rectangle

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

A rectangle

```
public record Rectangle : IEquatable<Rectangle>
```

Inheritance

object ← Rectangle

Implements

IEquatable<Rectangle>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

Rectangle(int, int, int, int)

A rectangle

```
public Rectangle(int X, int Y, int Width, int Height)
```

Parameters

`X` int

`Y` int

`Width` int

`Height` int

Properties

Area

Area of the rectangle

```
public int Area { get; }
```

Property Value

int

Center

Center point of the Rectangle

```
public Location Center { get; }
```

Property Value

Location

Height

```
public int Height { get; init; }
```

Property Value

int

Width

```
public int Width { get; init; }
```

Property Value

int

X

```
public int X { get; init; }
```

Property Value

int

Y

```
public int Y { get; init; }
```

Property Value

int

Methods

Contains(Location)

If the give location `loc` is within the rectangle.

```
public bool Contains(Location loc)
```

Parameters

`loc` Location

location

Returns

bool

If it's within or not.

Deconstruct(out int, out int, out int, out int)

```
public void Deconstruct(out int x, out int y, out int width, out int height)
```

Parameters

`x` int

`y` int

`width` int

`height` int

ToString()

Returns a string that represents the current object.

```
public override string ToString()
```

Returns

string

A string that represents the current object.

Class RectangleWithConfidence

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

Rectangle with confidence

```
public record RectangleWithConfidence : IEquatable<RectangleWithConfidence>
```

Inheritance

object ← RectangleWithConfidence

Implements

IEquatable<RectangleWithConfidence>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RectangleWithConfidence(Rectangle, double)

Rectangle with confidence

```
public RectangleWithConfidence(Rectangle Rectangle, double Confidence)
```

Parameters

`Rectangle` Rectangle

Rectangle

`Confidence` double

Confidence

Properties

Confidence

Confidence

```
public double Confidence { get; init; }
```

Property Value

double

Rectangle

Rectangle

```
public Rectangle Rectangle { get; init; }
```

Property Value

Rectangle

Methods

Deconstruct(out Rectangle, out double)

```
public void Deconstruct(out Rectangle rectangle, out double confidence)
```

Parameters

`rectangle` Rectangle

`confidence` double

Operators

implicit operator Rectangle(RectangleWithConfidence)

```
public static implicit operator Rectangle(RectangleWithConfidence rwc)
```

Parameters

`rwc` RectangleWithConfidence

Returns

Rectangle

Class ScreenshotExtensions

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

```
public static class ScreenshotExtensions
```

Inheritance

object ← ScreenshotExtensions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

ClickOnScreen(IScreenshotController, string, double, double)

```
public static void ClickOnScreen(this IScreenshotController ctl, string imgFileToBeFound,
double confidence = 0.99, double timeoutSeconds = 5)
```

Parameters

`ctl` IScreenshotController

`imgFileToBeFound` string

`confidence` double

`timeoutSeconds` double

Highlight(IScreenshotController, BitmapData, double)

```
public static void Highlight(this IScreenshotController ctl, BitmapData imgFileToBeFound,
double confidence = 0.99)
```

Parameters

`ctl` IScreenshotController

`imgFileToBeFound` BitmapData

`confidence` double

LocateAllOnScreen(IScreenshotController, BitmapData, double)

```
public static Rectangle[] LocateAllOnScreen(this IScreenshotController ctrl, BitmapData
imgFileToBeFound, double confidence = 0.99)
```

Parameters

`ctrl` IScreenshotController

`imgFileToBeFound` BitmapData

`confidence` double

Returns

Rectangle[]

LocateOnScreen(IScreenshotController, BitmapData, double)

```
public static Rectangle? LocateOnScreen(this IScreenshotController ctl, BitmapData
imgFileToBeFound, double confidence = 0.99)
```

Parameters

`ctl` IScreenshotController

`imgFileToBeFound` BitmapData

`confidence` double

Returns

Rectangle

WaitOnScreen(IScreenshotController, BitmapData, double, double)

```
public static Rectangle WaitOnScreen(this IScreenshotController ctl, BitmapData
imgFileToBeFound, double confidence = 0.99, double timeoutSeconds = 5)
```

Parameters

`ctl` IScreenshotController

`imgFileToBeFound` BitmapData

`confidence` double

`timeoutSeconds` double

Returns

Rectangle

WaitOnScreen(IScreenshotController, string, double, double)

```
public static Rectangle WaitOnScreen(this IScreenshotController ctl, string
imgFileToBeFound, double confidence = 0.99, double timeoutSeconds = 5)
```

Parameters

`ctl` IScreenshotController

`imgFileToBeFound` string

`confidence` double

`timeoutSeconds` double

Returns

Rectangle

WaitOnScreenAsync(IScreenshotController, BitmapData, double, double, CancellationToken)

```
public static Task<Rectangle> WaitOnScreenAsync(this IScreenshotController ctl,
BitmapData imgFileToBeFound, double confidence = 0.99, double timeoutSeconds = 5,
CancellationToken cancellationToken = default)
```

Parameters

`ctl` IScreenshotController

`imgFileToBeFound` BitmapData

`confidence` double

`timeoutSeconds` double

`cancellationToken` CancellationToken

Returns

Task<Rectangle>

Class Size

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

Size

```
public record Size : IEquatable<Size>
```

Inheritance

object ← Size

Implements

IEquatable<Size>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Size(int, int)

Size

```
public Size(int Width, int Height)
```

Parameters

`Width` int

Width

`Height` int

Height

Properties

Height

Height

```
public int Height { get; init; }
```

Property Value

int

Width

Width

```
public int Width { get; init; }
```

Property Value

int

Methods

Deconstruct(out int, out int)

```
public void Deconstruct(out int width, out int height)
```

Parameters

`width` int

`height` int

Enum VirtualKeyCode

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

```
public enum VirtualKeyCode
```

Fields

`ACCEPT = 30`

IME accept

`ADD = 107`

Add key

`APPS = 93`

Applications key (Natural keyboard)

`ATTN = 246`

Attn key

`BACK = 8`

BACKSPACE key

`BROWSER_BACK = 166`

Windows 2000/XP: Browser Back key

`BROWSER_FAVORITES = 171`

Windows 2000/XP: Browser Favorites key

`BROWSER_FORWARD = 167`

Windows 2000/XP: Browser Forward key

`BROWSER_HOME = 172`

Windows 2000/XP: Browser Start and Home key

`BROWSER_REFRESH = 168`

Windows 2000/XP: Browser Refresh key

`BROWSER_SEARCH = 170`

Windows 2000/XP: Browser Search key

`BROWSER_STOP = 169`

Windows 2000/XP: Browser Stop key

`CANCEL = 3`

Control-break processing

`CAPITAL = 20`

CAPS LOCK key

`CLEAR = 12`

CLEAR key

`CONTROL = 17`

CTRL key

`CONVERT = 28`

IME convert

`CRSEL = 247`

CrSel key

`DECIMAL = 110`

Decimal key

`DELETE = 46`

DEL key

`DIVIDE = 111`

Divide key

`DOWN = 40`

DOWN ARROW key

`END = 35`

END key

`EREOF = 249`

Erase EOF key

`ESCAPE = 27`

ESC key

`EXECUTE = 43`

EXECUTE key

`EXSEL = 248`

ExSel key

`F1 = 112`

F1 key

`F10 = 121`

F10 key

`F11 = 122`

F11 key

`F12 = 123`

F12 key

`F13 = 124`

F13 key

`F14 = 125`

F14 key

`F15 = 126`

F15 key

`F16 = 127`

F16 key

`F17 = 128`

F17 key

`F18 = 129`

F18 key

`F19 = 130`

F19 key

`F2 = 113`

F2 key

`F20 = 131`

F20 key

`F21 = 132`

F21 key

`F22 = 133`

F22 key

`F23 = 134`

F23 key

`F24 = 135`

F24 key

`F3 = 114`

F3 key

`F4 = 115`

F4 key

`F5 = 116`

F5 key

`F6 = 117`

F6 key

`F7 = 118`

F7 key

`F8 = 119`

F8 key

`F9 = 120`

F9 key

`FINAL = 24`

IME final mode

`HANGEUL = 21`

IME Hanguel mode (maintained for compatibility; use HANGUL)

`HANGUL = 21`

IME Hangul mode

`HANJA = 25`

IME Hanja mode

`HELP = 47`

HELP key

`HOME = 36`

HOME key

`INSERT = 45`

INS key

`JUNJA = 23`

IME Junja mode

`KANA = 21`

Input Method Editor (IME) Kana mode

`KANJI = 25`

IME Kanji mode

`LAUNCH_APP1 = 182`

Windows 2000/XP: Start Application 1 key

`LAUNCH_APP2 = 183`

Windows 2000/XP: Start Application 2 key

`LAUNCH_MAIL = 180`

Windows 2000/XP: Start Mail key

`LAUNCH_MEDIA_SELECT = 181`

Windows 2000/XP: Select Media key

`LBUTTON = 1`

Left mouse button

`LCONTROL = 162`

Left CONTROL key - Used only as parameters to GetAsyncKeyState() and GetKeyState()

`LEFT = 37`

LEFT ARROW key

`LMENU = 164`

Left MENU key - Used only as parameters to GetAsyncKeyState() and GetKeyState()

`LSHIFT = 160`

Left SHIFT key - Used only as parameters to GetAsyncKeyState() and GetKeyState()

`LWIN = 91`

Left Windows key (Microsoft Natural keyboard)

`MBUTTON = 4`

Middle mouse button (three-button mouse) - NOT contiguous with LBUTTON and RBUTTON

`MEDIA_NEXT_TRACK = 176`

Windows 2000/XP: Next Track key

`MEDIA_PLAY_PAUSE = 179`

Windows 2000/XP: Play/Pause Media key

`MEDIA_PREV_TRACK = 177`

Windows 2000/XP: Previous Track key

`MEDIA_STOP = 178`

Windows 2000/XP: Stop Media key

`MENU = 18`

ALT key

`MODECHANGE = 31`

IME mode change request

`MULTIPLY = 106`

Multiply key

`NEXT = 34`

PAGE DOWN key

`NONAME = 252`

Reserved

`NONCONVERT = 29`

IME nonconvert

`NUMLOCK = 144`

NUM LOCK key

`NUMPAD0 = 96`

Numeric keypad 0 key

`NUMPAD1 = 97`

Numeric keypad 1 key

`NUMPAD2 = 98`

Numeric keypad 2 key

`NUMPAD3 = 99`

Numeric keypad 3 key

`NUMPAD4 = 100`

Numeric keypad 4 key

`NUMPAD5 = 101`

Numeric keypad 5 key

`NUMPAD6 = 102`

Numeric keypad 6 key

`NUMPAD7 = 103`

Numeric keypad 7 key

`NUMPAD8 = 104`

Numeric keypad 8 key

`NUMPAD9 = 105`

Numeric keypad 9 key

`NUMPAD_RETURN = 1073741837`

Numeric keypad ENTER key

`OEM_1 = 186`

Used for miscellaneous characters; it can vary by keyboard. Windows 2000/XP: For the US standard keyboard, the ';:' key

`OEM_102 = 226`

Windows 2000/XP: Either the angle bracket key or the backslash key on the RT 102-key keyboard

`OEM_2 = 191`

Used for miscellaneous characters; it can vary by keyboard. Windows 2000/XP: For the US standard keyboard, the '/?' key

`OEM_3 = 192`

Used for miscellaneous characters; it can vary by keyboard. Windows 2000/XP: For the US standard keyboard, the '`~' key

`OEM_4 = 219`

Used for miscellaneous characters; it can vary by keyboard. Windows 2000/XP: For the US standard keyboard, the '[{' key

`OEM_5 = 220`

Used for miscellaneous characters; it can vary by keyboard. Windows 2000/XP: For the US standard keyboard, the '|' key

`OEM_6 = 221`

Used for miscellaneous characters; it can vary by keyboard. Windows 2000/XP: For the US standard keyboard, the ']}' key

`OEM_7 = 222`

Used for miscellaneous characters; it can vary by keyboard. Windows 2000/XP: For the US standard keyboard, the 'single-quote/double-quote' key

`OEM_8 = 223`

Used for miscellaneous characters; it can vary by keyboard.

`OEM_CLEAR = 254`

Clear key

`OEM_COMMA = 188`

Windows 2000/XP: For any country/region, the ',' key

`OEM_MINUS = 189`

Windows 2000/XP: For any country/region, the '-' key

`OEM_PERIOD = 190`

Windows 2000/XP: For any country/region, the '.' key

`OEM_PLUS = 187`

Windows 2000/XP: For any country/region, the '+' key

`PA1 = 253`

PA1 key

`PACKET = 231`

Windows 2000/XP: Used to pass Unicode characters as if they were keystrokes. The PACKET key is the low word of a 32-bit Virtual Key value used for non-keyboard input methods. For more information, see Remark in KEYBDINPUT, SendInput, WM_KEYDOWN, and WM_KEYUP

`PAUSE = 19`

PAUSE key

`PLAY = 250`

Play key

`PRINT = 42`

PRINT key

`PRIOR = 33`

PAGE UP key

`PROCESSKEY = 229`

Windows 95/98/Me, Windows NT 4.0, Windows 2000/XP: IME PROCESS key

`RBUTTON = 2`

Right mouse button

`RCONTROL = 163`

Right CONTROL key - Used only as parameters to GetAsyncKeyState() and GetKeyState()

`RETURN = 13`

ENTER key

`RIGHT = 39`

RIGHT ARROW key

`RMENU = 165`

Right MENU key - Used only as parameters to GetAsyncKeyState() and GetKeyState()

`RSHIFT = 161`

Right SHIFT key - Used only as parameters to GetAsyncKeyState() and GetKeyState()

`RWIN = 92`

Right Windows key (Natural keyboard)

`SCROLL = 145`

SCROLL LOCK key

`SELECT = 41`

SELECT key

`SEPARATOR = 108`

Separator key

`SHIFT = 16`

SHIFT key

`SLEEP = 95`

Computer Sleep key

`SNAPSHOT = 44`

PRINT SCREEN key

`SPACE = 32`

SPACEBAR

`SUBTRACT = 109`

Subtract key

`TAB = 9`

TAB key

`UP = 38`

UP ARROW key

`VK_0 = 48`

0 key

`VK_1 = 49`

1 key

`VK_2 = 50`

2 key

`VK_3 = 51`

3 key

`VK_4 = 52`

4 key

`VK_5 = 53`

5 key

`VK_6 = 54`

6 key

`VK_7 = 55`

7 key

`VK_8 = 56`

8 key

`VK_9 = 57`

9 key

`VK_A = 65`

A key

`VK_B = 66`

B key

`VK_C = 67`

C key

`VK_D = 68`

D key

`VK_E = 69`

E key

`VK_F = 70`

F key

`VK_G = 71`

G key

`VK_H = 72`

H key

`VK_I = 73`

I key

`VK_J = 74`

J key

`VK_K = 75`

K key

`VK_L = 76`

L key

`VK_M = 77`

M key

`VK_N = 78`

N key

`VK_O = 79`

O key

`VK_P = 80`

P key

`VK_Q = 81`

Q key

`VK_R = 82`

R key

`VK_S = 83`

S key

`VK_T = 84`

T key

`VK_U = 85`

U key

`VK_V = 86`

V key

`VK_W = 87`

W key

`VK_X = 88`

X key

`VK_Y = 89`

Y key

`VK_Z = 90`

Z key

`VOLUME_DOWN = 174`

Windows 2000/XP: Volume Down key

`VOLUME_MUTE = 173`

Windows 2000/XP: Volume Mute key

`VOLUME_UP = 175`

Windows 2000/XP: Volume Up key

`XBUTTON1 = 5`

Windows 2000/XP: X1 mouse button - NOT contiguous with LBUTTON and RBUTTON

`XBUTTON2 = 6`

Windows 2000/XP: X2 mouse button - NOT contiguous with LBUTTON and RBUTTON

`ZOOM = 251`

Zoom key

Class Window

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

A window on desktop

```
public class Window
```

Inheritance

object ← Window

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Extension Methods

WindowExtensions.Click(Window, int?, int?, MouseButtonType, int, double) ,
WindowExtensions.DoubleClick(Window, int?, int?, MouseButtonType, double) ,
WindowExtensions.Highlight(Window, params Rectangle[]) ,
WindowExtensions.LocateAll(Window, BitmapData, double) ,
WindowExtensions.MouseDown(Window, int?, int?, MouseButtonType) ,
WindowExtensions.MouseUp(Window, int?, int?, MouseButtonType) ,
WindowExtensions.MoveMouseTo(Window, int, int) ,
WindowExtensions.Wait(Window, BitmapData, double, double) ,
WindowExtensions.WaitAndClick(Window, BitmapData, double, double) ,
WindowExtensions.WaitAndClickAsync(Window, BitmapData, double, double, CancellationToken)
,
WindowExtensions.WaitAsync(Window, BitmapData, double, double, CancellationToken)

Constructors

Window(long)

```
public Window(long id)
```

Parameters

`id` long

Properties

Boundary

```
public Rectangle Boundary { get; }
```

Property Value

Rectangle

Id

```
public long Id { get; }
```

Property Value

long

Title

```
public string Title { get; }
```

Property Value

string

Methods

Close()

```
public void Close()
```

PressKey(VirtualKeyCode)

```
public void PressKey(VirtualKeyCode keyCode)
```

Parameters

`keyCode` VirtualKeyCode

Class WindowExtensions

Namespace: NetAutoGUI

Assembly: NetAutoGUI.dll

```
public static class WindowExtensions
```

Inheritance

object ← WindowExtensions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Click(Window, int?, int?, MouseButtonType, int, double)

Simulate a single mouse click at the given position relative to the given window`window`.

```
public static void Click(this Window window, int? winX = null, int? winY = null,
MouseButtonType button = MouseButtonType.Left, int clicks = 1, double intervalInSeconds
= 0)
```

Parameters

`window` Window

window

`winX` int?

mouse x to window origin. The default value is current mouse x.

`winY` int?

mouse y to window origin. The default value is current mouse y.

`button` MouseButtonType

which mouse button

`clicks` int

click times

`intervalInSeconds` double

interval in seconds between clicks

DoubleClick(Window, int?, int?, MouseButtonType, double)

Simulate a double mouse click at the given position relative to the given window`window`.

```
public static void DoubleClick(this Window window, int? winX = null, int? winY = null,
MouseButtonType button = MouseButtonType.Left, double intervalInSeconds = 0)
```

Parameters

`window` Window

window

`winX` int?

mouse x to window origin. The default value is current mouse x.

`winY` int?

mouse y to window origin. The default value is current mouse y.

`button` MouseButtonType

which mouse button

`intervalInSeconds` double

interval in seconds between clicks

Highlight(Window, params Rectangle[])

Highlight several areas

```
public static void Highlight(this Window window, params Rectangle[] relativeRects)
```

Parameters

`window` Window

window

`relativeRects` Rectangle[]

multiple areas to highlight

LocateAll(Window, BitmapData, double)

Locates all occurrences of a given bitmap within the window with a specified confidence level.

```
public static Rectangle[] LocateAll(this Window window, BitmapData imgFileToBeFound,
double confidence = 0.99)
```

Parameters

`window` Window

The window where the search is performed

`imgFileToBeFound` BitmapData

The image to locate within the window

`confidence` double

The confidence level required for a match, ranging from 0.0 to 1.0. A value closer to 1.0 ensures higher accuracy but may result in fewer matches.

Returns

Rectangle[]

An array of Rectangle objects, each representing a located instance of `imgFileToBeFound` within `window`.

MouseDown(Window, int?, int?, MouseButtonType)

Press down a mouse key down on a window

```
public static void MouseDown(this Window window, int? winX = null, int? winY = null,
MouseButtonType button = MouseButtonType.Left)
```

Parameters

`window` Window

window

`winX` int?

x to the window. Default value is the current mouse position.

`winY` int?

y to the window. Default value is the current mouse position.

`button` MouseButtonType

which button

MouseUp(Window, int?, int?, MouseButtonType)

Release a mouse key down on a window

```
public static void MouseUp(this Window window, int? winX = null, int? winY = null,
MouseButtonType button = MouseButtonType.Left)
```

Parameters

`window` Window

window

`winX` int?

x to the window. Default value is the current mouse position.

`winY` int?

y to the window. Default value is the current mouse position.

`button` MouseButtonType

which button

MoveMouseTo(Window, int, int)

Move the mouse cursor to the specific location

```
public static void MoveMouseTo(this Window window, int winX, int winY)
```

Parameters

`window` Window

window

`winX` int

`winY` int

Wait(Window, BitmapData, double, double)

Wait for the first matched area(matched with`imgFileToBeFound`)

```
public static Rectangle Wait(this Window window, BitmapData imgFileToBeFound, double
confidence = 0.99, double timeoutSeconds = 5)
```

Parameters

`window` Window

window

`imgFileToBeFound` BitmapData

The image to locate within the window

`confidence` double

The confidence level required for a match, ranging from 0.0 to 1.0. A value closer to 1.0 ensures higher accuracy but may result in fewer matches.

`timeoutSeconds` double

timeout in seconds

Returns

Rectangle

Rectangle of the first found area relative to `window`

Exceptions

TimeoutException

not found after timeout

WaitAndClick(Window, BitmapData, double, double)

Wait for the first matched area(matched with`imgFileToBeFound`) and click the centre of the area

```
public static void WaitAndClick(this Window window, BitmapData imgFileToBeFound, double
confidence = 0.99, double timeoutSeconds = 5)
```

Parameters

`window` Window

window

`imgFileToBeFound` BitmapData

The image to locate within the window

`confidence` double

The confidence level required for a match, ranging from 0.0 to 1.0. A value closer to 1.0 ensures higher accuracy but may result in fewer matches.

`timeoutSeconds` double

timeout in seconds

Exceptions

TimeoutException

not found after timeout

WaitAndClickAsync(Window, BitmapData, double, double, CancellationToken)

Wait for the first matched area(matched with`imgFileToBeFound`) and click the centre of the area

```
public static Task WaitAndClickAsync(this Window window, BitmapData imgFileToBeFound,
double confidence = 0.99, double timeoutSeconds = 5, CancellationToken cancellationToken
= default)
```

Parameters

`window` Window

window

`imgFileToBeFound` BitmapData

The image to locate within the window

`confidence` double

The confidence level required for a match, ranging from 0.0 to 1.0. A value closer to 1.0 ensures higher accuracy but may result in fewer matches.

`timeoutSeconds` double

timeout in seconds

`cancellationToken` CancellationToken

cancellationToken

Returns

Task

Exceptions

TimeoutException

not found after timeout

WaitAsync(Window, BitmapData, double, double, CancellationToken)

Wait for the first matched area(matched with`imgFileToBeFound`)

```
public static Task<Rectangle> WaitAsync(this Window window, BitmapData imgFileToBeFound,
double confidence = 0.99, double timeoutSeconds = 5, CancellationToken cancellationToken
= default)
```

Parameters

`window` Window

window

`imgFileToBeFound` BitmapData

The image to locate within the window

`confidence` double

The confidence level required for a match, ranging from 0.0 to 1.0. A value closer to 1.0 ensures higher accuracy but may result in fewer matches.

`timeoutSeconds` double

timeout in seconds

`cancellationToken` CancellationToken

cancellationToken

Returns

Task < Rectangle >

Rectangle of the first found area

Exceptions

TimeoutException

not found after timeout

Namespace NetAutoGUI.Internals

Classes

AbstractMouseController

AbstractScreenshotController

Constants

KeyHoldContext

TimeBoundWaiter

ValidationHelpers

Class AbstractMouseController

Namespace: NetAutoGUI.Internals

Assembly: NetAutoGUI.dll

```
public abstract class AbstractMouseController : IMouseController
```

Inheritance

object ← AbstractMouseController

Implements

IMouseController

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Click(int?, int?, MouseButtonType, int, double)

Simulate a single mouse click.

```
public abstract void Click(int? x = null, int? y = null, MouseButtonType button =
MouseButtonType.Left, int clicks = 1, double intervalInSeconds = 0)
```

Parameters

`x` int?

mouse x. The default value is current mouse x.

`y` int?

mouse y. The default value is current mouse y.

`button` MouseButtonType

which mouse button to click

`clicks` int?

click count

`intervalInSeconds` double

interval in seconds between clicks

DoubleClick(int?, int?, MouseButtonType, double)

Simulate a double mouse click.

```
public void DoubleClick(int? x = null, int? y = null, MouseButtonType button =
MouseButtonType.Left, double intervalInSeconds = 0)
```

Parameters

`x` int?

move mouse to (x,y), then click the button

`y` int?

move mouse to (x,y), then click the button

`button` MouseButtonType

which mouse button to click

`intervalInSeconds` double

interval in seconds

MouseDown(int?, int?, MouseButtonType)

Simulate a mouse down

```
public abstract void MouseDown(int? x = null, int? y = null, MouseButtonType button
= MouseButtonType.Left)
```

Parameters

`x` int?

x

`y` int?

y

`button` MouseButtonType

which button

MouseUp(int?, int?, MouseButtonType)

Simulate a mouse up

```
public abstract void MouseUp(int? x = null, int? y = null, MouseButtonType button
= MouseButtonType.Left)
```

Parameters

`x` int?

x

`y` int?

y

`button` MouseButtonType

which button

Move(int, int)

move the mouse cursor over a few pixels relative to its current position

```
public void Move(int offsetX, int offsetY)
```

Parameters

`offsetX` int

`offsetY` int

MoveTo(int, int)

Move the mouse cursor to the specific location

```
public abstract void MoveTo(int x, int y)
```

Parameters

`x` int

`y` int

Position()

Get current location of the mouse cursor

```
public abstract Location Position()
```

Returns

Location

Scroll(int)

Scroll the mouse wheel

```
public abstract void Scroll(int value)
```

Parameters

`value` int

positive value is for scrolling up, negative is value for scrolling down

Class AbstractScreenshotController

Namespace: NetAutoGUI.Internals

Assembly: NetAutoGUI.dll

```
public abstract class AbstractScreenshotController : IScreenshotController
```

Inheritance

object ← AbstractScreenshotController

Implements

IScreenshotController

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Extension Methods

ScreenshotExtensions.ClickOnScreen(IScreenshotController, string, double, double) , ScreenshotExtensions.Highlight(IScreenshotController, BitmapData, double) , ScreenshotExtensions.LocateAllOnScreen(IScreenshotController, BitmapData, double) , ScreenshotExtensions.LocateOnScreen(IScreenshotController, BitmapData, double) , ScreenshotExtensions.WaitOnScreen(IScreenshotController, BitmapData, double, double) , ScreenshotExtensions.WaitOnScreen(IScreenshotController, string, double, double) , ScreenshotExtensions.WaitOnScreenAsync(IScreenshotController, BitmapData, double, double, CancellationToken)

Methods

Highlight(params Rectangle[])

Highlight several areas

```
public abstract void Highlight(params Rectangle[] rectangles)
```

Parameters

rectangles Rectangle[]

multiple areas to highlight

LocateAllWithConfidence(BitmapData, BitmapData, double)

Locates all occurrences of a given bitmap within a base image with a specified confidence level.

```
public RectangleWithConfidence[] LocateAllWithConfidence(BitmapData basePicture,
BitmapData bitmapToBeFound, double confidence = 0.99)
```

Parameters

basePicture BitmapData

The base image where the search is performed

bitmapToBeFound BitmapData

The image to locate within the base image

confidence double

The confidence level required for a match, ranging from 0.0 to 1.0. A value closer to 1.0 ensures higher accuracy but may result in fewer matches.

Returns

RectangleWithConfidence[]

An array of RectangleWithConfidence objects, each representing a located instance of `bitmapToBeFound` within `basePicture`, along with the confidence score.

Screenshot()

Take a screenshot. If there are multiple monitors, they will be displayed into a single image with system's multiple displays' arrangement. On Windows, please invoke GUIWindows.Initialize() at the beginning of application's entry, for example Main() or Program.cs

```
public abstract BitmapData Screenshot()
```

Returns

BitmapData

Screenshot(Window)

Take a screenshot of a window.

```
public abstract BitmapData Screenshot(Window window)
```

Parameters

`window` Window

Returns

BitmapData

ScreenshotLocationToRelativeLocation(int, int)

Convert the location of the screenshot to the relative location to the primary screen.

```
public abstract (int x, int y) ScreenshotLocationToRelativeLocation(int x, int y)
```

Parameters

`x` int

`y` int

Returns

(int x, int y)

Class Constants

Namespace: NetAutoGUI.Internals

Assembly: NetAutoGUI.dll

```
public static class Constants
```

Inheritance

object ← Constants

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Fields

DefaultWaitSeconds

```
public const int DefaultWaitSeconds = 5
```

Field Value

int

Class KeyHoldContext

Namespace: NetAutoGUI.Internals

Assembly: NetAutoGUI.dll

```
public class KeyHoldContext : IDisposable
```

Inheritance

object ← KeyHoldContext

Implements

IDisposable

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

KeyHoldContext(VirtualKeyCode, IKeyboardController)

```
public KeyHoldContext(VirtualKeyCode holdedKey, IKeyboardController keyboardController)
```

Parameters

`holdedKey` VirtualKeyCode

`keyboardController` IKeyboardController

Methods

Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public void Dispose()
```

Class TimeBoundWaiter

Namespace: NetAutoGUI.Internals

Assembly: NetAutoGUI.dll

```
public static class TimeBoundWaiter
```

Inheritance

object ← TimeBoundWaiter

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

WaitForNotNullAsync<T>(Func<T?>, double, string, CancellationToken)

```
public static Task<T> WaitForNotNullAsync<T>(Func<T?> func, double timeoutSeconds, string
errorMessageWhenTimeout, CancellationToken cancellationToken)
```

Parameters

`func` Func<T>

`timeoutSeconds` double

`errorMessageWhenTimeout` string

`cancellationToken` CancellationToken

Returns

Task<T>

Type Parameters

T

WaitForNotNull<T>(Func<T?>, double, string)

```
public static T WaitForNotNull<T>(Func<T?> func, double timeoutSeconds,
string errorMessageWhenTimeout)
```

Parameters

func Func<T>

timeoutSeconds double

errorMessageWhenTimeout string

Returns

T

Type Parameters

T

Class ValidationHelpers

Namespace: NetAutoGUI.Internals

Assembly: NetAutoGUI.dll

```
public static class ValidationHelpers
```

Inheritance

object ← ValidationHelpers

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

CheckReturn(bool, string)

```
public static void CheckReturn(this bool retValue, string funcName)
```

Parameters

`retValue` bool

`funcName` string

NotNegative(double, string)

```
public static void NotNegative(this double value, string argName)
```

Parameters

`value` double

`argName` string

NotNegative(int, string)

```
public static void NotNegative(this int value, string argName)
```

Parameters

`value` int

`argName` string

Namespace System.Runtime.Compiler Services

Classes

IsExternalInit

Class IsExternalInit

Namespace: System.Runtime.CompilerServices

Assembly: NetAutoGUI.dll

```
public class IsExternalInit
```

Inheritance

object ← IsExternalInit

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Namespace WildcardMatch

Classes

StringExtensions

Extensions to string

Class StringExtensions

Namespace: WildcardMatch

Assembly: NetAutoGUI.dll

Extensions to string

```
public static class StringExtensions
```

Inheritance

object ← StringExtensions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

WildcardMatch(string, string, bool)

Tells if the given string matches the given wildcard. Two wildcards are allowed: '' *and '?' '* ' matches 0 or more characters '?' matches any character

```
public static bool WildcardMatch(this string wildcard, string s, bool ignoreCase = false)
```

Parameters

`wildcard` string

The wildcard.

`s` string

The s.

`ignoreCase` bool

if set to `true` [ignore case].

Returns

bool