Namespace AetherUtils.Core.Attributes

Classes

ConfigAttribute

Specifies the name of a property in a YAML configuration file.

Class ConfigAttribute

Namespace: AetherUtils.Core.Attributes

Assembly: AetherUtils.Core.dll

Specifies the name of a property in a YAML configuration file.

```
[AttributeUsage(AttributeTargets.Property)]
public sealed class ConfigAttribute : Attribute
```

Inheritance

object ← Attribute ← ConfigAttribute

Inherited Members

Attribute.Equals(object) , Attribute.GetCustomAttribute(Assembly, Type) ,
Attribute.GetCustomAttribute(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(Module, Type) , Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) , Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) , Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) , Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) , Attribute.GetCustomAttributes(Module) ,
Attribute.GetCustomAttributes(Module, bool) , Attribute.GetCustomAttributes(Module, Type) ,
Attribute.GetCustomAttributes(Module, Type, bool) , Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) , Attribute.GetHashCode() ,
Attribute.IsDefaultAttribute() , Attribute.IsDefined(Assembly, Type) ,
Attribute.IsDefined(Assembly, Type, bool) , Attribute.IsDefined(MemberInfo, Type) ,
Attribute.IsDefined(MemberInfo, Type, bool) , Attribute.IsDefined(Module, Type) ,
Attribute.IsDefined(Module, Type, bool) , Attribute.IsDefined(ParameterInfo, Type) ,
Attribute.IsDefined(ParameterInfo, Type, bool) , Attribute.Match(object) , Attribute.TypeId ,
object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) ,
object.ToString()

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Constructors

ConfigAttribute(string)

Specifies the name of a property in a YAML configuration file.

```
public ConfigAttribute(string name)
```

Parameters

`name` string

Properties

Name

The YAML name of the property in the configuration file.

```
public string Name { get; }
```

Property Value

string

Namespace AetherUtils.Core.Configuration

Classes

ConfigManager<T>

Provides methods for saving, loading, and querying a generic configuration. This class cannot be instantiated. A child class must be created inheriting from this class.

DefaultConfig

A DTO representing the default, bare configuration of a new application. This class can be used as is, or a new class can be created to store the configuration for an application.
If a new class is needed, it must be a DTO and it's properties should be marked with ConfigAttribute attributes in order to be saved and loaded from disk by a ConfigManager<T>.

This class can contain instances of other DTO classes so long as those classes also have the ConfigAttribute on their properties.

LogOptions

A DTO representing options for how log files are handled and formatted for an application.

YamlConfigManager

Represents a configuration manager for a YAML configuration file using DefaultConfig as the base configuration.

Interfaces

IConfig

Interface that all ConfigManager<T> classes must implement.

Class ConfigManager<T>

Namespace: AetherUtils.Core.Configuration

Assembly: AetherUtils.Core.dll

Provides methods for saving, loading, and querying a generic configuration. This class cannot be instantiated. A child class must be created inheriting from this class.

```
public abstract class ConfigManager<T> : IConfig where T : class
```

Type Parameters

`T`

The DTO class that represents the configuration.

Inheritance

object ← ConfigManager<T>

Implements

IConfig

Derived

YamlConfigManager

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Remarks

The custom class should have its properties related to configuration marked with a ConfigAttribute. These properties are the only ones which will be serialized and de-serialized from disk.

Constructors

ConfigManager(string)

Provides methods for saving, loading, and querying a generic configuration. This class cannot be instantiated. A child class must be created inheriting from this class.

```
protected ConfigManager(string configFilePath)
```

Parameters

`configFilePath` string

Remarks

The custom class should have its properties related to configuration marked with a ConfigAttribute. These properties are the only ones which will be serialized and de-serialized from disk.

Properties

ConfigExists

Get a value indicating whether the configuration file exists on disk.

```
public bool ConfigExists { get; }
```

Property Value

bool

ConfigFilePath

Get or set the path to the configuration file on disk.

```
public string? ConfigFilePath { get; set; }
```

Property Value

string

CurrentConfig

Get or set the current configuration.

```
protected T? CurrentConfig { get; set; }
```

Property Value

T

IsInitialized

Get a value indicating whether the configuration has been initialized.

```
public bool IsInitialized { get; }
```

Property Value

bool

Methods

CreateDefaultConfig()

Create the default configuration.

```
public abstract bool CreateDefaultConfig()
```

Returns

bool

Get(ConfigOption)

Get a configuration value specified by the configuration `option`.

```
public object? Get(ConfigOption option)
```

Parameters

`option` ConfigOption

The ConfigOption containing information about the value to get.

Returns

object

The configuration value or `null` if the value did not exist.

Get(string)

Get a configuration value specified by `configName`.

```
public object? Get(string configName)
```

Parameters

`configName` string

The name of the configuration to get.

Returns

object

The configuration value or `null` if the value did not exist.

Load()

Load a configuration file from disk based on the ConfigFilePath.

```
public bool Load()
```

Returns

bool

`true` if the config loaded successfully; `false` otherwise.

Exceptions

ArgumentException

If ConfigFilePath is `null` or empty.

FileNotFoundException

LoadAsync()

Asynchronously load a configuration file from disk based on the ConfigFilePath.

```
public Task<bool> LoadAsync()
```

Returns

Task<bool>

`true` if the config loaded successfully; `false` otherwise.

Exceptions

ArgumentException

If ConfigFilePath is `null` or empty.

FileNotFoundException

If the configuration file specified by ConfigFilePath was not found on disk.

Save()

Save a configuration file to disk based on the CurrentConfig and the ConfigFilePath.

```
public bool Save()
```

Returns

bool

`true` if the config saved successfully; `false` otherwise.

Exceptions

ArgumentException

If ConfigFilePath is `null` or empty.

SaveAsync()

Asynchronously save a configuration file to disk based on the CurrentConfig and the ConfigFilePath.

```
public Task<bool> SaveAsync()
```

Returns

Task<bool>

`true` if the config saved successfully; `false` otherwise.

Exceptions

ArgumentException

If ConfigFilePath is `null` or empty.

Set(ConfigOption)

Set a configuration value specified by the configuration `option`.

```
public bool Set(ConfigOption option)
```

Parameters

`option` ConfigOption

The ConfigOption containing information about the value to set.

Returns

bool

`true` if the value was set successfully; `false` otherwise.

Set(string, object?)

Set a configuration `value` specified by `configName`.

```
public bool Set(string configName, object? value)
```

Parameters

`configName` string

The name of the configuration to set.

`value` object

The value to set.

Returns

bool

`true` if the value was set successfully; `false` otherwise.

Class DefaultConfig

Namespace: AetherUtils.Core.Configuration

Assembly: AetherUtils.Core.dll

A DTO representing the default, bare configuration of a new application. This class can be used as is, or a new class can be created to store the configuration for an application.
If a new class is needed, it must be a DTO and it's properties should be marked with ConfigAttribute attributes in order to be saved and loaded from disk by a ConfigManager<T>.

This class can contain instances of other DTO classes so long as those classes also have the ConfigAttribute on their properties.

```
public sealed class DefaultConfig
```

Inheritance

object ← DefaultConfig

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Properties

ConnectionString

The connection string used for connecting to a database.

```
[Config("connectionString")]
public string ConnectionString { get; set; }
```

Property Value

string

LicenseFile

The full path to a valid license file for an application.

```
[Config("licenseFile")]
public string LicenseFile { get; set; }
```

Property Value

string

LogOptions

A collection of options used for logging.

```
[Config("logOptions")]
public LogOptions LogOptions { get; set; }
```

Property Value

LogOptions

Interface IConfig

Namespace: AetherUtils.Core.Configuration

Assembly: AetherUtils.Core.dll

Interface that all ConfigManager<T> classes must implement.

```
public interface IConfig
```

Extension Methods
TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Properties

ConfigExists

Get a value indicating whether the configuration file specified by ConfigFilePath exists.

```
bool ConfigExists { get; }
```

Property Value

bool

ConfigFilePath

The file path to a configuration file. This path can contain Windows path variables (i.e., `%TEMP%`). They will be expanded when saving and loading.

```
string? ConfigFilePath { get; set; }
```

Property Value

string

IsInitialized

Get a value indicating whether this configuration is initialized and ready to be used.

```
bool IsInitialized { get; }
```

Property Value

bool

Methods

Get(ConfigOption)

Get the configuration value for the named config property.

```
object? Get(ConfigOption option)
```

Parameters

`option` ConfigOption

The ConfigOption defining the configuration parameters to get.

Returns

object

The value of the configuration property.

Load()

Deserialize a configuration file from disk, if it exists.

```
bool Load()
```

Returns

bool

`true` if the file loaded successfully; `false`, otherwise.

LoadAsync()

Asynchronously deserialize a configuration file from disk, if it exists.

```
Task<bool> LoadAsync()
```

Returns

Task <bool>

`true` if the file loaded successfully; `false`, otherwise.

Save()

Serialize and save a configuration file to disk based on the current configuration.

```
bool Save()
```

Returns

bool

`true` if the file saved successfully; `false`, otherwise.

SaveAsync()

Asynchronously serialize and save a configuration file to disk based on the current configuration.

```
Task<bool> SaveAsync()
```

Returns

Task <bool>

`true` if the file saved successfully; `false`, otherwise.

Set(ConfigOption)

Set a configuration value for the named config property.

```
bool Set(ConfigOption option)
```

Parameters

`option` ConfigOption

The ConfigOption defining the configuration parameters to set.

Returns

bool

`true` if the value was set successfully; `false` otherwise.

Class LogOptions

Namespace: AetherUtils.Core.Configuration

Assembly: AetherUtils.Core.dll

A DTO representing options for how log files are handled and formatted for an application.

```
public sealed class LogOptions
```

Inheritance

object ← LogOptions

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Properties

AppName

The name of the application doing the logging.

```
[Config("appName")]
public string AppName { get; set; }
```

Property Value

string

IncludeDateOnly

Indicates if the log file name should include the date only, instead of the full formatted DateTime.

```
[Config("includeDateOnly")]
public bool IncludeDateOnly { get; set; }
```

Property Value

bool

IncludeDateTime

Indicates if the log file name should include the current formatted DateTime the file was created.

```
[Config("includeDateTime")]
public bool IncludeDateTime { get; set; }
```

Property Value

bool

LogFileDirectory

The directory that the log file should be saved to.

```
[Config("logFileDirectory")]
public string LogFileDirectory { get; set; }
```

Property Value

string

LogFooter

Specifies the footer to add at the end of each log file.

```
[Config("logFooter")]
public string LogFooter { get; set; }
```

Property Value

string

LogHeader

Specifies the header to add at the top of each log file.

```
[Config("logHeader")]
public string LogHeader { get; set; }
```

Property Value

string

LogLayout

Specifies the default log layout to use for NLog.

```
public string LogLayout { get; set; }
```

Property Value

string

NewFileEveryLaunch

Indicates whether a new log file should be created for every new launch of the application.

```
[Config("newFileEveryLaunch")]
public bool NewFileEveryLaunch { get; set; }
```

Property Value

bool

WriteLogToConsole

Indicates whether the log should write to the system console in addition to a log file.

```
[Config("writeLogToConsole")]
public bool WriteLogToConsole { get; set; }
```

Property Value

bool

Class YamlConfigManager

Namespace: AetherUtils.Core.Configuration

Assembly: AetherUtils.Core.dll

Represents a configuration manager for a YAML configuration file using DefaultConfig as the base configuration.

```
public sealed class YamlConfigManager : ConfigManager<DefaultConfig>, IConfig
```

Inheritance

object ← ConfigManager<DefaultConfig> ← YamlConfigManager

Implements

IConfig

Inherited Members

ConfigManager<DefaultConfig>.ConfigFilePath , ConfigManager<DefaultConfig>.IsInitialized , ConfigManager<DefaultConfig>.ConfigExists , ConfigManager<DefaultConfig>.LoadAsync() , ConfigManager<DefaultConfig>.Load() , ConfigManager<DefaultConfig>.SaveAsync() , ConfigManager<DefaultConfig>.Save() , ConfigManager<DefaultConfig>.Get(ConfigOption) , ConfigManager<DefaultConfig>.Get(string) , ConfigManager<DefaultConfig>.Set(ConfigOption) , ConfigManager<DefaultConfig>.Set(string, object) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Constructors

YamlConfigManager(string)

Represents a configuration manager for a YAML configuration file using DefaultConfig as the base configuration.

```
public YamlConfigManager(string configFilePath)
```

Parameters

`configFilePath` string

The path to the configuration file.

Methods

CreateDefaultConfig()

Create a new, default configuration in memory.

```
public override bool CreateDefaultConfig()
```

Returns

bool

Namespace AetherUtils.Core.Exceptions

Classes

PasswordRuleException

Custom exception class for password rule validation.

QrException

Custom class for exceptions that occur when handling QR codes.

Class PasswordRuleException

Namespace: AetherUtils.Core.Exceptions

Assembly: AetherUtils.Core.dll

Custom exception class for password rule validation.

```
public class PasswordRuleException : Exception, ISerializable
```

Inheritance

object ← Exception ← PasswordRuleException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Constructors

PasswordRuleException(string)

Custom exception class for password rule validation.

```
public PasswordRuleException(string message)
```

Parameters

`message` string

Class QrException

Namespace: AetherUtils.Core.Exceptions

Assembly: AetherUtils.Core.dll

Custom class for exceptions that occur when handling QR codes.

```
public class QrException : Exception, ISerializable
```

Inheritance

object ← Exception ← QrException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Constructors

QrException(string)

```
public QrException(string message)
```

Parameters

`message` string

QrException(string, Exception)

```
public QrException(string message, Exception innerException)
```

Parameters

`message` string

`innerException` Exception

Namespace AetherUtils.Core.Extensions

Classes

ArrayExtensions

Provides extension methods for manipulating Array objects.

CollectionExtensions

Provides extension methods for manipulating IDictionary<TKey, TValue> and ICollection<T> objects.

ControlExtensions

Provides extension methods for manipulating WinForm Control objects.

DataTableExtensions

Provides extension methods for manipulating DataTable objects.

EnumExtensions

Provides extension methods for manipulating Enum objects.

ImageExtensions

Provides extension methods for manipulating Image objects.

NumberExtensions

Provides extension methods for manipulating various number objects.

SizeExtensions

Provides extension methods for manipulating Size objects.

StreamExtensions

Provides extension methods for manipulating Stream objects.

StringExtensions

Provides extension methods for manipulating string objects.

TimeExtensions

Provides extension methods for manipulating various time related objects.

TypeExtensions

Provides extension methods for manipulating generic object types.

Class ArrayExtensions

Namespace: AetherUtils.Core.Extensions

Assembly: AetherUtils.Core.dll

Provides extension methods for manipulating Array objects.

```
public static class ArrayExtensions
```

Inheritance

object ← ArrayExtensions

Inherited Members

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

Methods

ToImage(byte[])

Get an Image represented by the byte array.

```
public static Image ToImage(this byte[] bytes)
```

Parameters

`bytes` byte[]

The byte array containing properly formatted image bytes.

Returns

Image

A new Image.

Exceptions

ArgumentNullException

If `bytes` was `null`.

FormatException

If `bytes` did not have a valid ImageFormat.

ToPrintableString(IEnumerable<byte>)

Convert the numeric value of each element of a specified array of bytes to its equivalent hexadecimal string representation.

```
public static string ToPrintableString(this IEnumerable<byte> bytes)
```

Parameters

`bytes` IEnumerable<byte>

An IEnumerable<T> of bytes.

Returns

string

A string of hexadecimal pairs separated by hyphens, where each pair represents the corresponding element in value; for example, "7F-2C-4A-00".

Exceptions

ArgumentNullException

If `bytes` was `null`.

ToStringList<T>(T[], char, string, string)

Convert an array of values to a string using the specified prefix, delimiter, and optional suffix.

```
public static string ToStringList<T>(this T[] values, char delimiter = ',', string prefix =
"", string suffix = "")
```

Parameters

`values` T[]

The array of values to combine.

`delimiter` char

The delimiter to separate the values with; default is `','`

`prefix` string

The prefix to add to the resulting string; default is an empty string.

`suffix` string

The suffix to add to the resulting string; default is an empty string.

Returns

string

A string formatted with the `values` seperated by the `delimiter`.

Type Parameters

`T`

The type of objects contained within the array.

Examples

```
string[] values = ["val1", "val2", "val3"];
string str = values.ToStringList(',', "PRE(", ")");
returns: "PRE(val1,val2,val3)"
```

Exceptions

ArgumentNullException

If `values`, `prefix`, or `delimiter` was `null`.

Class CollectionExtensions

Namespace: AetherUtils.Core.Extensions

Assembly: AetherUtils.Core.dll

Provides extension methods for manipulating IDictionary<TKey, TValue> and ICollection<T> objects.

```
public static class CollectionExtensions
```

Inheritance

object ← CollectionExtensions

Inherited Members

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

Methods

RenameKey<TKey, TValue>(IDictionary<TKey, TValue>, TKey, TKey)

Rename a key contained in a dictionary. This method is thread-safe.

```
public static void RenameKey<TKey, TValue>(this IDictionary<TKey, TValue> dictionary, TKey
oldKey, TKey newKey) where TKey : notnull
```

Parameters

`dictionary` IDictionary<TKey, TValue>

The dictionary to perform the rename on.

`oldKey` TKey

The name of the old key.

`newKey` TKey

The name to change `oldKey` to.

Type Parameters

`TKey`

The Type of the key.

`TValue`

The Type of the value.

Remarks

If the `newKey` is the same as the `oldKey` using the default equality operator, no operation is performed.

Exceptions

ArgumentNullException

If `dictionary` is `null`.

ArgumentException

If `newKey` already exists in `dictionary`.

Class ControlExtensions

Namespace: AetherUtils.Core.Extensions

Assembly: AetherUtils.Core.dll

Provides extension methods for manipulating WinForm Control objects.

```
public static class ControlExtensions
```

Inheritance

object ← ControlExtensions

Inherited Members

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

Methods

InvokeIfRequired<T>(T, Action<T>)

Invoke the specified action on the specified control, if required.

```
public static void InvokeIfRequired<T>(this T control, Action<T> action) where T
: ISynchronizeInvoke
```

Parameters

`control` T

The Control to invoke an action on.

`action` Action<T>

The Action to invoke on the control.

Type Parameters

`T`

The Control type.

ResizeColumns(ListView)

Resizes the columns of a ListView control to be a best-fit compromise between the header and the content.

```
public static void ResizeColumns(this ListView listView)
```

Parameters

`listView` ListView

The ListView to auto-size the columns of.

Class DataTableExtensions

Namespace: AetherUtils.Core.Extensions

Assembly: AetherUtils.Core.dll

Provides extension methods for manipulating DataTable objects.

```
public static class DataTableExtensions
```

Inheritance

object ← DataTableExtensions

Inherited Members

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

Methods

ToCsvFile(DataTable, string, char)

Create and save a generated CSV file from a DataTable.

```
public static void ToCsvFile(this DataTable table, string filePath, char delimiter = ',')
```

Parameters

`table` DataTable

The DataTable to create a CSV file from.

`filePath` string

The path of a new CSV file to create or overwrite.

`delimiter` char

The delimiter to separate fields with; default is `,`.

Exceptions

ArgumentNullException

If `table` was `null`.

ArgumentException

If `filePath` was `null` or empty.

ToCsvFiles(List<DataTable>, string, char)

Create and save individual CSV files based on a List<T> containing DataTable objects. Each file is saved as `tablename.csv`.

```
public static void ToCsvFiles(this List<DataTable> tables, string folderPath, char delimiter
= ',')
```

Parameters

`tables` List<DataTable>

The List<T> containing DataTable objects.

`folderPath` string

The folder path that the CSV files should be saved into.

`delimiter` char

The delimiter to separate fields with; default is `,`.

Exceptions

ArgumentNullException

If `tables` was `null`.

ArgumentException

If `folderPath` was `null` or empty.

Class EnumExtensions

Namespace: AetherUtils.Core.Extensions

Assembly: AetherUtils.Core.dll

Provides extension methods for manipulating Enum objects.

```
public static class EnumExtensions
```

Inheritance

object ← EnumExtensions

Inherited Members

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

Methods

ToDescriptionString<T>(T)

Get the string associated with a DescriptionAttribute attribute on an Enum value.

```
public static string ToDescriptionString<T>(this T val) where T : Enum
```

Parameters

`val` T

The Enum value to get the description string of.

Returns

string

The description string or Empty if no DescriptionAttribute was found.

Type Parameters

`T`

The Enum type to get the attributes of.

Class ImageExtensions

Namespace: AetherUtils.Core.Extensions

Assembly: AetherUtils.Core.dll

Provides extension methods for manipulating Image objects.

```
public static class ImageExtensions
```

Inheritance

object ← ImageExtensions

Inherited Members

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

Methods

IsValidImage(string)

Get a value indicating if the image specified by the `filePath` is a valid image file.

```
public static bool IsValidImage(this string filePath)
```

Parameters

`filePath` string

The file path to an image file.

Returns

bool

`true` if the image was valid; `false` otherwise.

Exceptions

FormatException

If the image specified by `filePath` was not in the correct format.

FileNotFoundException

If the file specified by `filePath` was not found.

ResizeImage(Image, int, int)

Resize the Image to the specified width and height.

```
public static Image ResizeImage(this Image image, int newWidth, int newHeight)
```

Parameters

`image` Image

The Image to resize.

`newWidth` int

The width to resize to.

`newHeight` int

The height to resize to.

Returns

Image

The resized Image.

Exceptions

ArgumentNullException

If `image` was `null`.

InvalidOperationException

If the `newWidth` or `newHeight` was less than 0.

ResizeImageScaled(Image, float, float)

Resize the image to be the desired width and height while attempting to maintain the aspect ratio.

```
public static Image ResizeImageScaled(this Image image, float desiredWidth,
float desiredHeight)
```

Parameters

`image` Image

The Image to resize.

`desiredWidth` float

The desired width of the new image.

`desiredHeight` float

The desired height of the new image.

Returns

Image

The scaled Image.

Exceptions

ArgumentNullException

If `image` was `null`.

InvalidOperationException

If the `desiredWidth` or `desiredHeight` was less than 0.

ToByteArray(Image)

Convert an Image into a byte array.

```
public static byte[] ToByteArray(this Image image)
```

Parameters

`image` Image

The image to convert.

Returns

byte[]

A byte array representing the image.

Exceptions

ArgumentNullException

If the `image` was `null`.

Class NumberExtensions

Namespace: AetherUtils.Core.Extensions

Assembly: AetherUtils.Core.dll

Provides extension methods for manipulating various number objects.

```
public static class NumberExtensions
```

Inheritance

object ← NumberExtensions

Inherited Members

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

Methods

FormatSize(ulong)

Formats a size in bytes (represented by an ulong value) to the next closest base-2 size representation and appends its suffix to the end; uses `1024` as the conversion factor.

```
public static string FormatSize(this ulong sizeInBytes)
```

Parameters

`sizeInBytes` ulong

The size, in bytes, to format.

Returns

string

The number formatted with its suffix as a string.

Exceptions

ArgumentNullException

If `sizeInBytes` was `null`.

ToInches(double, double)

Get an absolute value in inches represented by the specified number of pixels.

```
public static double ToInches(this double pixels, double dpi = 96)
```

Parameters

`pixels` double

The number of pixels to convert.

`dpi` double

The DPI (dots per inch) of the screen; defaults to `96`.

Returns

double

The inches equivalent of the pixels.

Exceptions

ArgumentNullException

If `pixels` was `null`.

DivideByZeroException

If `dpi` was 0.

ToPixels(double, double)

Get an absolute value in pixels represented by the specified number of inches.

```
public static double ToPixels(this double inches, double dpi = 96)
```

Parameters

`inches` double

The number of inches to convert.

`dpi` double

The DPI (dots per inch) of the screen; default is `96`.

Returns

double

The pixels equivalent of the inches.

Exceptions

ArgumentNullException

If `inches` was `null`.

Class SizeExtensions

Namespace: AetherUtils.Core.Extensions

Assembly: AetherUtils.Core.dll

Provides extension methods for manipulating Size objects.

```
public static class SizeExtensions
```

Inheritance

object ← SizeExtensions

Inherited Members

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

Methods

GetCenterPoint(Size)

Get the center Point for the specified Size.

```
public static Point GetCenterPoint(this Size size)
```

Parameters

`size` Size

The Size to get the center of.

Returns

Point

A Point representing the center point of the Size.

Exceptions

ArgumentNullException

If the `size` was `null`.

GetCenterPointF(SizeF)

Get the center PointF for the specified SizeF.

```
public static PointF GetCenterPointF(this SizeF size)
```

Parameters

`size` SizeF

The SizeF to get the center of.

Returns

PointF

A PointF representing the center point of the SizeF.

Exceptions

ArgumentNullException

If the `size` was `null`.

Class StreamExtensions

Namespace: AetherUtils.Core.Extensions

Assembly: AetherUtils.Core.dll

Provides extension methods for manipulating Stream objects.

```
public static class StreamExtensions
```

Inheritance

object ← StreamExtensions

Inherited Members

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

Methods

ToImage(Stream)

Get an Image represented by a Stream.

```
public static Image ToImage(this Stream stream)
```

Parameters

`stream` Stream

The Stream containing properly formatted image data.

Returns

Image

A new Image.

Exceptions

ArgumentNullException

If `stream` was `null`.

FormatException

If `stream` did not have a valid ImageFormat.

Class StringExtensions

Namespace: AetherUtils.Core.Extensions

Assembly: AetherUtils.Core.dll

Provides extension methods for manipulating string objects.

```
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

BytesFromString(string)

Get a byte array representing the string using UTF8.

```
public static byte[] BytesFromString(this string value)
```

Parameters

`value` string

A string to get the bytes of.

Returns

byte[]

A byte array containing the bytes representing the string.

BytesToEncodedBytes(byte[], HashEncoding)

Encode the byte array to the specified HashEncoding.

```
public static byte[] BytesToEncodedBytes(this byte[] bytes, HashEncoding encoding)
```

Parameters

`bytes` byte[]

A byte array to encode.

`encoding` HashEncoding

The HashEncoding to use.

Returns

byte[]

A byte array containing the encoded bytes.

BytesToEncodedString(byte[], HashEncoding)

Encode the byte array to the specified HashEncoding.

```
public static string BytesToEncodedString(this byte[] bytes, HashEncoding encoding)
```

Parameters

`bytes` byte[]

A byte array to encode.

`encoding` HashEncoding

The HashEncoding to use.

Returns

string

A string encoded using the specified HashEncoding.

DecodedBytesFromEncodedString(string, HashEncoding)

Decodes the encoded string according to the specified HashEncoding.

```
public static byte[] DecodedBytesFromEncodedString(this string value, HashEncoding encoding)
```

Parameters

`value` string

The encoded string.

`encoding` HashEncoding

The HashEncoding to use.

Returns

byte[]

A byte array containing the decoded bytes.

DecodedStringFromEncodedString(string, HashEncoding)

Decodes an encoded string according to the specified HashEncoding.

```
public static string DecodedStringFromEncodedString(this string value,
HashEncoding encoding)
```

Parameters

`value` string

The encoded string.

`encoding` HashEncoding

The HashEncoding to use.

Returns

string

A decoded string.

DeserializeJson<T>(string)

Deserializes an object from a JSON string.

```
public static T? DeserializeJson<T>(this string json) where T : class
```

Parameters

`json` string

The JSON string to deserialize.

Returns

T

The deserialized object or `null` if the deserialization failed.

Type Parameters

`T`

The type of object to deserialize to.

Exceptions

ArgumentException

If `json` was `null` or empty.

DeserializeXml<T>(string)

Deserializes an object from an XML string.

```
public static T? DeserializeXml<T>(this string xml) where T : class
```

Parameters

`xml` string

The XML string to deserialize.

Returns

T

The deserialized object or `null` if the deserialization failed.

Type Parameters

`T`

The type of object to deserialize to.

Exceptions

ArgumentException

If `xml` was `null` or empty.

FromSecureString(SecureString)

Convert a SecureString to an unsecured string.

```
public static string FromSecureString(this SecureString secure)
```

Parameters

`secure` SecureString

The SecureString.

Returns

string

The unsecured string.

Exceptions

ArgumentNullException

If `secure` was `null`.

ImageFromString(string)

Convert the Base64 representation of a picture into a usable Image object for drawing.

```
public static Image ImageFromString(this string base64)
```

Parameters

`base64` string

The Image as a Base64 string.

Returns

Image

An Image or `null` if `base64` is an invalid string.

Exceptions

ArgumentNullException

If `base64` was `null`.

FormatException

If the input was not a valid Base64 string.

IsBase64Encoded(string)

Get a value indicating if the string appears to be Base64 encoded.

```
public static bool IsBase64Encoded(this string input)
```

Parameters

`input` string

The string to check encoding on.

Returns

bool

`true` if the string is Base64 encoded; `false` otherwise.

Exceptions

ArgumentNullException

If `input` was `null`.

IsHexEncoded(string)

Get a value indicating if the string appears to be Hex (Base16) encoded.

```
public static bool IsHexEncoded(this string input)
```

Parameters

`input` string

The string to check encoding on.

Returns

bool

`true` if the string is Hex encoded; `false` otherwise.

Exceptions

ArgumentNullException

If `input` was `null`.

RemoveWhitespace(string)

Removes all whitespace characters from the specified string.

```
public static string RemoveWhitespace(this string str)
```

Parameters

`str` string

The string to remove whitespace from.

Returns

string

A new string, without the whitespace characters.

StringToEncodedBytes(string, HashEncoding)

Encode the string to the specified HashEncoding.

```
public static byte[] StringToEncodedBytes(this string value, HashEncoding encoding)
```

Parameters

`value` string

The string to encode.

`encoding` HashEncoding

The HashEncoding to use.

Returns

byte[]

A byte array containing the encoded bytes.

StringToEncodedString(string, HashEncoding)

Encode the string to the specified HashEncoding.

```
public static string StringToEncodedString(this string value, HashEncoding encoding)
```

Parameters

`value` string

The string to encode.

`encoding` HashEncoding

The HashEncoding to use.

Returns

string

A string encoded using the specified HashEncoding.

ToSecureString(string)

Convert an unsecure string value into a SecureString.

```
public static SecureString ToSecureString(this string unsecure)
```

Parameters

`unsecure` string

The unsecure string.

Returns

SecureString

A new read-only SecureString.

Exceptions

ArgumentNullException

If `unsecure` was `null`.

Trim(string, int)

Trims the string to be the length specified before a new-line character is inserted.

If the new-line character would be inserted at the position of a period (`.`), the new line is inserted at the index of `(.) + 1`. If the line being checked contains a new-line character already, nothing is done for that line.

```
public static TrimmedString Trim(this string input, int lineLength = 80)
```

Parameters

`input` string

The string to trim.

`lineLength` int

The number of characters in the line; default is `80`.

Returns

TrimmedString

A new TrimmedString containing the trimmed string and the number of new lines.

Exceptions

ArgumentNullException

If `input` was `null`.

Class TimeExtensions

Namespace: AetherUtils.Core.Extensions

Assembly: AetherUtils.Core.dll

Provides extension methods for manipulating various time related objects.

```
public static class TimeExtensions
```

Inheritance

object ← TimeExtensions

Inherited Members

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

Methods

EndOfMonth(DateTime)

Get the ending DateTime of the month based on the supplied DateTime.

```
public static DateTime EndOfMonth(this DateTime date)
```

Parameters

`date` DateTime

The DateTime representing the current date for the month.

Returns

DateTime

A DateTime representing the end of the month.

Exceptions

ArgumentNullException

If the `date` was `null`.

StartOfMonth(DateTime)

Get the starting DateTime of the month based on the supplied DateTime.

```
public static DateTime StartOfMonth(this DateTime date)
```

Parameters

`date` DateTime

The DateTime to get the start of the month of.

Returns

DateTime

A DateTime representing the start of the month.

Exceptions

ArgumentNullException

If the `date` was `null`.

StartOfWeek(DateTime)

Get the starting DateTime of the week based on the supplied DateTime.

```
public static DateTime StartOfWeek(this DateTime date)
```

Parameters

`date` DateTime

The DateTime to get the start of the week of.

Returns

DateTime

A DateTime representing the start of the week.

Exceptions

ArgumentNullException

If the `date` was `null`.

TimeSpanToString(TimeSpan)

Get a string representing the TimeSpan in the form of:
`w day(s), x hour(s), y minute(s), z second(s)`

```
public static string TimeSpanToString(this TimeSpan timeSpan)
```

Parameters

`timeSpan` TimeSpan

The TimeSpan to convert.

Returns

string

A string representing the TimeSpan.

Exceptions

ArgumentNullException

If `timeSpan` was `null`.

Class TypeExtensions

Namespace: AetherUtils.Core.Extensions

Assembly: AetherUtils.Core.dll

Provides extension methods for manipulating generic object types.

```
public static class TypeExtensions
```

Inheritance

object ← TypeExtensions

Inherited Members

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

Methods

CanSerializeJson<T>(T)

Get a value indicating if this object can be serialized via JSON serializion.

```
public static bool CanSerializeJson<T>(this T obj) where T : class
```

Parameters

`obj` T

The instance of the object with type `T`.

Returns

bool

`true` if the object can be serialized; `false` otherwise.

Type Parameters

`T`

The Type of the object to check.

Exceptions

ArgumentNullException

If `obj` was `null`.

CanSerializeXml<T>(T)

Get a value indicating if this object can be serialized via XML serialization.

```
public static bool CanSerializeXml<T>(this T obj) where T : class
```

Parameters

`obj` T

The instance of the object with type `T`.

Returns

bool

`true` if the object can be serialized; `false` otherwise.

Type Parameters

`T`

The Type of the object to check.

Exceptions

ArgumentNullException

If `obj` was `null`.

GetFriendlyName(Type)

Get the friendly, displayable name for the Type.

```
public static string GetFriendlyName(this Type type)
```

Parameters

`type` Type

The Type to get the name of.

Returns

string

The friendly name for the Type.

Exceptions

ArgumentNullException

If `type` was `null`.

SerializeJson<T>(T)

Serializes a .NET object to a JSON string.

```
public static string SerializeJson<T>(this T obj) where T : class
```

Parameters

`obj` T

An instance of `T` to serialize.

Returns

string

A JSON string representing the serialized object, `obj`.

Type Parameters

`T`

The Type of the object to serialize.

Exceptions

ArgumentNullException

If `obj` was `null`.

SerializeXml<T>(T)

Serializes a .NET object to an XML string.

```
public static string SerializeXml<T>(this T obj) where T : class
```

Parameters

`obj` T

An instance of `T` to serialize.

Returns

string

An XML string representing the serialized object, `obj`.

Type Parameters

`T`

The Type of the object to serialize.

Exceptions

ArgumentNullException

If `obj` was `null`.

Namespace AetherUtils.Core.Files

Classes

FileHelper

Provides methods to manipulate files and folders on the Windows file system.

Json<T>

Implements serializing and de-serializing generic object types to/from JSON files.

Xml<T>

Implements serializing and de-serializing generic object types to/from XML files.

Class FileHelper

Namespace: AetherUtils.Core.Files

Assembly: AetherUtils.Core.dll

Provides methods to manipulate files and folders on the Windows file system.

```
public static class FileHelper
```

Inheritance

object ← FileHelper

Inherited Members

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

Methods

CreateDirectories(string, bool)

Create directories along the specified file path if they don't already exist.

```
public static void CreateDirectories(string filePath, bool expandPath = true)
```

Parameters

`filePath` string

The path to create directories on.

`expandPath` bool

Should the `filePath` be expanded before creating directories?

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

DeleteFile(string, bool)

Delete the file, if it exists, specified by `filePath`.

```
public static void DeleteFile(string filePath, bool expandPath = true)
```

Parameters

`filePath` string

The path to a file to delete.

`expandPath` bool

Should the `filePath` be expanded before accessing?

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

DoesFileExist(string, bool)

Check if a file exists on the file system.

```
public static bool DoesFileExist(string filePath, bool expandPath = true)
```

Parameters

`filePath` string

The path to a file.

`expandPath` bool

Should the `filePath` be expanded before checking?

Returns

bool

`true` if the file exists; `false` otherwise.

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

DoesFolderExist(string, bool)

Check if a folder exists on the file system.

```
public static bool DoesFolderExist(string folderPath, bool expandPath = true)
```

Parameters

`folderPath` string

The path to a folder.

`expandPath` bool

Should the `folderPath` be expanded before checking?

Returns

bool

`true` if the folder exists; `false` otherwise.

Exceptions

ArgumentException

If the `folderPath` was `null` or empty.

ExpandPath(string)

Expand a file path containing environment path variables, if possible.

```
public static string ExpandPath(string filePath)
```

Parameters

`filePath` string

The path to expand.

Returns

string

The expanded path, or the original path if no expansion was necessary.

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

GetExtension(string, bool)

Get the extension component, including the `.`, of a file name specified by `filePath`.

```
public static string GetExtension(string filePath, bool expandPath = true)
```

Parameters

`filePath` string

The path to a file.

`expandPath` bool

Should the `filePath` be expanded before accessing?

Returns

string

The file extension, including the `.`.

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

IsValidPath(string)

Get whether the specified `path` is a valid absolute file path on Windows.

```
public static bool IsValidPath(string path)
```

Parameters

`path` string

A path to check.

Returns

bool

Exceptions

ArgumentException

If the `path` was `null` or empty.

OpenFile(string, bool)

Open a text file, read all text within to a string, and close the file.

```
public static string OpenFile(string filePath, bool expandPath = true)
```

Parameters

`filePath` string

The path to the file.

`expandPath` bool

Should the `filePath` be expanded before attempting to open the file?

Returns

string

The contents of `filePath` or Empty if the file could not be opened.

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

OpenFileAsync(string, bool)

Open a text file, read all text within asynchronously to a string, and close the file.

```
public static Task<string> OpenFileAsync(string filePath, bool expandPath = true)
```

Parameters

`filePath` string

The path to the file.

`expandPath` bool

Should the `filePath` be expanded before attempting to open the file?

Returns

Task<string>

A Task<TResult> which contains the string contents upon completion.

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

OpenNonTextFile(string, bool)

Open a binary file, read all bytes within to a byte array, and close the file.

```
public static byte[] OpenNonTextFile(string filePath, bool expandPath = true)
```

Parameters

`filePath` string

The path to the file.

`expandPath` bool

Should the `filePath` be expanded before attempting to open the file?

Returns

byte[]

The contents of `filePath` as a byte array.

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

OpenNonTextFileAsync(string, bool)

Open a binary file asynchronously, read all bytes within to a byte array, and close the file.

```
public static Task<byte[]> OpenNonTextFileAsync(string filePath, bool expandPath = true)
```

Parameters

`filePath` string

The path to the file.

`expandPath` bool

Should the `filePath` be expanded before attempting to open the file?

Returns

Task<byte[]>

A Task<TResult> which contains the byte array upon completion.

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

RemoveInvalidFileNameChars(string)

Remove the platform-specific invalid file name characters from the specified file name.

```
public static string RemoveInvalidFileNameChars(string fileName)
```

Parameters

`fileName` string

The filename.

Returns

string

A string with the invalid characters removed from the filename.

Exceptions

ArgumentException

If the `fileName` was `null` or empty.

RemoveInvalidPathChars(string)

Remove the platform-specific invalid path characters from the specified path.

```
public static string RemoveInvalidPathChars(string path)
```

Parameters

`path` string

The path.

Returns

string

A string with the invalid characters removed from the path.

Exceptions

ArgumentException

If the `path` was `null` or empty.

SaveFile(string, byte[], bool)

Create a new file and write the specified byte array to it. If the file exists, it is overwritten.

```
public static void SaveFile(string filePath, byte[] contents, bool expandPath = true)
```

Parameters

`filePath` string

The path of the file to save.

`contents` byte[]

The bytes to save to the file.

`expandPath` bool

Should the `filePath` be expanded before creating file?

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

SaveFile(string, string, bool)

Create a new file and write the specified content to it.
If the file already exists, it is overwritten.

```
public static void SaveFile(string filePath, string content, bool expandPath = true)
```

Parameters

`filePath` string

The path to the file.

`content` string

The text to save to the file.

`expandPath` bool

Should the `filePath` be expanded before creating file?

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

SaveFile(string, string, Encoding, bool)

Create a new file and write the specified content to it, using the specified encoding. If the file exists, it is overwritten.

```
public static void SaveFile(string filePath, string content, Encoding encoding, bool
expandPath = true)
```

Parameters

`filePath` string

The path of the file to save.

`content` string

The text to save to the file.

`encoding` Encoding

The Encoding to use when saving the file.

`expandPath` bool

Should the `filePath` be expanded before creating file?

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

SaveFileAsync(string, byte[], bool)

Asynchronously create a new file and write the specified byte array to it. If the file exists, it is overwritten.

```
public static void SaveFileAsync(string filePath, byte[] content, bool expandPath = true)
```

Parameters

`filePath` string

The path of the file to save.

`content` byte[]

The bytes to save to the file.

`expandPath` bool

Should the `filePath` be expanded before creating file?

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

SaveFileAsync(string, string, bool)

Asynchronously create a new file and write the specified content to it.

If the file exists, it is overwritten.

```
public static void SaveFileAsync(string filePath, string content, bool expandPath = true)
```

Parameters

`filePath` string

The path of the file to save.

`content` string

The text to save to the file.

`expandPath` bool

Should the `filePath` be expanded before creating file?

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

SaveFileAsync(string, string, Encoding, bool)

Asynchronously create a new file and write the specified content to it, using the specified encoding. If the file exists, it is overwritten.

```
public static void SaveFileAsync(string filePath, string content, Encoding encoding, bool
expandPath = true)
```

Parameters

`filePath` string

The path of the file to save.

`content` string

The text to save to the file.

`encoding` Encoding

The Encoding to use when saving the file.

`expandPath` bool

Should the `filePath` be expanded before creating file?

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

Class Json<T>

Namespace: AetherUtils.Core.Files

Assembly: AetherUtils.Core.dll

Implements serializing and de-serializing generic object types to/from JSON files.

```
public sealed class Json<T> where T : class
```

Type Parameters

T

The type of object to serialize/deserialize.

Inheritance

object ← Json<T>

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Remarks

T must support JSON serialization.

Constructors

Json()

```
public Json()
```

Methods

FromJson(string)

```
public T? FromJson(string json)
```

Parameters

`json` string

Returns

T

LoadJson(string)

Deserialize and load a .NET object from a JSON file.

```
public T? LoadJson(string filePath)
```

Parameters

`filePath` string

The file to load.

Returns

T

The `T` object, or `null` if the object could not be deserialized.

Exceptions

ArgumentException

If `filePath` was `null` or empty.

FileNotFoundException

If the `filePath` did not exist.

SaveJson(string, T)

Serialize a .NET object, `T`, to a JSON string and save to a file.
If the file already exists, it is overwritten.

```
public bool SaveJson(string filePath, T obj)
```

Parameters

`filePath` string

The file to save.

`obj` T

The .NET object to serialize and save.

Returns

bool

`true` if the object was serialized and the file was saved; `false` otherwise.

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

ToJson(T)

```
public string ToJson(T obj)
```

Parameters

`obj` T

Returns

string

Class Xml<T>

Namespace: AetherUtils.Core.Files

Assembly: AetherUtils.Core.dll

Implements serializing and de-serializing generic object types to/from XML files.

```
public sealed class Xml<T> where T : class
```

Type Parameters

T

The type of object to serialize/deserialize.

Inheritance

object ← Xml<T>

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Remarks

T must support XML serialization.

Methods

LoadXml(string)

Deserialize and load a .NET object from an XML file.

```
public T? LoadXml(string filePath)
```

Parameters

`filePath` string

The file to load.

Returns

T

The `T` object, or `null` if the object could not be deserialized.

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

SaveXml(string, T)

Serialize an object of type `T` to an XML string and save to a file.
If the file already exists, it is overwritten.

```
public bool SaveXml(string filePath, T obj)
```

Parameters

`filePath` string

The file to save.

`obj` T

The object to serialize and save.

Returns

bool

`true` if the object was serialized and the file was saved; `false` otherwise.

Exceptions

ArgumentException

If the `filePath` was `null` or empty.

ArgumentNullException

If the `obj` was `null`.

Namespace AetherUtils.Core.Licensing

Classes

KeyManager

Handles loading and saving private and public keys to file system.

Licenser

Structs

KeyPair

Stores a read-only pair of keys.

Class KeyManager

Namespace: AetherUtils.Core.Licensing

Assembly: AetherUtils.Core.dll

Handles loading and saving private and public keys to file system.

```
public static class KeyManager
```

Inheritance

object ← KeyManager

Inherited Members

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

Methods

DoesKeyFileExist(string, string)

Get a value indicating whether the keyPair.keys file exists at the specified base path.

```
public static bool DoesKeyFileExist(string basePath, string fileName = "keyPair.keys")
```

Parameters

`basePath` string

The base folder the keys are located in.

`fileName` string

The name of the file to check: default is `keyPair.keys`

Returns

bool

`true` if the file was found; `false`, otherwise.

LoadKeys(string)

Load a key pair from a file.

```
public static KeyPair? LoadKeys(string fullPath)
```

Parameters

`fullPath` string

The full, absolute path to a key file.

Returns

KeyPair?

A KeyPair representing the key values.

LoadKeys(string, string)

Load a key pair from a file.

```
public static KeyPair? LoadKeys(string basePath, string fileName = "keyPair.keys")
```

Parameters

`basePath` string

The base folder the keys are located in.

`fileName` string

The name of the file to load: default is `keyPair.keys`

Returns

KeyPair?

A KeyPair representing the key values.

SaveKeys(KeyPair, string, string)

Save the key pair to a file.

```
public static void SaveKeys(KeyPair keys, string basePath, string fileName = "keyPair.keys")
```

Parameters

`keys` KeyPair

The pair of keys (private and public). Private key should be first.

`basePath` string

The base folder to save the keys in.

`fileName` string

The name of the file to save: default is `keyPair.keys`

SavePublicKey(KeyPair, string, string)

Save only the public key to a file.

```
public static void SavePublicKey(KeyPair keys, string basePath, string fileName
= "public.key")
```

Parameters

`keys` KeyPair

The pair of keys (private and public). Private key should be first.

`basePath` string

The base folder to save the public key in.

`fileName` string

The name of the file to save: default is `public.key`

SavePublicKey(string, string, string)

Save only the public key to a file.

```
public static void SavePublicKey(string publicKey, string basePath, string fileName
= "public.key")
```

Parameters

`publicKey` string

The public key to save.

`basePath` string

The base folder to save the public key in.

`fileName` string

The name of the file to save: default is `public.key`

Struct KeyPair

Namespace: AetherUtils.Core.Licensing

Assembly: AetherUtils.Core.dll

Stores a read-only pair of keys.

```
public readonly struct KeyPair
```

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Constructors

KeyPair(string, string)

Stores a read-only pair of keys.

```
public KeyPair(string privateKey, string publicKey)
```

Parameters

`privateKey` string

The private key.

`publicKey` string

The public key.

Properties

PrivateKey

```
public string PrivateKey { get; }
```

Property Value

string

PublicKey

```
public string PublicKey { get; }
```

Property Value

string

Class Licenser

Namespace: AetherUtils.Core.Licensing

Assembly: AetherUtils.Core.dll

```
public static class Licenser
```

Inheritance

object ← Licenser

Inherited Members

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

Methods

Validate(string, string)

Validates a license with the specified public key.

```
public static List<IValidationFailure> Validate(string license, string publicKey)
```

Parameters

`license` string

The license XML string or file path to a license file (`.lic`).

`publicKey` string

The public key to validate against.

Returns

List<IValidationFailure>

A string representing the results of validation.

Namespace AetherUtils.Core.Licensing.Models

Classes

Customer

Feature

License

Class Customer

Namespace: AetherUtils.Core.Licensing.Models

Assembly: AetherUtils.Core.dll

```
public class Customer
```

Inheritance

object ← Customer

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Constructors

Customer()

```
public Customer()
```

Properties

Company

```
public string Company { get; set; }
```

Property Value

string

Email

```
public string Email { get; set; }
```

Property Value

string

Name

```
public string Name { get; set; }
```

Property Value

string

Class Feature

Namespace: AetherUtils.Core.Licensing.Models

Assembly: AetherUtils.Core.dll

```
[Serializable]
public class Feature
```

Inheritance

object ← Feature

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Constructors

Feature()

```
public Feature()
```

Feature(string, string)

```
public Feature(string name, string text)
```

Parameters

`name` string

`text` string

Properties

Name

```
[Browsable(true)]
public string Name { get; set; }
```

Property Value

string

Text

```
[Browsable(true)]
public string Text { get; set; }
```

Property Value

string

Class License

Namespace: AetherUtils.Core.Licensing.Models

Assembly: AetherUtils.Core.dll

```
[TypeConverter(typeof(ExpandableObjectConverter))]
public sealed class License : ExpandableObjectConverter
```

Inheritance

object ← TypeConverter ← ExpandableObjectConverter ← License

Inherited Members

ExpandableObjectConverter.GetProperties(ITypeDescriptorContext, object, Attribute[]) ,
ExpandableObjectConverter.GetPropertiesSupported(ITypeDescriptorContext) ,
TypeConverter.CanConvertFrom(ITypeDescriptorContext, Type) ,
TypeConverter.CanConvertFrom(Type) , TypeConverter.CanConvertTo(ITypeDescriptorContext, Type) ,
TypeConverter.CanConvertTo(Type) ,
TypeConverter.ConvertFrom(ITypeDescriptorContext, CultureInfo, object) ,
TypeConverter.ConvertFrom(object) ,
TypeConverter.ConvertFromInvariantString(ITypeDescriptorContext, string) ,
TypeConverter.ConvertFromInvariantString(string) ,
TypeConverter.ConvertFromString(ITypeDescriptorContext, CultureInfo, string) ,
TypeConverter.ConvertFromString(ITypeDescriptorContext, string) ,
TypeConverter.ConvertFromString(string) ,
TypeConverter.ConvertTo(ITypeDescriptorContext, CultureInfo, object, Type) ,
TypeConverter.ConvertTo(object, Type) ,
TypeConverter.ConvertToInvariantString(ITypeDescriptorContext, object) ,
TypeConverter.ConvertToInvariantString(object) ,
TypeConverter.ConvertToString(ITypeDescriptorContext, CultureInfo, object) ,
TypeConverter.ConvertToString(ITypeDescriptorContext, object) ,
TypeConverter.ConvertToString(object) , TypeConverter.CreateInstance(IDictionary) ,
TypeConverter.CreateInstance(ITypeDescriptorContext, IDictionary) ,
TypeConverter.GetCreateInstanceSupported() ,
TypeConverter.GetCreateInstanceSupported(ITypeDescriptorContext) ,
TypeConverter.GetProperties(ITypeDescriptorContext, object) , TypeConverter.GetProperties(object) ,
TypeConverter.GetPropertiesSupported() , TypeConverter.GetStandardValues() ,
TypeConverter.GetStandardValues(ITypeDescriptorContext) ,
TypeConverter.GetStandardValuesExclusive() ,
TypeConverter.GetStandardValuesExclusive(ITypeDescriptorContext) ,

TypeConverter.GetStandardValuesSupported() ,
TypeConverter.GetStandardValuesSupported(ITypeDescriptorContext) ,
TypeConverter.IsValid(ITypeDescriptorContext, object) , TypeConverter.IsValid(object) ,
object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() ,
object.ReferenceEquals(object, object)

Extension Methods
TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) ,
TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Properties

Customer

```
[Browsable(false)]
public Customer Customer { get; set; }
```

Property Value

Customer

Expiration

```
[Browsable(true)]
public DateTime Expiration { get; set; }
```

Property Value

DateTime

Id

```
[Browsable(false)]
public string Id { get; set; }
```

Property Value

string

ProductFeatures

```
[Browsable(false)]
public List<Feature> ProductFeatures { get; set; }
```

Property Value

List<Feature>

Quantity

```
[Browsable(true)]
public int Quantity { get; set; }
```

Property Value

int

Signature

```
[Browsable(false)]
public string Signature { get; set; }
```

Property Value

string

Type

```
[Browsable(true)]
public LicenseType Type { get; set; }
```

Property Value

LicenseType

Methods

ToString()

Get a string that represents this license object:
ID:
Type:
Expiration:
Quantity:
Customer Name:
Customer Email:

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

Returns

string

A string representing this License.

Namespace AetherUtils.Core.Logging

Classes

CLogger

Provides static methods to facilitate logging to a file as well as to the console for an application. Uses NLog internally.

Class CLogger

Namespace: AetherUtils.Core.Logging

Assembly: AetherUtils.Core.dll

Provides static methods to facilitate logging to a file as well as to the console for an application. Uses NLog internally.

```
public static class CLogger
```

Inheritance

object ← CLogger

Inherited Members

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

Properties

IsInitialized

```
public static bool IsInitialized { get; }
```

Property Value

bool

Methods

GetCurrentLogger(string)

Get a logger reference based on the specified name.

```
public static Logger GetCurrentLogger(string name)
```

Parameters

`name` string

The name of the logger to get.

Returns

Logger

A NLog.Logger object that can be used for logging. Returns default name if logging has not been initialized.

GetCurrentLogger<T>()

Get a logger reference based on the type of class.

```
public static Logger GetCurrentLogger<T>() where T : class
```

Returns

Logger

A NLog.Logger object that can be used for logging. Returns default name if logging has not been initialized via Initialize(LogOptions)

Type Parameters

`T`

The class type to get the logger of.

GetCurrentLogger<T>(string)

Get a logger reference based on the type of class as well as an additional name identifier.

```
public static Logger GetCurrentLogger<T>(string name) where T : class
```

Parameters

`name` string

The additional name to add to the logger. (i.e., the method name)

Returns

Logger

A NLog.Logger object that can be used for logging. Returns default name if logging has not been initialized.

Type Parameters

`T`

The class type to get the logger of.

Initialize(LogOptions)

Initialize the logger for the application. This should be called before any logging takes place; usually, directly after reading or creating the initial configuration for the application.

```
public static void Initialize(LogOptions options)
```

Parameters

`options` LogOptions

The LogOptions that are used to setup the logger.

Namespace AetherUtils.Core.Reflection

Classes

Reflect

Contains methods to help with runtime reflection of classes, types, and attributes.

Class Reflect

Namespace: AetherUtils.Core.Reflection

Assembly: AetherUtils.Core.dll

Contains methods to help with runtime reflection of classes, types, and attributes.

```
public static class Reflect
```

Inheritance

object ← Reflect

Inherited Members

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

Methods

GetAttributesRecurse<T>(object?, Type, HashSet<Type>?)

Retrieve all of the Type, PropertyInfo, Attribute, and objects related to the custom attribute `T` specified, recursively.

```
public static IEnumerable<(Type Class, PropertyInfo Property, Attribute Attribute, object?
Instance)> GetAttributesRecurse<T>(object? instance, Type type, HashSet<Type>? visited =
null) where T : Attribute
```

Parameters

`instance` object

The initial object instance to start the search on.

`type` Type

The initial Type to start the search on.

`visited` HashSet<Type>

Defaults to `null`. Used to maintain a HashSet<T> of already visited types while searching.

Returns

IEnumerable<(Type Class, PropertyInfo Property, Attribute Attribute, object Instance)>

An IEnumerable of tuples:
(Type, PropertyInfo, Attribute, object)

Type Parameters

`T`

The custom attribute to search for.

GetCollectionElementType(Type)

Retrieves the collection element type from this type.

```
public static Type? GetCollectionElementType(Type type)
```

Parameters

`type` Type

The type to query.

Returns

Type

The element type of the collection or `null` if the type was not a collection.

IsList(Type)

Indicates whether or not the specified type is a List<T>.

```
public static bool IsList(Type type)
```

Parameters

`type` Type

The type to query.

Returns

bool

`true` if the type is a List<T>; `false` otherwise

Namespace AetherUtils.Core.RegEx

Classes

RegexGenerator

Class RegexGenerator

Namespace: AetherUtils.Core.RegEx

Assembly: AetherUtils.Core.dll

```
public static class RegexGenerator
```

Inheritance

object ← RegexGenerator

Inherited Members

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

Methods

Base64Regex()

```
[GeneratedRegex("^([A-Za-z0-9+/]{4})*([A-Za-z0-9+/]{3}=|[A-Za-z0-9+/]{2}==)?$",
RegexOptions.CultureInvariant, 1000)]
public static Regex Base64Regex()
```

Returns

Regex

Remarks

Pattern:

```
^([A-Za-z0-9+/]{4})*([A-Za-z0-9+/]{3}=|[A-Za-z0-9+/]{2}==)?$
```

Options:

```
RegexOptions.CultureInvariant
```

Explanation:

- Match if at the beginning of the string.
- Loop greedily any number of times.
    - 1st capture group.
        - Match a character in the set [+/-9A-Za-z] exactly 4 times.
- Optional (greedy).
    - 2nd capture group.
        - Match with 2 alternative expressions.
            - Match a sequence of expressions.
                - Match a character in the set [+/-9A-Za-z] exactly 3 times.
                - Match '='.
            - Match a sequence of expressions.
                - Match a character in the set [+/-9A-Za-z] exactly 2 times.
                - Match the string "==".
- Match if at the end of the string or if before an ending newline.

HexRegex()

```
[GeneratedRegex("^(0x|0X)?[a-fA-F0-9]+$", RegexOptions.CultureInvariant, 1000)]
public static Regex HexRegex()
```

Returns

Regex

Remarks

Pattern:

```
^(0x|0X)?[a-fA-F0-9]+$
```

Options:

```
RegexOptions.CultureInvariant
```

Explanation:

- Match if at the beginning of the string.
- Optional (greedy).
    - 1st capture group.
        - Match '0'.
        - Match a character in the set [Xx].
- Match a character in the set [0-9A-Fa-f] atomically at least once.
- Match if at the end of the string or if before an ending newline.

PathRegex()

```
[GeneratedRegex("^(([a-zA-Z]:)|(\\))(\\{1}|((\\{1})[^\\]([^/:*?<>\"|]*))+)$",
RegexOptions.CultureInvariant, 1000)]
public static Regex PathRegex()
```

Returns

Regex

Remarks

Pattern:

```
^(([a-zA-Z]:)|(\\))(\\{1}|((\\{1})[^\\]([^/:*?<>"|]*))+)$
```

Options:

```
RegexOptions.CultureInvariant
```

Explanation:

- Match if at the beginning of the string.
- 1st capture group.
    - Match with 2 alternative expressions.
        - 2nd capture group.
            - Match a character in the set [A-Za-z].
            - Match ':'.
        - Match a sequence of expressions.
            - 3rd capture group.
                - Match ')'.

- Loop greedily at least once.
    - 4th capture group.
        - Match with 2 alternative expressions.
            - Match the string "{1}".
            - 5th capture group.
                - 6th capture group.
                    - Match the string "{1}".
                - Match a character in the set [^"(*/:<>?[]^|] greedily any number of times.
- Match if at the end of the string or if before an ending newline.

Namespace AetherUtils.Core.Security

Classes

SecretQa

Represents a secret question and answer pair.

Class SecretQa

Namespace: AetherUtils.Core.Security

Assembly: AetherUtils.Core.dll

Represents a secret question and answer pair.

```
public sealed class SecretQa
```

Inheritance

object ← SecretQa

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Remarks

Once instantiated, the question cannot be changed. However, the answer can be changed using Change Answer(string).

Constructors

SecretQa(string, string)

Create a new SecretQa with the specified `question` and `answer`.

```
public SecretQa(string question, string answer)
```

Parameters

`question` string

The question.

`answer` string

The answer.

Properties

Answer

The answer component.

```
public string Answer { get; }
```

Property Value

string

Question

The question component.

```
public string Question { get; }
```

Property Value

string

Methods

ChangeAnswer(string)

Set a new answer for the secret question.

```
public void ChangeAnswer(string newAnswer)
```

Parameters

`newAnswer` string

The new answer to set for the Question.

Namespace AetherUtils.Core.Security.Encryption

Classes

ByteEncryptionService

Provides methods to encrypt and decrypt byte arrays.

EncryptionBase

Represents the base class for all encryption service classes. All encryption is performed using AES-256 and encoded with UTF8.

FileEncryptionService

Provides methods to encrypt and decrypt files on the file system. This service contains potentially destructive methods that can render a file unreadable for a few reasons:

1. If a file is encrypted with EncryptFileAsync(string, string, string) and then opened in a text editor (to view/edit the contents), and then the file is saved, it could render the file unable to be decrypted.
2. If after performing item 1, you attempt to encrypt the file again, it could be lost forever.
3. If the file extension of an encrypted file is changed manually, the file will be unable to be decrypted.

Use at your own risk!

ObjectEncryptionService<T>

Provides methods to encrypt and decrypt XML serializable .NET objects.

StringEncryptionService

Provides methods to encrypt and decrypt strings.

Class ByteEncryptionService

Namespace: AetherUtils.Core.Security.Encryption

Assembly: AetherUtils.Core.dll

Provides methods to encrypt and decrypt byte arrays.

```
public class ByteEncryptionService : EncryptionBase
```

Inheritance

object ← EncryptionBase ← ByteEncryptionService

Inherited Members

EncryptionBase.GetRandomKeyPhrase(int) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Methods

DecryptAsync(byte[], string)

Decrypt the `input` using the specified passphrase.

```
public Task<byte[]> DecryptAsync(byte[] input, string passphrase)
```

Parameters

`input` byte[]

The data to decrypt.

`passphrase` string

The passphrase used to derive the decryption key.

Returns

Task<byte[]>

The decrypted data.

EncryptAsync(byte[], string)

Encrypt the `input` using the specified passphrase.

```
public Task<byte[]> EncryptAsync(byte[] input, string passphrase)
```

Parameters

`input` byte[]

The data to encrypt.

`passphrase` string

A passphrase used to derive the encryption key.

Returns

Task<byte[]>

The encrypted data.

Class EncryptionBase

Namespace: AetherUtils.Core.Security.Encryption

Assembly: AetherUtils.Core.dll

Represents the base class for all encryption service classes. All encryption is performed using AES-256 and encoded with UTF8.

```
public class EncryptionBase
```

Inheritance

object ← EncryptionBase

Derived

ByteEncryptionService, FileEncryptionService, ObjectEncryptionService<T>, StringEncryptionService

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Methods

GetRandomKeyPhrase(int)

Get a cryptographically strong random key phrase that can be used for encryption.

```
public static string GetRandomKeyPhrase(int keySize = 32)
```

Parameters

`keySize` int

The size, in bytes, for the generated key.

Returns

string

A cryptographically strong and random string.

Class FileEncryptionService

Namespace: AetherUtils.Core.Security.Encryption

Assembly: AetherUtils.Core.dll

Provides methods to encrypt and decrypt files on the file system. This service contains potentially destructive methods that can render a file unreadable for a few reasons:

1. If a file is encrypted with EncryptFileAsync(string, string, string) and then opened in a text editor (to view/edit the contents), and then the file is saved, it could render the file unable to be decrypted.
2. If after performing item 1, you attempt to encrypt the file again, it could be lost forever.
3. If the file extension of an encrypted file is changed manually, the file will be unable to be decrypted.

Use at your own risk!

```
public sealed class FileEncryptionService : EncryptionBase
```

Inheritance

object ← EncryptionBase ← FileEncryptionService

Inherited Members

EncryptionBase.GetRandomKeyPhrase(int) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Constructors

FileEncryptionService(string)

```
public FileEncryptionService(string filePath)
```

Parameters

`filePath` string

Properties

FilePath

Get or set the file path that is used for encryption.

```
public string FilePath { get; set; }
```

Property Value

string

Methods

DecryptAsync(string, string)

Decrypt a file specified by `filePath` using the `passphrase`.

```
public Task<string> DecryptAsync(string filePath, string passphrase)
```

Parameters

`filePath` string

The path to the file to decrypt.

`passphrase` string

The passphrase used for decryption.

Returns

Task <string>

The decrypted contents of the file.

Exceptions

FileNotFoundException

If `filePath` was not a valid path to a file.

ArgumentException

If `filePath` was `null` or empty.

DecryptFileAsync(string, string)

Decrypt an encrypted file from disk using the specified `passphrase` to derive the key.

```
public static Task<string> DecryptFileAsync(string filePath, string passphrase)
```

Parameters

`filePath` string

The path to the file on disk to decrypt.

`passphrase` string

The passphrase used for decryption.

Returns

Task<string>

The path to the decrypted file.

Remarks

If the encrypted file's extension is different than the original file's, the encrypted file is deleted after decryption occurs and the file extension is changed back to the original extension.

Exceptions

FileNotFoundException

If the file could not be found at the `filePath`.

EncryptAsync(string, string)

Create and encrypt a file with the specified `content` using the `passphrase`.
This method will either create a new file if one does not exist or overwrite the existing file.

```
public Task<string> EncryptAsync(string content, string passphrase)
```

Parameters

`content` string

The string to encrypt into the file.

`passphrase` string

The passphrase used for encryption.

Returns

Task<string>

The path to the encrypted file.

Exceptions

ArgumentException

If `content` was `null` or empty.

ArgumentNullException

If `passphrase` was `null`.

InvalidOperationException

If the FilePath property is empty.

EncryptFileAsync(string, string, string)

Encrypt an existing file on disk using the specified `passphrase` to derive the key.

```
public static Task<string> EncryptFileAsync(string filePath, string passphrase, string
newExtension = ".enc")
```

Parameters

`filePath` string

The path to the file on disk to encrypt.

`passphrase` string

The passphrase used for encryption.

`newExtension` string

The file extensions to change the encrypted file to. If no file extension change is wanted, supply the original file's extension; default is `.enc`.

Returns

Task<string>

The path to the encrypted file.

Remarks

If the original file's extension is different than the encrypted file's, the original file is deleted after encryption occurs and the file extension is changed to the new extension.

Exceptions

FileNotFoundException

If the file could not be found at the `filePath`.

Class ObjectEncryptionService<T>

Namespace: AetherUtils.Core.Security.Encryption

Assembly: AetherUtils.Core.dll

Provides methods to encrypt and decrypt XML serializable .NET objects.

```
public sealed class ObjectEncryptionService<T> : EncryptionBase where T : class
```

Type Parameters

`T`

The object type to encrypt/decrypt. This type must support XML serialization.

Inheritance

object ← EncryptionBase ← ObjectEncryptionService<T>

Inherited Members

EncryptionBase.GetRandomKeyPhrase(int) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Methods

DecryptAsync(byte[], string)

Decrypt a .NET object represented as an encrypted XML serialized byte array.

```
public Task<T> DecryptAsync(byte[] input, string passphrase)
```

Parameters

`input` byte[]

The serialized, encrypted .NET object as a byte array.

`passphrase` string

The passphrase for decryption.

Returns

Task<T>

The `T` object, decrypted and deserialized.

Exceptions

ArgumentNullException

If the `input` was `null` or 0-length.

FormatException

If the `input` could not be deserialized to type `T`.

DecryptFromFileAsync(string, string)

Decrypt a .NET object from an encrypted file.

```
public Task<T> DecryptFromFileAsync(string filePath, string passphrase)
```

Parameters

`filePath` string

The path to the file containing an encrypted, serialized .NET object.

`passphrase` string

The passphrase for decryption.

Returns

Task<T>

The `T` object, decrypted and deserialized.

Exceptions

FileNotFoundException

If the file specified by `filePath` did not exist.

FormatException

If the object could not be deserialized to a new .NET object.

EncryptAsync(T, string)

Encrypt a .NET XML serializable object to an encrypted byte array.

```
public Task<byte[]> EncryptAsync(T input, string passphrase)
```

Parameters

`input` T

The serializable .NET object.

`passphrase` string

The passphrase for encryption.

Returns

Task<byte[]>

An encrypted .NET object represented as a serialized byte array.

Exceptions

InvalidOperationException

If the `input` could not serialized to type `T`.

EncryptToFileAsync(T, string, string)

Encrypt a .NET XML serializable object to an encrypted file.

```
public Task<byte[]> EncryptToFileAsync(T input, string filePath, string passphrase)
```

Parameters

`input` T

The serializable .NET object.

`filePath` string

The file to create (or overwrite).

`passphrase` string

The passphrase for encryption.

Returns

Task<byte[]>

An encrypted .NET object represented as a serialized byte array; or an empty byte array if the encryption failed.

Exceptions

InvalidOperationException

If the `input` could not serialized to type `T`.

Class StringEncryptionService

Namespace: AetherUtils.Core.Security.Encryption

Assembly: AetherUtils.Core.dll

Provides methods to encrypt and decrypt strings.

```
public sealed class StringEncryptionService : EncryptionBase
```

Inheritance

object ← EncryptionBase ← StringEncryptionService

Inherited Members

EncryptionBase.GetRandomKeyPhrase(int) , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Methods

DecryptAsync(byte[], string)

Decrypt an encrypted byte array using the `passphrase`.

```
public Task<string> DecryptAsync(byte[] encrypted, string passphrase)
```

Parameters

`encrypted` byte[]

An encrypted byte array.

`passphrase` string

The passphrase used for decryption.

Returns

Task<string>

The decrypted string.

EncryptAsync(string, string)

Encrypt a string using the `passphrase`.

```
public Task<byte[]> EncryptAsync(string input, string passphrase)
```

Parameters

`input` string

The string to encrypt.

`passphrase` string

The passphrase used for encryption.

Returns

Task<byte[]>

The encrypted byte array.

Namespace AetherUtils.Core.Security.Hashing

Classes

HashService

Provides a service for creating cryptographically strong hashes of string s and comparing plain-text string s against them for equality.
This class hashes strings based on the provided HashOptions.

Enums

HashEncoding

Represents valid encoding options for hashing.

Enum HashEncoding

Namespace: AetherUtils.Core.Security.Hashing

Assembly: AetherUtils.Core.dll

Represents valid encoding options for hashing.

```
public enum HashEncoding
```

Extension Methods

EnumExtensions.ToDescriptionString<T>(T)

Fields

`Base64 = 64`

Represents BASE-64 encoding.

`Hex = 16`

Represents Hexadecimal encoding (BASE-16).

Class HashService

Namespace: AetherUtils.Core.Security.Hashing

Assembly: AetherUtils.Core.dll

Provides a service for creating cryptographically strong hashes of string s and comparing plain-text string s against them for equality.
This class hashes strings based on the provided HashOptions.

```
public sealed class HashService
```

Inheritance

object ← HashService

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Constructors

HashService(HashOptions)

Create a new service with the specified HashOptions.

```
public HashService(HashOptions options)
```

Parameters

`options` HashOptions

The options that should be used for hashing.

Properties

HashedString

Get the hashed string for this HashService.

```
public string HashedString { get; }
```

Property Value

string

Methods

CompareHash(string)

Compare a non-hashed input string against the last hashed string for this HashService.

```
public bool CompareHash(string input)
```

Parameters

`input` string

Returns

bool

`true` if the two strings are equivalent; `false` otherwise.

Exceptions

ArgumentException

Thrown if `input` is `null` or empty.

CompareHash(string, string)

Compare a non-hashed input string against a hashed string.

```
public static bool CompareHash(string input, string hashString)
```

Parameters

`input` string

The non-hashed string.

`hashString` string

A hash string to compare against.

Returns

bool

`true` if the two strings are equivalent; `false` otherwise.

Exceptions

ArgumentException

Thrown if `input` or `hashString` are `null` or empty.

HashString(string)

Generate a hash string for the specified plain-text string according to this object's HashOptions.

```
public string HashString(string value)
```

Parameters

`value` string

The plain text string to hash.

Returns

string

A cryptographically strong hashed string.

Exceptions

ArgumentNullException

If `value` was `null`.

Namespace AetherUtils.Core.Security.Passwords

Classes

PasswordRule

Represents a password rule with various options. A password rule should be built using the Rule Builder factory: IPasswordRuleBuilder.
This class contains methods to parse and save a password rule to/from a file as well as a string.

A password can be validated against a rule using Validate(string).

PasswordRuleData

Holds the data relating to a PasswordRule built using IPasswordRuleBuilder.

Interfaces

IPasswordRuleBuilder

Builder interface for creating a new PasswordRule.

Interface IPasswordRuleBuilder

Namespace: AetherUtils.Core.Security.Passwords

Assembly: AetherUtils.Core.dll

Builder interface for creating a new PasswordRule.

```
public interface IPasswordRuleBuilder
```

Extension Methods
TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Methods

AllowNumbers()

Allows the password to contain numbers.

```
IPasswordRuleBuilder AllowNumbers()
```

Returns

IPasswordRuleBuilder

The builder instance.

AllowSpecials()

Allows the password to contain special characters.

```
IPasswordRuleBuilder AllowSpecials()
```

Returns

IPasswordRuleBuilder

The builder instance.

AllowWhitespace()

Allows the password to contain whitespace.

```
IPasswordRuleBuilder AllowWhitespace()
```

Returns

IPasswordRuleBuilder

The builder instance.

Build()

Build and compile the password rule.

```
PasswordRule Build()
```

Returns

PasswordRule

The built PasswordRule.

Expires(DateTime)

Set the date that a password should expire at.

```
IPasswordRuleBuilder Expires(DateTime expires)
```

Parameters

`expires` DateTime

The expiration expires of passwords validated against the rule.

Returns

IPasswordRuleBuilder

The builder instance.

MinimumLength(int)

Set the minimum length a password should be.

```
IPasswordRuleBuilder MinimumLength(int length)
```

Parameters

`length` int

Returns

IPasswordRuleBuilder

The builder instance.

MinimumNumberCount(int)

Set the minimum count of numbers a password should contain.

```
IPasswordRuleBuilder MinimumNumberCount(int count)
```

Parameters

`count` int

The minimum number of digits allowed.

Returns

IPasswordRuleBuilder

The builder instance.

MinimumSpecialCount(int)

Set the minimum count of special characters a password should contain.

```
IPasswordRuleBuilder MinimumSpecialCount(int count)
```

Parameters

`count` int

The minimum number of special characters allowed.

Returns

IPasswordRuleBuilder

The builder instance.

Class PasswordRule

Namespace: AetherUtils.Core.Security.Passwords

Assembly: AetherUtils.Core.dll

Represents a password rule with various options. A password rule should be built using the Rule Builder factory: IPasswordRuleBuilder.
This class contains methods to parse and save a password rule to/from a file as well as a string.

A password can be validated against a rule using Validate(string).

```
public sealed class PasswordRule
```

Inheritance

object ← PasswordRule

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Properties

PasswordRules

Get the string representing this password rule.

```
public string PasswordRules { get; }
```

Property Value

string

Methods

FromEncryptedString(string, string)

Parse a password rule from an encrypted string.

```
public static PasswordRule FromEncryptedString(string encrypted, string passphrase)
```

Parameters

`encrypted` string

The encrypted password rule to parse.

`passphrase` string

The passphrase used for decryption.

Returns

PasswordRule

The PasswordRule or `null` if parsing failed.

GetValidPassword()

Get a random, cryptographically strong password that follows the password rule.

```
public string GetValidPassword()
```

Returns

string

A new, random password that adheres to the password rule created.

New()

Start building the password rule.

```
public static IPasswordRuleBuilder New()
```

Returns

IPasswordRuleBuilder

The builder instance.

Parse(string)

Parse a password rule from a Json string.

```
public static PasswordRule Parse(string json)
```

Parameters

`json` string

The password rule as Json to parse.

Returns

PasswordRule

The PasswordRule or `null` if parsing failed.

ParseFromFileAsync(string, string)

Parse an encrypted password rule from a file.

```
public static Task<PasswordRule?> ParseFromFileAsync(string filePath, string passphrase)
```

Parameters

`filePath` string

The path to the file to load; can contain Windows path variables (i.e., `%temp%`)

`passphrase` string

The passphrase used for decryption. Should be the same as the one used for encryption.

Returns

Task<PasswordRule>

The PasswordRule or `null` if parsing failed.

SaveToFileAsync(string, string)

Save the encrypted password rule to a file.

```
public Task<bool> SaveToFileAsync(string filePath, string passphrase)
```

Parameters

`filePath` string

The path to the file to save; can contain Windows path variables (i.e., `%temp%`)

`passphrase` string

The passphrase used for encryption.

Returns

Task<bool>

`true` if the password rule saved successfully; `false` otherwise.

ToEncryptedString(string)

Get an encrypted string representing this password rule.

```
public string ToEncryptedString(string passphrase)
```

Parameters

`passphrase` string

The passphrase used for encryption.

Returns

string

A base64 string representing the password rule.

ToString()

Get the Json string representing this password rule.

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

Returns

string

Validate(string)

Validate a password against the password rule.

```
public List<IPasswordValidationFailure> Validate(string password)
```

Parameters

`password` string

The password to validate.

Returns

List<IPasswordValidationFailure>

A list of IPasswordValidationFailure containing the reasons the password failed to validate. If this list is empty, the password validated successfully.

Class PasswordRuleData

Namespace: AetherUtils.Core.Security.Passwords

Assembly: AetherUtils.Core.dll

Holds the data relating to a PasswordRule built using IPasswordRuleBuilder.

```
public class PasswordRuleData
```

Inheritance

object ← PasswordRuleData

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Properties

Expiration

```
public DateTime? Expiration { get; set; }
```

Property Value

DateTime?

MinimumLengthAllowed

```
public int MinimumLengthAllowed { get; set; }
```

Property Value

int

MinimumNumberCount

```
public int MinimumNumberCount { get; set; }
```

Property Value

int

MinimumSpecialCount

```
public int MinimumSpecialCount { get; set; }
```

Property Value

int

NumbersAllowed

```
public bool NumbersAllowed { get; set; }
```

Property Value

bool

SpecialsAllowed

```
public bool SpecialsAllowed { get; set; }
```

Property Value

bool

WhitespaceAllowed

```
public bool WhitespaceAllowed { get; set; }
```

Property Value

bool

Namespace AetherUtils.Core.Security.Passwords.Validation

Classes

PasswordValidator

Validates a password against a PasswordRule. This class cannot be inherited and should only be called from an instance of PasswordRule to validate a password.

ValidationFailure

Represents a validation failure when checking a password against a password rule.

Interfaces

IPasswordValidationFailure

Represents a failure of a password rule check.

Interface IPasswordValidationFailure

Namespace: AetherUtils.Core.Security.Passwords.Validation

Assembly: AetherUtils.Core.dll

Represents a failure of a password rule check.

```
public interface IPasswordValidationFailure
```

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Properties

HowToResolve

Gets or sets the message that describes how to resolve the validation failure.

```
string HowToResolve { get; set; }
```

Property Value

string

Message

Gets or sets the message that describes the password validation failure.

```
string Message { get; set; }
```

Property Value

string

Class PasswordValidator

Namespace: AetherUtils.Core.Security.Passwords.Validation

Assembly: AetherUtils.Core.dll

Validates a password against a PasswordRule. This class cannot be inherited and should only be called from an instance of PasswordRule to validate a password.

```
public abstract class PasswordValidator
```

Inheritance

object ← PasswordValidator

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Class ValidationFailure

Namespace: AetherUtils.Core.Security.Passwords.Validation

Assembly: AetherUtils.Core.dll

Represents a validation failure when checking a password against a password rule.

```
public class ValidationFailure : IPasswordValidationFailure
```

Inheritance

object ← ValidationFailure

Implements

IPasswordValidationFailure

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Constructors

ValidationFailure()

```
public ValidationFailure()
```

ValidationFailure(string, string)

Represents a validation failure when checking a password against a password rule.

```
public ValidationFailure(string message, string howToResolve)
```

Parameters

`message` string

The message associated with the failure.

`howToResolve` string

Indicates how to fix the failure.

Properties

HowToResolve

Gets or sets the message that describes how to resolve the validation failure.

```
public string HowToResolve { get; set; }
```

Property Value

string

Message

Gets or sets the message that describes the password validation failure.

```
public string Message { get; set; }
```

Property Value

string

Namespace AetherUtils.Core.Security.TwoFactor

Classes

Encoder

Class that handles encoding strings and byte arrays to/from URLs and Base-32.

SetupCode

Represents a setup code needed by the user to setup two factor authentication.

TwoFactorAuth

Provides methods to generate and validate two-factor authentication codes.

TwoFactorUser

Represents a 2-factor authentication user.

Enums

HashType

Represents the various hash types valid for two factor authentication keys.

Class Encoder

Namespace: AetherUtils.Core.Security.TwoFactor

Assembly: AetherUtils.Core.dll

Class that handles encoding strings and byte arrays to/from URLs and Base-32.

```
public static class Encoder
```

Inheritance

object ← Encoder

Inherited Members

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

Methods

Base32Encode(byte[])

Encode the byte array of data to a base-32 encoded string.

```
public static string Base32Encode(byte[] data)
```

Parameters

`data` byte[]

The byte array containing the data.

Returns

string

A string representing the `data` in base-32.

Exceptions

ArgumentNullException

If `data` is `null` or empty.

ToBytes(string)

Encode the base-32 string to a byte array.

```
public static byte[] ToBytes(string input)
```

Parameters

`input` string

The base-32 encoded string.

Returns

byte[]

A byte array.

Exceptions

ArgumentException

If `input` is `null` or empty.

UrlEncode(string)

Encode the string `value` to a valid URL string according to the OATH specification.

```
public static string UrlEncode(string value)
```

Parameters

`value` string

The string to URL encode.

Returns

string

A URL encoded string.

Enum HashType

Namespace: AetherUtils.Core.Security.TwoFactor

Assembly: AetherUtils.Core.dll

Represents the various hash types valid for two factor authentication keys.

```
public enum HashType
```

Extension Methods

EnumExtensions.ToDescriptionString<T>(T)

Fields

`Sha1 = 0`

Indicates the 2FA codes should use the SHA1 algorithm.

`Sha256 = 1`

Indicates the 2FA codes should use the SHA256 algorithm.

`Sha512 = 2`

Indicates the 2FA codes should use the SHA512 algorithm.

Class SetupCode

Namespace: AetherUtils.Core.Security.TwoFactor

Assembly: AetherUtils.Core.dll

Represents a setup code needed by the user to setup two factor authentication.

```
public sealed class SetupCode
```

Inheritance

object ← SetupCode

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Constructors

SetupCode(string, string)

Represents a setup code needed by the user to setup two factor authentication.

```
public SetupCode(string manualEntryKey, string qrCodeSetupImageUrl)
```

Parameters

`manualEntryKey` string

The secret key that can be manually entered in a 2FA app.

`qrCodeSetupImageUrl` string

The base64 string representing the QR code.

Properties

ManualEntryKey

The secret key that can be manually entered in a 2FA app.

```
public string ManualEntryKey { get; }
```

Property Value

string

QrCodeSetupImageUrl

The base64 string representing the QR code.

```
public string QrCodeSetupImageUrl { get; }
```

Property Value

string

Class TwoFactorAuth

Namespace: AetherUtils.Core.Security.TwoFactor

Assembly: AetherUtils.Core.dll

Provides methods to generate and validate two-factor authentication codes.

```
public class TwoFactorAuth
```

Inheritance

object ← TwoFactorAuth

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Constructors

TwoFactorAuth()

```
public TwoFactorAuth()
```

TwoFactorAuth(HashType)

Provides methods to generate and validate two-factor authentication codes.

```
public TwoFactorAuth(HashType hashType)
```

Parameters

`hashType` HashType

The HashTypefor the generated codes.

Methods

GeneratePinAtInterval(byte[], long, int)

This method is generally called via GetCurrentPin(string, bool)/>

```
public string GeneratePinAtInterval(byte[] accountSecretKey, long counter, int digits = 6)
```

Parameters

`accountSecretKey` byte[]

The account secret key as a byte array

`counter` long

The number of 30-second (by default) intervals since the unix epoch

`digits` int

The desired length of the returned PIN

Returns

string

A 'PIN' that is valid for the specified time interval

GeneratePinAtInterval(string, long, int, bool)

This method is generally called via GetCurrentPin(string, bool)

```
public string GeneratePinAtInterval(string accountSecretKey, long counter, int digits = 6,
bool secretIsBase32 = false)
```

Parameters

`accountSecretKey` string

The account secret key as a string

`counter` long

The number of 30-second (by default) intervals since the unix epoch

`digits` int

The desired length of the returned PIN

`secretIsBase32` bool

Flag saying if accountSecretKey is in Base32 format or original secret

Returns

string

A 'PIN' that is valid for the specified time interval

GenerateSetupInformation(ref TwoFactorUser, byte[])

Generate the setup information needed for a user to scan with their authenticator app.

```
public void GenerateSetupInformation(ref TwoFactorUser user, byte[] secretKey)
```

Parameters

`user` TwoFactorUser

The TwoFactorUser containing the setup information, as a reference.

`secretKey` byte[]

The secret key used to generate the setup information.

GetCurrentPin(byte[])

Get the PIN for current time; the same code that a 2FA app would generate for the current time. Do not validate directly against this as clock drift may cause a different PIN to be generated.

```
public string GetCurrentPin(byte[] accountSecretKey)
```

Parameters

`accountSecretKey` byte[]

Account Secret Key

Returns

string

A 6-digit PIN

GetCurrentPin(byte[], DateTime)

Get the PIN for current time; the same code that a 2FA app would generate for the current time. Do not validate directly against this as clock drift may cause a different PIN to be generated.

```
public string GetCurrentPin(byte[] accountSecretKey, DateTime now)
```

Parameters

`accountSecretKey` byte[]

Account Secret Key

`now` DateTime

The time you wish to generate the pin for

Returns

string

A 6-digit PIN

GetCurrentPin(string, bool)

Get the PIN for current time; the same code that a 2FA app would generate for the current time. Do not validate directly against this as clock drift may cause a different PIN to be generated than one you did a second ago.

```
public string GetCurrentPin(string accountSecretKey, bool secretIsBase32 = false)
```

Parameters

`accountSecretKey` string

Account Secret Key

`secretIsBase32` bool

Flag saying if accountSecretKey is in Base32 format or original secret

Returns

string

A 6-digit PIN

GetCurrentPin(string, DateTime, bool)

Get the PIN for current time; the same code that a 2FA app would generate for the current time. Do not validate directly against this as clock drift may cause a different PIN to be generated than one you did a second ago.

```
public string GetCurrentPin(string accountSecretKey, DateTime now, bool secretIsBase32
= false)
```

Parameters

`accountSecretKey` string

Account Secret Key

`now` DateTime

The time you wish to generate the pin for

`secretIsBase32` bool

Flag saying if accountSecretKey is in Base32 format or original secret

Returns

string

A 6-digit PIN

GetCurrentPins(byte[])

Get all the PINs that would be valid within the time window allowed for by the default clock drift.

```
public string[] GetCurrentPins(byte[] accountSecretKey)
```

Parameters

`accountSecretKey` byte[]

Account Secret Key

Returns

string[]

GetCurrentPins(byte[], int)

Get all the PINs that would be valid within the time window allowed for by the specified clock drift.

```
public string[] GetCurrentPins(byte[] accountSecretKey, int iterationOffset)
```

Parameters

`accountSecretKey` byte[]

Account Secret Key

`iterationOffset` int

The counter drift size you want to generate PINs for

Returns

string[]

GetCurrentPins(byte[], TimeSpan)

Get all the PINs that would be valid within the time window allowed for by the specified clock drift.

```
public string[] GetCurrentPins(byte[] accountSecretKey, TimeSpan timeTolerance)
```

Parameters

`accountSecretKey` byte[]

Account Secret Key

`timeTolerance` TimeSpan

The clock drift size you want to generate PINs for

Returns

string[]

GetCurrentPins(string, bool)

Get all the PINs that would be valid within the time window allowed for by the default clock drift.

```
public string[] GetCurrentPins(string accountSecretKey, bool secretIsBase32 = false)
```

Parameters

`accountSecretKey` string

Account Secret Key

`secretIsBase32` bool

Flag saying if accountSecretKey is in Base32 format or original secret

Returns

string[]

GetCurrentPins(string, TimeSpan, bool)

Get all the PINs that would be valid within the time window allowed for by the specified clock drift.

```
public string[] GetCurrentPins(string accountSecretKey, TimeSpan timeTolerance, bool
secretIsBase32 = false)
```

Parameters

`accountSecretKey` string

Account Secret Key

`timeTolerance` TimeSpan

The clock drift size you want to generate PINs for

`secretIsBase32` bool

Flag saying if accountSecretKey is in Base32 format or original secret

Returns

string[]

ValidateTwoFactorPin(byte[], string)

Given a PIN from a client, check if it is valid at the current time.

```
public bool ValidateTwoFactorPin(byte[] accountSecretKey, string twoFactorCodeFromClient)
```

Parameters

`accountSecretKey` byte[]

Account Secret Key

`twoFactorCodeFromClient` string

The PIN from the client

Returns

bool

True if PIN is currently valid

ValidateTwoFactorPin(byte[], string, int)

Given a PIN from a client, check if it is valid at the current time.

```
public bool ValidateTwoFactorPin(byte[] accountSecretKey, string twoFactorCodeFromClient,
int iterationOffset)
```

Parameters

`accountSecretKey` byte[]

Account Secret Key

`twoFactorCodeFromClient` string

The PIN from the client

`iterationOffset` int

The counter window within which to check to allow for clock drift between devices.

Returns

bool

True if PIN is currently valid

ValidateTwoFactorPin(byte[], string, TimeSpan)

Given a PIN from a client, check if it is valid at the current time.

```
public bool ValidateTwoFactorPin(byte[] accountSecretKey, string twoFactorCodeFromClient,
TimeSpan timeTolerance)
```

Parameters

`accountSecretKey` byte[]

Account Secret Key

`twoFactorCodeFromClient` string

The PIN from the client

`timeTolerance` TimeSpan

The time window within which to check to allow for clock drift between devices.

Returns

bool

True if PIN is currently valid

ValidateTwoFactorPin(string, string, bool)

Given a PIN from a client, check if it is valid at the current time.

```
public bool ValidateTwoFactorPin(string accountSecretKey, string twoFactorCodeFromClient,
bool secretIsBase32 = false)
```

Parameters

`accountSecretKey` string

Account Secret Key

`twoFactorCodeFromClient` string

The PIN from the client

`secretIsBase32` bool

Flag saying if accountSecretKey is in Base32 format or original secret

Returns

bool

True if PIN is currently valid

ValidateTwoFactorPin(string, string, int, bool)

Given a PIN from a client, check if it is valid at the current time.

```
public bool ValidateTwoFactorPin(string accountSecretKey, string twoFactorCodeFromClient,
int iterationOffset, bool secretIsBase32 = false)
```

Parameters

`accountSecretKey` string

Account Secret Key

`twoFactorCodeFromClient` string

The PIN from the client

`iterationOffset` int

The counter window within which to check to allow for clock drift between devices.

`secretIsBase32` bool

Flag saying if accountSecretKey is in Base32 format or original secret

Returns

bool

True if PIN is currently valid

ValidateTwoFactorPin(string, string, TimeSpan, bool)

Given a PIN from a client, check if it is valid at the current time.

```
public bool ValidateTwoFactorPin(string accountSecretKey, string twoFactorCodeFromClient,
TimeSpan timeTolerance, bool secretIsBase32 = false)
```

Parameters

`accountSecretKey` string

Account Secret Key

`twoFactorCodeFromClient` string

The PIN from the client

`timeTolerance` TimeSpan

The time window within which to check to allow for clock drift between devices.

`secretIsBase32` bool

Flag saying if accountSecretKey is in Base32 format or original secret

Returns

bool

True if PIN is currently valid

Class TwoFactorUser

Namespace: AetherUtils.Core.Security.TwoFactor

Assembly: AetherUtils.Core.dll

Represents a 2-factor authentication user.

```
public sealed class TwoFactorUser
```

Inheritance

object ← TwoFactorUser

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Constructors

TwoFactorUser(string, string)

Represents a 2-factor authentication user.

```
public TwoFactorUser(string issuer, string accountTitle)
```

Parameters

`issuer` string

The issuer of the 2FA tokens.

`accountTitle` string

The account title as shown in the user's authenticator app.

Properties

AccountTitle

The account title as shown in the user's authenticator app.

```
public string AccountTitle { get; }
```

Property Value

string

Issuer

The issuer of the 2FA tokens.

```
public string Issuer { get; }
```

Property Value

string

SetupInformation

The information needed for the user to set the account up in their authenticator app.

```
public SetupCode? SetupInformation { get; }
```

Property Value

SetupCode

Namespace AetherUtils.Core.Structs

Structs

ConfigOption

Represents a single configuration option and it's value. This option can contain a single-dimensional list indexer.

HashOptions

Represents the options used for performing hashing functions. If unspecified at struct creation, the HashAlgorithm defaults to SHA384 and the Encoding defaults to Base64.
Once created, this struct's options cannot be changed.

Pair<K, V>

Represents a generic key-value pair.

ReadOnlyPair<K, V>

Represents a generic key-value pair that is read-only; Once the pair is instantiated, the values cannot be changed.

TrimmedString

Represents a string that has new line characters (\n) inserted and has been formatted to be a specific length.

Struct ConfigOption

Namespace: AetherUtils.Core.Structs

Assembly: AetherUtils.Core.dll

Represents a single configuration option and it's value. This option can contain a single-dimensional list indexer.

```
public readonly struct ConfigOption
```

Inherited Members

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

Remarks

`var option = new ConfigOption("listOption[0]", 2);`
would allow you to change the value of the listOption at index 0 to the value 2.

Constructors

ConfigOption(string, object?)

```
public ConfigOption(string option, object? value)
```

Parameters

`option` string

`value` object

Properties

ArrayIndexExists

```
public bool ArrayIndexExists { get; }
```

Property Value

bool

ArrayIndexer

```
public int ArrayIndexer { get; }
```

Property Value

int

Name

```
public string Name { get; }
```

Property Value

string

Value

```
public object? Value { get; }
```

Property Value

object

Struct HashOptions

Namespace: AetherUtils.Core.Structs

Assembly: AetherUtils.Core.dll

Represents the options used for performing hashing functions. If unspecified at struct creation, the Hash Algorithm defaults to SHA384 and the Encoding defaults to Base64.
Once created, this struct's options cannot be changed.

```
public readonly struct HashOptions
```

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Constructors

HashOptions(ReadOnlyPair<int, int>)

```
public HashOptions(ReadOnlyPair<int, int> iterations)
```

Parameters

`iterations` ReadOnlyPair<int, int>

HashOptions(ReadOnlyPair<int, int>, HashEncoding)

```
public HashOptions(ReadOnlyPair<int, int> iterations, HashEncoding hashEncoding)
```

Parameters

`iterations` ReadOnlyPair<int, int>

`hashEncoding` HashEncoding

HashOptions(ReadOnlyPair<int, int>, HashAlgorithmName, HashEncoding)

```
public HashOptions(ReadOnlyPair<int, int> iterations, HashAlgorithmName hashAlgorithm,
HashEncoding hashEncoding)
```

Parameters

`iterations` ReadOnlyPair<int, int>

`hashAlgorithm` HashAlgorithmName

`hashEncoding` HashEncoding

HashOptions(int, ReadOnlyPair<int, int>, HashAlgorithmName)

```
public HashOptions(int saltLength, ReadOnlyPair<int, int> iterations,
HashAlgorithmName hashAlgorithm)
```

Parameters

`saltLength` int

`iterations` ReadOnlyPair<int, int>

`hashAlgorithm` HashAlgorithmName

HashOptions(int, ReadOnlyPair<int, int>, HashAlgorithmName, HashEncoding)

```
public HashOptions(int saltLength, ReadOnlyPair<int, int> iterations, HashAlgorithmName
hashAlgorithm, HashEncoding hashEncoding)
```

Parameters

`saltLength` int

`iterations` ReadOnlyPair<int, int>

`hashAlgorithm` HashAlgorithmName

`hashEncoding` HashEncoding

HashOptions(int, int)

```
public HashOptions(int minIterations, int maxIterations)
```

Parameters

`minIterations` int

`maxIterations` int

HashOptions(int, int, HashEncoding)

```
public HashOptions(int minIterations, int maxIterations, HashEncoding hashEncoding)
```

Parameters

`minIterations` int

`maxIterations` int

`hashEncoding` HashEncoding

HashOptions(int, int, int)

```
public HashOptions(int saltLength, int minIterations, int maxIterations)
```

Parameters

`saltLength` int

`minIterations` int

`maxIterations` int

HashOptions(int, int, int, int)

```
public HashOptions(int saltLength, int keySize, int minIterations, int maxIterations)
```

Parameters

`saltLength` int

`keySize` int

`minIterations` int

`maxIterations` int

HashOptions(int, int, int, int, HashAlgorithmName)

```
public HashOptions(int saltLength, int keySize, int minIterations, int maxIterations,
HashAlgorithmName hashAlgorithm)
```

Parameters

`saltLength` int

`keySize` int

`minIterations` int

`maxIterations` int

`hashAlgorithm` HashAlgorithmName

HashOptions(int, int, int, int, HashAlgorithmName, HashEncoding)

```
public HashOptions(int saltLength, int keySize, int minIterations, int maxIterations,
```

```
HashAlgorithmName hashAlgorithm, HashEncoding hashEncoding)
```

Parameters

`saltLength` int

`keySize` int

`minIterations` int

`maxIterations` int

`hashAlgorithm` HashAlgorithmName

`hashEncoding` HashEncoding

HashOptions(int, int, HashAlgorithmName)

```
public HashOptions(int minIterations, int maxIterations, HashAlgorithmName hashAlgorithm)
```

Parameters

`minIterations` int

`maxIterations` int

`hashAlgorithm` HashAlgorithmName

HashOptions(int, int, HashAlgorithmName, HashEncoding)

```
public HashOptions(int minIterations, int maxIterations, HashAlgorithmName hashAlgorithm,
HashEncoding hashEncoding)
```

Parameters

`minIterations` int

`maxIterations` int

`hashAlgorithm` HashAlgorithmName

`hashEncoding` HashEncoding

Properties

Encoding

The encoding scheme to use when hashing.

```
public HashEncoding Encoding { get; }
```

Property Value

HashEncoding

HashAlgorithm

The algorithm to use when hashing.

```
public HashAlgorithmName HashAlgorithm { get; }
```

Property Value

HashAlgorithmName

Iterations

The number of iterations to perform when hashing.
A random value between the minimum and maximum iterations is retrieved every time this property is called.

```
public int Iterations { get; }
```

Property Value

int

KeySize

The size (in bytes) of the hash key used when hashing.

```
public int KeySize { get; }
```

Property Value

int

SaltLength

The length (in bytes) used for the salt when hashing.

```
public int SaltLength { get; }
```

Property Value

int

Struct Pair<K, V>

Namespace: AetherUtils.Core.Structs

Assembly: AetherUtils.Core.dll

Represents a generic key-value pair.

```
public struct Pair<K, V>
```

Type Parameters

K

The Type for the key.

V

The Type for the value.

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Constructors

Pair(K?, V?)

Represents a generic key-value pair.

```
public Pair(K? key, V? value)
```

Parameters

key K

value V

Properties

Key

```
public K? Key { readonly get; set; }
```

Property Value

K

Value

```
public V? Value { readonly get; set; }
```

Property Value

V

Struct ReadOnlyPair<K, V>

Namespace: AetherUtils.Core.Structs

Assembly: AetherUtils.Core.dll

Represents a generic key-value pair that is read-only; Once the pair is instantiated, the values cannot be changed.

```
public readonly struct ReadOnlyPair<K, V>
```

Type Parameters

K

The Type for the key.

V

The Type for the value.

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Constructors

ReadOnlyPair(K?, V?)

Represents a generic key-value pair that is read-only; Once the pair is instantiated, the values cannot be changed.

```
public ReadOnlyPair(K? key, V? value)
```

Parameters

key K

value V

Fields

Key

```
public readonly K? Key
```

Field Value

K

Value

```
public readonly V? Value
```

Field Value

V

Struct TrimmedString

Namespace: AetherUtils.Core.Structs

Assembly: AetherUtils.Core.dll

Represents a string that has new line characters (\n) inserted and has been formatted to be a specific length.

```
public readonly struct TrimmedString
```

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Constructors

TrimmedString(string)

Create a new TrimmedString struct from the specified string.

```
public TrimmedString(string s)
```

Parameters

`s` string

The string to initialize the struct with.

Fields

Lines

The number of new line characters (\n) in the string.

```
public readonly int Lines
```

Field Value

int

String

The trimmed string.

```
public readonly string String
```

Field Value

string

Namespace AetherUtils.Core.Utility

Classes

ProportionalRandomSelector<T>

Represents a cryptographically strong, random item selector using proportional percentages.

SerializableDictionary<TKey, TValue>

Represents a generic key-value dictionary that is serializable via XML serialization.
This derived type from Dictionary<TKey, TValue> does not allow `null` for its values.

Class ProportionalRandomSelector<T>

Namespace: AetherUtils.Core.Utility

Assembly: AetherUtils.Core.dll

Represents a cryptographically strong, random item selector using proportional percentages.

```
public sealed class ProportionalRandomSelector<T> where T : notnull
```

Type Parameters

`T`

The type of item to select using this selector.

Inheritance

object ← ProportionalRandomSelector<T>

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Methods

AddPercentage(T, int)

Add an item and its percentage to this selector.

```
public void AddPercentage(T item, int percentage)
```

Parameters

`item` T

The item to add.

`percentage` int

The percentage indicating the likelihood of the item being selected.

SelectItem()

Get a random item from the selector using the percentages defined within.

```
public T? SelectItem()
```

Returns

T

A random item or `null` if an item could not be selected.

Class SerializableDictionary<TKey, TValue>

Namespace: AetherUtils.Core.Utility

Assembly: AetherUtils.Core.dll

Represents a generic key-value dictionary that is serializable via XML serialization.
This derived type from Dictionary<TKey, TValue> does not allow `null` for its values.

```
[Serializable]
public class SerializableDictionary<TKey, TValue> : Dictionary<TKey, TValue>,
IDictionary<TKey, TValue>, ICollection<KeyValuePair<TKey, TValue>>,
IReadOnlyDictionary<TKey, TValue>, IReadOnlyCollection<KeyValuePair<TKey, TValue>>,
IEnumerable<KeyValuePair<TKey, TValue>>, IDictionary, ICollection, IEnumerable,
IDeserializationCallback, ISerializable, IXmlSerializable where TKey : notnull where TValue
: notnull
```

Type Parameters

`TKey`

The Type for the keys in this dictionary.

`TValue`

The Type for the values in this dictionary.

Inheritance

object ← Dictionary<TKey, TValue> ← SerializableDictionary<TKey, TValue>

Implements

IDictionary<TKey, TValue>, ICollection<KeyValuePair<TKey, TValue>>, IReadOnlyDictionary<TKey, TValue>, IReadOnlyCollection<KeyValuePair<TKey, TValue>>, IEnumerable<KeyValuePair<TKey, TValue>>, IDictionary, ICollection, IEnumerable, IDeserializationCallback, ISerializable, IXmlSerializable

Inherited Members

Dictionary<TKey, TValue>.Add(TKey, TValue) , Dictionary<TKey, TValue>.Clear() , Dictionary<TKey, TValue>.ContainsKey(TKey) , Dictionary<TKey, TValue>.ContainsValue(TValue) , Dictionary<TKey, TValue>.EnsureCapacity(int) , Dictionary<TKey, TValue>.GetEnumerator() , Dictionary<TKey, TValue>.OnDeserialization(object) , Dictionary<TKey, TValue>.Remove(TKey) , Dictionary<TKey, TValue>.Remove(TKey, out TValue) , Dictionary<TKey, TValue>.TrimExcess() ,

Dictionary<TKey, TValue>.TrimExcess(int) , Dictionary<TKey, TValue>.TryAdd(TKey, TValue) , Dictionary<TKey, TValue>.TryGetValue(TKey, out TValue) , Dictionary<TKey, TValue>.Comparer , Dictionary<TKey, TValue>.Count , Dictionary<TKey, TValue>.this[TKey] , Dictionary<TKey, TValue>.Keys , Dictionary<TKey, TValue>.Values , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T) , CollectionExtensions.RenameKey<TKey, TValue>(IDictionary<TKey, TValue>, TKey, TKey)

Methods

GetSchema()

This method is reserved and should not be used. When implementing the `IXmlSerializable` interface, you should return null (`Nothing` in Visual Basic) from this method, and instead, if specifying a custom schema is required, apply the XmlSchemaProviderAttribute to the class.

```
public XmlSchema? GetSchema()
```

Returns

XmlSchema

An XmlSchema that describes the XML representation of the object that is produced by the WriteXml(XmlWriter) method and consumed by the ReadXml(XmlReader) method.

ReadXml(XmlReader)

Generates an object from its XML representation.

```
public void ReadXml(XmlReader reader)
```

Parameters

`reader` XmlReader

The XmlReader stream from which the object is deserialized.

WriteXml(XmlWriter)

Converts an object into its XML representation.

```
public void WriteXml(XmlWriter writer)
```

Parameters

`writer` XmlWriter

The XmlWriter stream to which the object is serialized.

Namespace AetherUtils.Core.WinForms

Classes

IconListManager

Maintains a list of currently added file extensions

IconReader

Provides static methods to read system icons for both folders and files. i.e.,

```
IconReader.GetFileIcon("c:\\general.xls");
```

Shell32

Wraps necessary Shell32.dll structures and functions required to retrieve Icon Handles using SHGetFileInfo. Code courtesy of MSDN Cold Rooster Consulting case study.

User32

Wraps necessary functions imported from User32.dll. Code courtesy of MSDN Cold Rooster Consulting example.

Structs

Shell32.BROWSEINFO

Shell32.ITEMIDLIST

Shell32.SHFILEINFO

Shell32.SHITEMID

Enums

IconReader.FolderType

Options to specify whether folders should be in the open or closed state.

IconReader.IconSize

Options to specify the size of icons to return.

Class IconListManager

Namespace: AetherUtils.Core.WinForms

Assembly: AetherUtils.Core.dll

Maintains a list of currently added file extensions

```
public class IconListManager
```

Inheritance

object ← IconListManager

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Constructors

IconListManager(ImageList, IconSize)

Creates an instance of `IconListManager` that will add icons to a single `ImageList` using the specified `IconSize`.

```
public IconListManager(ImageList imageList, IconReader.IconSize iconSize)
```

Parameters

`imageList` ImageList

`ImageList` to add icons to.

`iconSize` IconReader.IconSize

Size to use (either 32 or 16 pixels).

IconListManager(ImageList, ImageList)

Creates an instance of IconListManager that will add icons to two `ImageList` types. The two image lists are intended to be one for large icons, and the other for small icons.

```
public IconListManager(ImageList smallImageList, ImageList largeImageList)
```

Parameters

`smallImageList` ImageList

The `ImageList` that will hold small icons.

`largeImageList` ImageList

The `ImageList` that will hold large icons.

Methods

AddFileIcon(string)

Called publicly to add a file's icon to the ImageList.

```
public int AddFileIcon(string filePath)
```

Parameters

`filePath` string

Full path to the file.

Returns

int

Integer of the icon's position in the ImageList

ClearLists()

Clears all ImageLists that IconListManager is managing.

```
public void ClearLists()
```

Class IconReader

Namespace: AetherUtils.Core.WinForms

Assembly: AetherUtils.Core.dll

Provides static methods to read system icons for both folders and files. i.e.,

```
IconReader.GetFileIcon("c:\\general.xls");
```

```
public class IconReader
```

Inheritance

object ← IconReader

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Methods

GetFileIcon(string, IconSize, bool)

Returns an icon for a given file - indicated by the name parameter.

```
public static Icon GetFileIcon(string name, IconReader.IconSize size, bool linkOverlay)
```

Parameters

`name` string

Pathname for file.

`size` IconReader.IconSize

Large or small

`linkOverlay` bool

Whether to include the link icon

Returns

Icon

System.Drawing.Icon

GetFolderIcon(IconSize, FolderType)

Used to access system folder icons.

```
public static Icon GetFolderIcon(IconReader.IconSize size, IconReader.FolderType folderType)
```

Parameters

`size` IconReader.IconSize

Specify large or small icons.

`folderType` IconReader.FolderType

Specify open or closed FolderType.

Returns

Icon

System.Drawing.Icon

Enum IconReader.FolderType

Namespace: AetherUtils.Core.WinForms

Assembly: AetherUtils.Core.dll

Options to specify whether folders should be in the open or closed state.

```
public enum IconReader.FolderType
```

Extension Methods

EnumExtensions.ToDescriptionString<T>(T)

Fields

`Closed = 1`

Specify closed folder.

`Open = 0`

Specify open folder.

Enum IconReader.IconSize

Namespace: AetherUtils.Core.WinForms

Assembly: AetherUtils.Core.dll

Options to specify the size of icons to return.

```
public enum IconReader.IconSize
```

Extension Methods

EnumExtensions.ToDescriptionString<T>(T)

Fields

`Large = 0`

Specify large icon - 32 pixels by 32 pixels.

`Small = 1`

Specify small icon - 16 pixels by 16 pixels.

Class Shell32

Namespace: AetherUtils.Core.WinForms

Assembly: AetherUtils.Core.dll

Wraps necessary Shell32.dll structures and functions required to retrieve Icon Handles using SHGetFileInfo. Code courtesy of MSDN Cold Rooster Consulting case study.

```
public class Shell32
```

Inheritance

object ← Shell32

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Fields

BIF_BROWSEFORCOMPUTER

```
public const uint BIF_BROWSEFORCOMPUTER = 4096
```

Field Value

uint

BIF_BROWSEFORPRINTER

```
public const uint BIF_BROWSEFORPRINTER = 8192
```

Field Value

uint

BIF_BROWSEINCLUDEFILES

```
public const uint BIF_BROWSEINCLUDEFILES = 16384
```

Field Value

uint

BIF_BROWSEINCLUDEURLS

```
public const uint BIF_BROWSEINCLUDEURLS = 128
```

Field Value

uint

BIF_DONTGOBELOWDOMAIN

```
public const uint BIF_DONTGOBELOWDOMAIN = 2
```

Field Value

uint

BIF_EDITBOX

```
public const uint BIF_EDITBOX = 16
```

Field Value

uint

BIF_NEWDIALOGSTYLE

```
public const uint BIF_NEWDIALOGSTYLE = 64
```

Field Value

uint

BIF_RETURNFSANCESTORS

```
public const uint BIF_RETURNFSANCESTORS = 8
```

Field Value

uint

BIF_RETURNONLYFSDIRS

```
public const uint BIF_RETURNONLYFSDIRS = 1
```

Field Value

uint

BIF_SHAREABLE

```
public const uint BIF_SHAREABLE = 32768
```

Field Value

uint

BIF_STATUSTEXT

```
public const uint BIF_STATUSTEXT = 4
```

Field Value

uint

BIF_USENEWUI

```
public const uint BIF_USENEWUI = 80
```

Field Value

uint

BIF_VALIDATE

```
public const uint BIF_VALIDATE = 32
```

Field Value

uint

FILE_ATTRIBUTE_DIRECTORY

```
public const uint FILE_ATTRIBUTE_DIRECTORY = 16
```

Field Value

uint

FILE_ATTRIBUTE_NORMAL

```
public const uint FILE_ATTRIBUTE_NORMAL = 128
```

Field Value

uint

MAX_PATH

```
public const int MAX_PATH = 256
```

Field Value

int

SHGFI_ADDOVERLAYS

```
public const uint SHGFI_ADDOVERLAYS = 32
```

Field Value

uint

SHGFI_ATTRIBUTES

```
public const uint SHGFI_ATTRIBUTES = 2048
```

Field Value

uint

SHGFI_ATTR_SPECIFIED

```
public const uint SHGFI_ATTR_SPECIFIED = 131072
```

Field Value

uint

SHGFI_DISPLAYNAME

```
public const uint SHGFI_DISPLAYNAME = 512
```

Field Value

uint

SHGFI_EXETYPE

```
public const uint SHGFI_EXETYPE = 8192
```

Field Value

uint

SHGFI_ICON

```
public const uint SHGFI_ICON = 256
```

Field Value

uint

SHGFI_ICONLOCATION

```
public const uint SHGFI_ICONLOCATION = 4096
```

Field Value

uint

SHGFI_LARGEICON

```
public const uint SHGFI_LARGEICON = 0
```

Field Value

uint

SHGFI_LINKOVERLAY

```
public const uint SHGFI_LINKOVERLAY = 32768
```

Field Value

uint

SHGFI_OPENICON

```
public const uint SHGFI_OPENICON = 2
```

Field Value

uint

SHGFI_OVERLAYINDEX

```
public const uint SHGFI_OVERLAYINDEX = 64
```

Field Value

uint

SHGFI_PIDL

```
public const uint SHGFI_PIDL = 8
```

Field Value

uint

SHGFI_SELECTED

```
public const uint SHGFI_SELECTED = 65536
```

Field Value

uint

SHGFI_SHELLICONSIZE

```
public const uint SHGFI_SHELLICONSIZE = 4
```

Field Value

uint

SHGFI_SMALLICON

```
public const uint SHGFI_SMALLICON = 1
```

Field Value

uint

SHGFI_SYSICONINDEX

```
public const uint SHGFI_SYSICONINDEX = 16384
```

Field Value

uint

SHGFI_TYPENAME

```
public const uint SHGFI_TYPENAME = 1024
```

Field Value

uint

SHGFI_USEFILEATTRIBUTES

```
public const uint SHGFI_USEFILEATTRIBUTES = 16
```

Field Value

uint

Methods

SHGetFileInfo(string, uint, ref SHFILEINFO, uint, uint)

```
public static extern nint SHGetFileInfo(string pszPath, uint dwFileAttributes, ref
Shell32.SHFILEINFO psfi, uint cbFileInfo, uint uFlags)
```

Parameters

pszPath string

dwFileAttributes uint

psfi Shell32.SHFILEINFO

cbFileInfo uint

uFlags uint

Returns

nint

Struct Shell32.BROWSEINFO

Namespace: AetherUtils.Core.WinForms

Assembly: AetherUtils.Core.dll

```
public struct Shell32.BROWSEINFO
```

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Fields

hwndOwner

```
public nint hwndOwner
```

Field Value

nint

iImage

```
public nint iImage
```

Field Value

nint

lParam

```
public int lParam
```

Field Value

int

lpfn

```
public nint lpfn
```

Field Value

nint

lpszTitle

```
public string lpszTitle
```

Field Value

string

pidlRoot

```
public nint pidlRoot
```

Field Value

nint

pszDisplayName

```
public nint pszDisplayName
```

Field Value

nint

ulFlags

```
public uint ulFlags
```

Field Value

uint

Struct Shell32.ITEMIDLIST

Namespace: AetherUtils.Core.WinForms

Assembly: AetherUtils.Core.dll

```
public struct Shell32.ITEMIDLIST
```

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Fields

mkid

```
public Shell32.SHITEMID mkid
```

Field Value

Shell32.SHITEMID

Struct Shell32.SHFILEINFO

Namespace: AetherUtils.Core.WinForms

Assembly: AetherUtils.Core.dll

```
public struct Shell32.SHFILEINFO
```

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Fields

NAMESIZE

```
public const int NAMESIZE = 80
```

Field Value

int

dwAttributes

```
public uint dwAttributes
```

Field Value

uint

hIcon

```
public nint hIcon
```

Field Value

nint

iIcon

```
public int iIcon
```

Field Value

int

szDisplayName

```
public string szDisplayName
```

Field Value

string

szTypeName

```
public string szTypeName
```

Field Value

string

Struct Shell32.SHITEMID

Namespace: AetherUtils.Core.WinForms

Assembly: AetherUtils.Core.dll

```
public struct Shell32.SHITEMID
```

Inherited Members
ValueType.Equals(object) , ValueType.GetHashCode() , ValueType.ToString() , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object)

Fields

abID

```
public byte[] abID
```

Field Value

byte[]

cb

```
public ushort cb
```

Field Value

ushort

Class User32

Namespace: AetherUtils.Core.WinForms

Assembly: AetherUtils.Core.dll

Wraps necessary functions imported from User32.dll. Code courtesy of MSDN Cold Rooster Consulting example.

```
public class User32
```

Inheritance

object ← User32

Inherited Members

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

Extension Methods

TypeExtensions.CanSerializeJson<T>(T) , TypeExtensions.CanSerializeXml<T>(T) , TypeExtensions.SerializeJson<T>(T) , TypeExtensions.SerializeXml<T>(T)

Methods

DestroyIcon(nint)

Provides access to function required to delete handle. This method is used internally and is not required to be called separately.

```
public static extern int DestroyIcon(nint hIcon)
```

Parameters

`hIcon` nint

Pointer to icon handle.

Returns

int

N/A