Namespace Checker

Classes

Firewall

Class for checking the firewall.

Class Firewall

Namespace: Checker

Assembly: Checker.dll

Class for checking the firewall.

```
public static class Firewall
```

Inheritance

object ← Firewall

Inherited Members

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

Methods

CheckIpAndPort(string, int)

Check if a port on a ip is open.

```
public static bool CheckIpAndPort(string ip, int portNumber)
```

Parameters

`ip` string

IP to check

`portNumber` int

Port to check.

Returns

bool

State if True or False.

PingIp(string)

Pings an IP.

```
public static bool PingIp(string ip)
```

Parameters

`ip` string

IP to ping.

Returns

bool

True or false.

Namespace Converter

Classes

DateTimeConverter

Class for converting DateTime objects.

Enums

DateTimeConverter.DateTimeModes

Enum for DateTime modes. Represents year, month or day.

Class DateTimeConverter

Namespace: Converter

Assembly: Converter.dll

Class for converting DateTime objects.

```
public static class DateTimeConverter
```

Inheritance

object ← DateTimeConverter

Inherited Members

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

Methods

SplitDateByMode(DateTime, DateTimeModes)

Extracts the Year, Month or Day depending on DateTimeModes Enum.

```
public static int SplitDateByMode(DateTime dt, DateTimeConverter.DateTimeModes mode)
```

Parameters

`dt` DateTime

Source DateTime Object.

`mode` DateTimeConverter.DateTimeModes

Mode DateTimeModes.Year, DateTimeModes.Month oder DateTimeModes.Day.

Returns

int

Year, Month or day as Integer.

Enum DateTimeConverter.DateTimeModes

Namespace: Converter

Assembly: Converter.dll

Enum for DateTime modes. Represents year, month or day.

```
public enum DateTimeConverter.DateTimeModes
```

Fields

`Day = 2`

The day

`Month = 1`

The month

`Year = 0`

The year

Namespace Extensions

Classes

DateTimeExtensions

Class DateTimeExtensions.

EnumerableExtensions

Class for IEnumerable Extensions

StringExtensions

Class StringExtensions.

Class DateTimeExtensions

Namespace: Extensions

Assembly: Extensions.dll

Class DateTimeExtensions.

```
public static class DateTimeExtensions
```

Inheritance

object ← DateTimeExtensions

Inherited Members

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

Methods

ConvertDateTimeToString(DateTime)

Converts a given DateTime Object to yyyy-MM-dd HH:mm:ssZ.

```
public static string ConvertDateTimeToString(this DateTime dt)
```

Parameters

`dt` DateTime

DateTime Object.

Returns

string

System.String.

ConvertDateToNumeric(DateTime)

Converts a given DateTime Object to yyyMMdd.

```
public static int ConvertDateToNumeric(this DateTime dt)
```

Parameters

`dt` DateTime

DateTime Object

Returns

int

Integer numeric DateTime

Class EnumerableExtensions

Namespace: Extensions

Assembly: Extensions.dll

Class for IEnumerable Extensions

```
public static class EnumerableExtensions
```

Inheritance

object ← EnumerableExtensions

Inherited Members

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

Methods

IsEmpty<T>(IEnumerable<T>?)

Checks if the IEnumerable is null or empty

```
public static bool IsEmpty<T>(this IEnumerable<T>? source)
```

Parameters

source IEnumerable<T>

IEnumeration to check.

Returns

bool

true or false

Type Parameters

T

Type of Source

IsNotEmpty<T>(IEnumerable<T>?)

Checks if the IEnumerable is not null and not empty

```
public static bool IsNotEmpty<T>(this IEnumerable<T>? source)
```

Parameters

`source` IEnumerable<T>

IEnumeration to check

Returns

bool

true or false

Type Parameters

`T`

Type of Source

Class StringExtensions

Namespace: Extensions

Assembly: Extensions.dll

Class StringExtensions.

```
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

GetSalutationText(string)

Returns a salutation based on a given gender.

```
public static string GetSalutationText(this string gender)
```

Parameters

gender string

Gender

Returns

string

Herr oder Frau

ReturnGenderId(string)

Returns a integer based on a given gender.

```
public static int ReturnGenderId(this string gender)
```

Parameters

`gender` string

Gender

Returns

int

Male = 1, Female = 2, Unknown = -1

Namespace Generators

Classes

DataTableGenerator<T>

Class for generating DataTables from a List of Model objects.

Hash

Some methods for computing and decoding Hash.

TemporaryDirectory

A class to create a temporary directory.

TemporaryFile

Generates a temporary file.

Class DataTableGenerator<T>

Namespace: Generators

Assembly: Generators.dll

Class for generating DataTables from a List of Model objects.

```
public class DataTableGenerator<T>
```

Type Parameters

T

Modeltyp

Inheritance

object ← DataTableGenerator<T>

Inherited Members

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

Constructors

DataTableGenerator(ILogger<DataTableGenerator<T>>)

Constructor

```
public DataTableGenerator(ILogger<DataTableGenerator<T>> logger)
```

Parameters

logger ILogger<DataTableGenerator<T>>

Class logger

Methods

GenerateDataTableFromModelList(IList<T>, bool)

Generates a DataTable from a List of Model objects.

```
public DataTable GenerateDataTableFromModelList(IList<T> modelList, bool withId)
```

Parameters

`modelList` IList<T>

List model

`withId` bool

Should a ID Field generated.

Returns

DataTable

DataTable

Class Hash

Namespace: Generators

Assembly: Generators.dll

Some methods for computing and decoding Hash.

```
public static class Hash
```

Inheritance

object ← Hash

Inherited Members

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

Methods

GetHashString(string)

Computes the SHA256 hash for the input string and returns it as a hexadecimal string.

```
public static string GetHashString(string inputString)
```

Parameters

`inputString` string

The string to be hashed.

Returns

string

The computed hash as a hexadecimal string.

Exceptions

TargetInvocationException

On the .NET Framework 4.6.1 and earlier versions only: The algorithm was used with Federal Information Processing Standards (FIPS) mode enabled, but is not FIPS compatible.

EncoderFallbackException

A fallback occurred (for more information, see Character Encoding in .NET) -and- EncoderFallback is set to EncoderExceptionFallback.

ObjectDisposedException

The object has already been disposed.

ArgumentOutOfRangeException

Enlarging the value of this instance would exceed MaxCapacity.

Class TemporaryDirectory

Namespace: Generators

Assembly: Generators.dll

A class to create a temporary directory.

```
public static class TemporaryDirectory
```

Inheritance

object ← TemporaryDirectory

Inherited Members

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

Methods

GetTemporaryDirectory()

Erstellt einen temporären Ordner und gibt den Pfad zurück.

```
public static string GetTemporaryDirectory()
```

Returns

string

Pfad zum temporären Ordner.

Class TemporaryFile

Namespace: Generators

Assembly: Generators.dll

Generates a temporary file.

```
public sealed class TemporaryFile : IDisposable
```

Inheritance

object ← TemporaryFile

Implements

IDisposable

Inherited Members

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

Constructors

TemporaryFile()

Initializes a new instance of the TemporaryFile class.

```
public TemporaryFile()
```

TemporaryFile(string)

Initializes a new instance of the TemporaryFile class.

```
public TemporaryFile(string directory)
```

Parameters

`directory` string

The directory.

Properties

FilePath

Gets the file path.

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

Property Value

string

The file path.

Methods

Dispose()

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

```
public void Dispose()
```

~TemporaryFile()

Finalizes an instance of the TemporaryFile class.

```
protected ~TemporaryFile()
```

Namespace Patterns

Classes

Pipeline<T>

Class Pipeline. Used to define the pipeline. More on https://medium.com/@martinstm/pipeline-pattern-c-e01e2dd7238c

Result

Base class for Result pattern.

Result<T>

Class Result.

Interfaces

IPipeline<T>

This Interface is used to define the pipeline. More Details on: https://medium.com/@martinstm/pipeline-pattern-c-e01e2dd7238c

IStep<T>

Interface IStep. It is used to define the step in the pipeline.

Interface IPipeline<T>

Namespace: Patterns

Assembly: Patterns.dll

This Interface is used to define the pipeline. More Details on: https://medium.com/@martinstm/pipeline-pattern-c-e01e2dd7238c

```
public interface IPipeline<T>
```

Type Parameters

T

Given Type

Properties

Name

Gets or sets the name of the Pipleine.

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

Property Value

string

The name.

Steps

Gets the steps.

```
IReadOnlyCollection<IStep<T>> Steps { get; }
```

Property Value

IReadOnlyCollection<IStep<T>>

The steps.

Methods

StartAsync(T)

Starts the asynchronous.

```
Task<T> StartAsync(T data)
```

Parameters

`data` T

The data.

Returns

Task<T>

Task<T>.

WithStep(IStep<T>)

Adds the step.

```
void WithStep(IStep<T> step)
```

Parameters

`step` IStep<T>

The step.

Interface IStep<T>

Namespace: Patterns

Assembly: Patterns.dll

Interface IStep. It is used to define the step in the pipeline.

```
public interface IStep<T>
```

Type Parameters

T

Given Type

Methods

ExecuteAsync(T)

Executes the asynchronous.

```
Task<T> ExecuteAsync(T data)
```

Parameters

data T

The data.

Returns

Task<T>

Task<T>.

Class Pipeline<T>

Namespace: Patterns

Assembly: Patterns.dll

Class Pipeline. Used to define the pipeline. More on https://medium.com/@martinstm/pipeline-pattern-c-e01e2dd7238c

```
public class Pipeline<T> : IPipeline<T> where T : class
```

Type Parameters

T

Given Type

Inheritance

object ← Pipeline<T>

Implements

IPipeline<T>

Inherited Members

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

Constructors

Pipeline(string, ILogger<Pipeline<T>>)

Initializes a new instance of the Pipeline<T> class.

```
public Pipeline(string name, ILogger<Pipeline<T>> logger)
```

Parameters

name string

The name.

`logger` ILogger<Pipeline<T>>

Instance logger

Properties

Name

Gets or sets the name of the Pipleine.

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

Property Value

string

The name.

Steps

Gets the steps.

```
public IReadOnlyCollection<IStep<T>> Steps { get; }
```

Property Value

IReadOnlyCollection<IStep<T>>

The steps.

Methods

StartAsync(T)

Start as an asynchronous operation.

```
public Task<T> StartAsync(T data)
```

Parameters

`data` T

The data.

Returns

Task<T>

A Task<T> representing the asynchronous operation.

WithStep(IStep<T>)

Adds the step.

```
public void WithStep(IStep<T> step)
```

Parameters

`step` IStep<T>

The step.

Class Result

Namespace: Patterns

Assembly: Patterns.dll

Base class for Result pattern.

```
public class Result
```

Inheritance

object ← Result

Derived

Result<T>

Inherited Members

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

Constructors

Result(bool, string)

Initializes a new instance of the Result class.

```
protected Result(bool isSuccess, string errorMessage)
```

Parameters

`isSuccess` bool

if set to `true` [is success].

`errorMessage` string

The error message.

Properties

ErrorMessage

Gets the error message.

```
public string ErrorMessage { get; }
```

Property Value

string

The error message.

IsSuccess

Gets a value indicating whether this instance is success.

```
public bool IsSuccess { get; }
```

Property Value

bool

`true` if this instance is success; otherwise, `false`.

Methods

Failure(string)

Failures the specified error message.

```
public static Result Failure(string errorMessage)
```

Parameters

`errorMessage` string

The error message.

Returns

Result

Result.

Success()

Successes this instance.

```
public static Result Success()
```

Returns

Result

Result.

Class Result<T>

Namespace: Patterns

Assembly: Patterns.dll

Class Result.

```
public class Result<T> : Result
```

Type Parameters

T

Inheritance

object ← Result ← Result<T>

Inherited Members

Result.IsSuccess , Result.ErrorMessage , Result.Success() , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Value

Gets the value.

```
public T Value { get; }
```

Property Value

T

The value.

Methods

Failure(string)

Failures the specified error message.

```
public static Result<T> Failure(string errorMessage)
```

Parameters

`errorMessage` string

The error message.

Returns

Result<T>

Result<T>.

Success(T)

Successes the specified value.

```
public static Result<T> Success(T value)
```

Parameters

`value` T

The value.

Returns

Result<T>

Result<T>.

See Also

https://medium.com/@davisaac8/an-alternative-to-try-catch-in-c-b0e5dfafa910

Namespace Services

Classes

CsvService

Service for Writing a CSV.

EmailService

Service for sending emails.

WebDavService

Service for using WebDav.

Interfaces

ICsvService

Interface ICsvService

IEmailService

Interface IEmailService

IWebDavService

Interface IWebDavService

Class CsvService

Namespace: Services

Assembly: Services.dll

Service for Writing a CSV.

```
public class CsvService : ICsvService
```

Inheritance

object ← CsvService

Implements

ICsvService

Inherited Members

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

Constructors

CsvService(ILogger<CsvService>)

Constructor

```
public CsvService(ILogger<CsvService> logger)
```

Parameters

`logger` ILogger<CsvService>

Class logger

Methods

Read<T>(string, string, ClassMap<T>, string)

Reads the specified target name.

```
public IList<T> Read<T>(string targetName, string delimiter, ClassMap<T> map,
string culture)
```

Parameters

`targetName` string

Name of the target.

`delimiter` string

The delimiter, like: ";"

`map` ClassMap<T>

The map.

`culture` string

A culture string like 'en-US'.

Returns

IList<T>

List<T>.

Type Parameters

`T`

Exceptions

ArgumentNullException

WriteAsync<T>(IList<T>, string, string, string)

Writes the CSV asynchronous.

```
public Task WriteAsync<T>(IList<T> list, string targetName, string delimiter,
string culture)
```

Parameters

`list` IList<T>

Listen

`targetName` string

Path to CSV file.

`delimiter` string

Delimiter, zB. ";"

`culture` string

A culture string like 'en-US'.

Returns

Task

Asynchroner Task.

Type Parameters

`T`

Modeltype

Exceptions

ArgumentException

If the given `list` is empty.

ArgumentNullException

If the `list` or `targetName` is empty

UnauthorizedAccessException

Access is denied for `list` or `targetName`.

SecurityException

The caller does not have the required permission.

DirectoryNotFoundException

The specified path in `targetName` is invalid (for example, it is on an unmapped drive).

IOException

`targetName` includes an incorrect or invalid syntax for file name, directory name, or volume label syntax.

Class EmailService

Namespace: Services

Assembly: Services.dll

Service for sending emails.

```
public class EmailService : IEmailService
```

Inheritance

object ← EmailService

Implements

IEmailService

Inherited Members

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

Constructors

EmailService(ILogger<EmailService>, IConfiguration)

Constructor for EmailService

```
public EmailService(ILogger<EmailService> logger, IConfiguration configuration)
```

Parameters

`logger` ILogger<EmailService>

Class logger.

`configuration` IConfiguration

The Configuration object.

Methods

SendMessageAsync(MimeMessage)

Method for sending an email..

```
public Task SendMessageAsync(MimeMessage message)
```

Parameters

`message` MimeMessage

MimeMessage.

Returns

Task

Exceptions

ArgumentNullException

`message` ist null.

Interface ICsvService

Namespace: Services

Assembly: Services.dll

Interface ICsvService

```
public interface ICsvService
```

Methods

Read<T>(string, string, ClassMap<T>, string)

Reads the specified target name.

```
IList<T> Read<T>(string targetName, string delimiter, ClassMap<T> map, string culture)
```

Parameters

`targetName` string

Name of the target.

`delimiter` string

The delimiter.

`map` ClassMap<T>

Class Map

`culture` string

Taeget culture like en-US

Returns

IList<T>

List<T>.

Type Parameters

T

WriteAsync<T>(IList<T>, string, string, string)

Writes the asynchronous.

```
Task WriteAsync<T>(IList<T> list, string targetName, string delimiter, string culture)
```

Parameters

list IList<T>

The list.

targetName string

Name of the target.

delimiter string

The delimiter.

culture string

Target culture like en-US

Returns

Task

Task.

Type Parameters

T

Interface IEmailService

Namespace: Services

Assembly: Services.dll

Interface IEmailService

```
public interface IEmailService
```

Methods

SendMessageAsync(MimeMessage)

Sends the message asynchronous.

```
Task SendMessageAsync(MimeMessage message)
```

Parameters

`message` MimeMessage

The message.

Returns

Task

Task.

Interface IWebDavService

Namespace: Services

Assembly: Services.dll

Interface IWebDavService

```
public interface IWebDavService
```

Methods

DeleteFileAsync(string)

Deletes the file asynchronous.

```
Task<bool> DeleteFileAsync(string remoteFilepath)
```

Parameters

`remoteFilepath` string

The remote filepath.

Returns

Task<bool>

Task<System.Boolean>.

DownloadFileAsync(string, string)

Downloads the file asynchronous.

```
Task<bool> DownloadFileAsync(string remoteFilepath, string localFilepath)
```

Parameters

`remoteFilepath` string

The remote filepath.

`localFilepath` string

The local filepath.

Returns

Task<bool>

Task<System.Boolean>.

GetParams()

Gets the wd parameters.

```
WebDavClientParams GetParams()
```

Returns

WebDavClientParams

WebDavClientParams.

UploadFileAsync(string, string)

Uploads the file asynchronous.

```
Task<bool> UploadFileAsync(string localFilepath, string remoteFilepath)
```

Parameters

`localFilepath` string

The local filepath.

`remoteFilepath` string

The remote filepath.

Returns

Task<bool>

Task<System.Boolean>.

Class WebDavService

Namespace: Services

Assembly: Services.dll

Service for using WebDav.

```
public class WebDavService : IWebDavService
```

Inheritance

object ← WebDavService

Implements

IWebDavService

Inherited Members

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

Constructors

WebDavService(IConfiguration, ILogger<WebDavService>)

Constructor for WebDavServiceOptions.

```
public WebDavService(IConfiguration configuration, ILogger<WebDavService> logger)
```

Parameters

`configuration` IConfiguration

IConfiguration

`logger` ILogger<WebDavService>

logger

Methods

DeleteFileAsync(string)

Deletes a file from server.

```
public Task<bool> DeleteFileAsync(string remoteFilepath)
```

Parameters

`remoteFilepath` string

Path to file.

Returns

Task<bool>

True oder False.

DownloadFileAsync(string, string)

Downloads a file.

```
public Task<bool> DownloadFileAsync(string remoteFilepath, string localFilepath)
```

Parameters

`remoteFilepath` string

Path where the file should be placed..

`localFilepath` string

Local File Path.

Returns

Task<bool>

True oder False, jenachdem ob erfolgreich.

Exceptions

Exception

Condition.

ArgumentException

Wenn *remoteFilepath* oder *localFilepath* null ist.

UploadFileAsync(string, string)

Uploads a file

```
public Task<bool> UploadFileAsync(string localFilepath, string remoteFilepath)
```

Parameters

`localFilepath` string

Local Filepath.

`remoteFilepath` string

Remote Path where the file should be placed.

Returns

Task<bool>

Exceptions

DirectoryNotFoundException

The specified path `localFilepath` or `remoteFilepath` is invalid, (for example, it is on an unmapped drive).

IOException

An I/O error occurred while opening the file.

UnauthorizedAccessException

`localFilepath` or `remoteFilepath` specified a directory. -or- The caller does not have the required permission.

FileNotFoundException

The file specified in `localFilepath` or `remoteFilepath` was not found.

ArgumentException

.`localFilepath` or `remoteFilepath` is a zero-length string, contains only white space, or contains one or more invalid characters. You can query for invalid characters by using the GetInvalidPathChars() method.

NotSupportedException

`localFilepath` or `remoteFilepath` is in an invalid format.

ArgumentNullException

`localFilepath` or `remoteFilepath` is null.

PathTooLongException

The specified path, file name, or both exceed the system-defined maximum length.